分类号 TP311 密 级 公开

UDC 600 学 号 20042402007

重庆大学

硕士学位论文

论文题目：聊天机器人知识库自动抽取算法的研究与实现

论文作者：黄际洲

指导教师姓名、职称、工作单位：杨 丹 教授 重庆大学

申请学位级别：硕士 专业名称：软件工程

论文提交日期：2006 年 10 月 25 日 答辩日期：2006 年 11 月 26 日

学位授予单位：重庆大学 授位日期：2006 年 月 日

答辩委员会主席：朱宗元 教授级高工

论文评阅人：吴 渝 教授 祝伟华 副教授

2006 年 10 月 25 日

摘 要

聊天机器人(chatbot)是一种通过自然语言同人进行交流的人机交流对话智能系统(conversational agent)，人机之间的对话交流往往都局限于某个领域或者主题。聊天机器人一般都基于刺激——反应原理：用户提出一个问题，聊天机器人回答问题；或者用户做出某些评论，聊天机器人做出相应的反应。为了增加对话的连续性，聊天机器人往往还会在聊天过程中主动向用户提出新的问题。通常，聊天机器人都含有一个聊天知识库以及对话控制模块，聊天知识库就像聊天机器人的大脑，存储着回复用户输入的聊天知识，而对话控制模块则用于控制对话进程。一般而言，典型的聊天知识库都由一系列的模板组成，模板用于匹配用户的输入并根据一定的规则产生相应的机器人回复。但是目前用在聊天机器人中的模板，都是由人来手工构建的，因此聊天知识库的构建是一件费时费力的工作，并且缺乏灵活性，由此导致这种人工书写的方法在应用到新的知识领域或者新语言时，困难重重，比如一旦聊天机器人的知识领域发生了变化，就需要重新构建知识库。

本文提出并实现了一种新颖的聊天知识库构建方法，通过该方法，就能自动从在线论坛中抽取出形式为<帖子标题，回复>对的聊天知识。使用该方法，就能快速、有效地为聊天机器人构建某个领域的聊天知识。本文介绍的这种方法，是一种瀑布式模型(cascaded framework)，给定一个论坛，经过以下步骤的处理，就能自动从中抽取出高质量的<帖子标题，回复>对。首先，通过使用一个基于回复和帖子标题之间的各种关系(如结构和内容特征)的 SVM 分类器，将所有和帖子标题在逻辑上相关的回复抽取出来。接着，再根据回复的内容质量等特征对这些抽取出来的<帖子标题，回复>对进行排序(采用的排序模型为 ranking SVM)。最后，将排在前 N 的<帖子标题，回复>对选用为聊天知识。通过在一个电影论坛中进行的相关实验得到的结果表明，本文提出的自动抽取聊天知识的方法能快速、有效地构建聊天知识库。

本文的主要贡献为：1. 本文第一个提出了使用在线论坛为聊天机器人自动抽取聊天知识；2. 本文设计并实现了一种瀑布式模型来从在线论坛中抽取出高质量的形式为<帖子标题，回复>对的聊天知识。瀑布式模型可以在不同的阶段优化使用不同的特征，因此保证了抽取出来的聊天知识具有很高的质量；3. 实验结果表明，在判别相关回复时，结构特征是最有效的特征，而在识别高质量回复时，作者信息是最有效的特征。

关键词：聊天机器人，机器学习，知识获取，自然语言处理

ABSTRACT

A chatbot is a conversational agent that interacts with users in a certain domain or on a certain topic with natural language sentences. Normally, a chatbot works by a user asking a question or making a comment, with the chatbot answering the question, or making a comment, or initiating a new topic. Most existing chatbots consist of dialog management modules to control the conversation process and chatbot knowledge bases to response to user input. Typical implementation of chatbot knowledge bases contains a set of templates that match user inputs and generate responses. Templates currently used in chatbots, however, are hand-coded. Therefore, the construction of chatbot knowledge bases is time consuming, and difficult to adapt to new domains.

This paper presents a novel approach for extracting high-quality <thread-title, reply> pairs as chat knowledge from online discussion forums so as to efficiently support the construction of a chatbot for a certain domain. Given a forum, the high-quality <thread-title, reply> pairs are extracted using a cascaded framework. First, the replies logically relevant to the thread title of the root message are extracted with an SVM classifier from all the replies, based on correlations such as structure and content. Then, the extracted <thread-title, reply> pairs are ranked with a ranking SVM based on their content qualities. Finally, the Top-N <thread-title, reply> pairs are selected as chatbot knowledge. Results from experiments conducted within a movie forum show the proposed approach is effective.

Contribution of this study can be summarized as follows: 1. Perhaps for the first time, this paper proposes using online discussion forums to extract chatbot knowledge. 2. A cascaded framework is designed to extract the high-quality <thread-title, reply> pairs as chabot knowledge from forums. It can optimally use different features in different passes, making the extracted chatbot knowledge of higher quality. 3. Experimental results show that structural features are the most effective features in identifying *RR* and author features are the most effective features in identifying high-quality *RR*.

Keywords: Chatbot, Machine Learning, Knowledge Acquisition, Natural Language Processing

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 重庆大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：黄阵林 签字日期： 2006 年11月26日

学位论文版权使用授权书

本学位论文作者完全了解 重庆大学 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 重庆大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

保密（ ），在____年解密后适用本授权书。

本学位论文属于

不保密（√）。

（请只在上述一个括号内打“√”）

学位论文作者签名：黄阵林 导师签名：

签字日期：2006 年11月26日 签字日期：2006 年 11 月 26日

1 Introduction

1.1 Motivation and Background

This study is based on the work of AskBill Chatbot project and derived in part from the paper by Huang et al. [1].[1]

Most existing chatbots knowledge bases implementation contains a set of templates that match user inputs and generate responses. Templates currently used in chatbots, however, are hand-coded. Therefore, the construction of chatbot knowledge bases is time consuming, and difficult to adapt to new domains. Furthermore, the coverage and update speeds of chat knowledge are narrow and slow.

In this paper, a novel approach is proposed for extracting high-quality <thread-title, reply> pairs from online discussion forums to supplement chatbot knowledge base. Given a forum, the high-quality <thread-title, reply> pairs are extracted using a cascaded framework. First, the replies logically relevant to the thread title of the root message are extracted with an SVM classifier from all the replies, based on correlations such as structure and content. Then, the extracted <thread-title, reply> pairs are ranked with a ranking SVM based on their content qualities. Finally, the Top-*N* <thread-title, reply> pairs are selected as chatbot knowledge.

With the proposed approach for extracting high-quality <thread-title, reply> pairs as chat knowledge from online discussion forums, a chatbot can be efficiently and quickly constructed for a certain domain. The results and evaluation also show that the extracted knowledge is effective to improve the coverage of topics, diversity of replies, and update speeds. Compared with manual knowledge construction methods, the approach proposed in this paper is more efficient in building a specific domain chatbot.

1.2 Chatbot

Artificial Intelligence (AI) has a long history since at least the 1950s. AI is the collection of problems and methodologies studied by artificial intelligence researchers. AI currently encompasses a huge variety of subfields, from general purpose areas such as perception and logical reasoning, to specific tasks such as playing chess, proving mathematical theorems, writing poetry, and diagnosing diseases, leading to many

[1] This work was conducted and finished when the author was visiting Microsoft Research Asia during Feb.2005 - Mar.2006 as a component of the project of AskBill Chatbot led by Dr. Ming Zhou.

different definitions on AI, but all the views of AI can fall into the following four categories: thinking humanly, thinking rationally, acting humanly, and acting rationally [2]. Rationality stands for the ideal concept of intelligence. Thinking humanly and thinking rationally describe systems that think like humans and systems that think rationally separately, they are concerned with thought processes and reasoning, while the other two address behavior. The cognitive modeling approach is a test for thinking humanly. It tries to express the actual workings of human minds as a computer program and brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind. The "laws of thought" (The law of identity, noncontradiction, and the excluded middle) approach describes the test for thinking rationally, it emphasizes correct inferences. Acting rationally means acting so as to achieve one's goals, given one's beliefs. An agent is just something that perceives and acts. In this approach, AI is viewed as the study and construction of rational agents. The issue of acting like a human comes up primarily when AI programs have to interact with people, as when an expert system explains how it came to its diagnosis, or a natural language processing system has a dialogue with a user. These programs must behave according to certain normal conventions of human interaction in order to make themselves understood. The underlying representation and reasoning in such a system may or may not be based on a human model. The ultimate test for acting humanly is the Turing Test [3].

In 1950, Alan Turing, a brilliant British mathematician, proposed a test of a machine's capability to perform human-like conversation in his paper which was known as the Turing Test. The Turing Test was designed to provide a satisfactory operational definition of intelligence. Turing defined intelligent behavior as the ability to achieve human-level performance in all cognitive tasks, sufficient to fool an interrogator. The Turing Test posed a question: "Can machines think?" It proceeds as follows: a human judge engages in a natural language conversation with two other parties, one a human and the other a machine; if the judge cannot reliably tell which is which, then the machine is said to pass the test. This operational test for intelligent behavior is also called the Imitation Game, Figure 1.1 illustrates this test. It is assumed that both the human and the machine try to appear human. In order to keep the test setting simple and universal, the conversation is usually limited to a text-only channel such as a teletype machine as Turing suggested. But there is plenty of work to do for passing the test, at least, the computer would need to possess the following capabilities: natural language

processing to enable it to communicate successfully in English (or some other human language), knowledge representation to store information provided before or during the interrogation, automated reasoning to use the stored information to answer questions and to draw new conclusions, and machine learning to adapt to new circumstances and to detect and extrapolate patterns.

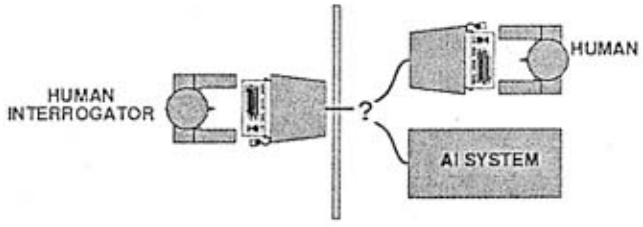

Figure 1.1 Operational Test for Intelligent Behavior: The Imitation Game

There are also some formal instantiations of a Turing Test such as the Loebner Prize[2] and ChatterBox Challenge[3], etc. The Loebner Prize for AI is The First Turing Test named after Alan Turing. It pledged a Grand Prize of $100,000 and a Gold Medal for the first computer whose responses were indistinguishable from a human's. Such a computer can be said "to think." Each year an annual prize of $2,000 and a bronze medal is awarded to the most human-like computer. The winner of the annual contest is the best entry relative to other entries that year, irrespective of how good it is in an absolute sense.

To pass the Turing Test, even the formal instantiations of a Turing Test, an agent must have command of language, wide range of knowledge, ability to reason and learn, etc., as one of these kinds of agents, a chatbot is a conversational agent that simulates and tries to maintain conversations with human users.

A chatbot, also be referred to as chatterbot, talk bot, chat bot, chatterbox, is a conversational agent that interacts with users in a certain domain or on a certain topic with natural language sentences. Normally, a chatbot works by a user asking a question or making a comment, with the chatbot answering the question, or making a comment, or initiating a new topic. Many chatbots have been deployed on the Internet for the purpose of seeking information, site guidance, FAQ answering, and so on, in a strictly limited domain. Existing famous chatbot systems include ELIZA [4], PARRY [5] and

[2] http://www.loebner.net/Prizef/loebner-prize.html
[3] http://www.chatterboxchallenge.com/

ALICE.[4] Most existing chatbots consist of dialog management modules to control the conversation process and chatbot knowledge bases to response to user input. Typical implementation of chatbot knowledge bases contains a set of templates that match user inputs and generate responses. Templates currently used in chatbots, however, are hand-coded. There are several kinds of templates employed by current chatbot systems such as transformation rule scripts, pattern matching based <stimulate, response> pairs stored as chat knowledge, etc.

ELIZA is the most famous one among chatbots using transformation rule scripts. It was originally designed to emulate a psychotherapist. ELIZA does not have a large knowledge database except for some syntax scripts, so it can not understand the meaning of user inputs, and just attempts to imitate a paranoid schizophrenic. But many users believed the machine understood them, even some psychologists thought of using ELIZA as an assistant. The main idea of ELIZA is to decompose sentences according to transformation rules by looking at specific keywords and sentence patterns such as "If the sentence has the keyword XXX in the sentence pattern YYY then say ZZZ", and generate responses based on reassembly rules associated with the decomposition rules. ELIZA also stores specific topics mentioned by the user, it makes people think that ELIZA remembers the conversation by this way. If everything else fails, ELIZA uses some "catch-all" rules to output a saved sentence (such as "I'm not sure I understand you fully.", "Please go on.", etc.) which may have no sense to current conversation. Figure 1.2 shows a transcript of the conversation between a young woman and ELIZA.

As containing chat knowledge which consists of pattern matching based <stimulate, response> pairs, ALICE is a much smarter chatbot. ALICE is the Artificial Linguistic Internet Computer Entity, which was implemented by Wallace in 1995. The knowledge of ALICE about conversation patterns is stored in Artificial Intelligence Mark-up Language[5] (AIML) files. AIML is just a derivative of Extensible Mark-up Language (XML). AIML enables people to input knowledge into chatbots.

[4] http://www.alicebot.org/
[5] http://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html

Young woman: Men are all alike.
ELIZA: In what way?
Young woman: They're always bugging us about something specific or other.
ELIZA: Can you think of a specific example?
Young woman: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here?
Young woman: He says I'm depressed much of the time.
ELIZA: I am sorry to hear you are depressed.
Young woman: It's true. I'm unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?
Young woman: I need some help, that much seems certain.
ELIZA: What would it mean to you if you got some help?
Young woman: Perhaps I could learn to get along with my mother.
ELIZA: Tell me more about your family.
Young woman: My mother takes care of me.
ELIZA: Who else in your family takes care of you?
Young woman: My father.
ELIZA: Your father.
Young woman: You are like my father in some ways.
ELIZA: What resemblance do you see?
Young woman: You are not very aggressive but I think you don't want me to notice that.
ELIZA: What makes you think I am not very aggressive?
Young woman: You don't argue with me.
ELIZA: Why do you think I don't argue with you?
Young woman: You are afraid of me.

Figure 1.2 A Transcript of the Conversation between a Chatter and ELIZA

AIML describes a class of data objects called AIML objects and partially describes the behavior of computer programs that process them. AIML objects are made up of units called topics and categories. The topic is an optional top-level element, has a name attribute and a set of categories related to that topic. Categories are the basic unit of knowledge in AIML; each category consists of an input question, an output answer, and an optional context. The question, or stimulus, is called the pattern. The answer, or response, is called the template. The two types of optional context are called "that" and "topic". The <that> tag appears inside the category, and its pattern matches the robot's

last utterance. The <topic> tag appears outside the category, and collects a group of categories together. So each category is a rule for matching an input and converting to an output, a pattern is used to match against the user input, and a template is used in generating the chatbot answer.

The AIML pattern language is simple, consisting only of words, spaces, and the wildcard symbols _ and *. The words may consist of letters and numerals, but no other characters. Words are separated by a single space, and the wildcard characters function like words. The pattern language is case invariant.

There are three types of categories: atomic categories, default categories, and recursive categories.

1) Atomic categories: are those with patterns that do not have wildcard symbols, _ and *, e.g.:

```
<category>
  <pattern>10 DOLLARS</pattern>
  <template>Wow, that is cheap. </template>
</category>
```

In the above category, if the user inputs "10 dollars", then ALICE answers "WOW, that is cheap".

2) Default categories: are those with patterns having wildcard symbols * or _. The wildcard symbols match any input but they differ in their alphabetical order. Assuming the previous input 10 Dollars, if the robot does not find the previous category with an atomic pattern, then it will try to find a category with a default pattern such as:

```
<category>
  <pattern>10 *</pattern>
  <template>It is ten.</template>
</category>
```

So ALICE answers "It is ten".

3) Recursive categories: are those with templates having <srai> and <sr> tags, which refer to simply recursive artificial intelligence and symbolic reduction. Recursive categories have many applications: symbolic reduction that reduces complex grammatical forms to simpler ones; divide and conquer that splits an input into two or more subparts, and combines the responses to each; and dealing with synonyms by mapping different ways of saying the same thing to the same reply. For example:

```
<category>
  <pattern>YES*</pattern>
  <template> <srai>YES</srai><sr/> <template>
</category>
```

The input is partitioned into two parts, "yes" and the second part; * is matched with the <sr/> tag. Here is another example:

```
<category>
  <pattern>DO YOU KNOW WHO * IS</pattern>
  <template><srai>WHO IS <star/></srai></template>
</category>
```

Whatever input matched this pattern, the portion bound to the wildcard * may be inserted into the reply with the markup <star/>. This category reduces any input of the form "Do you know who X is?" to "Who is X?"

Before ALICE could use the categories in AIML files as its chat knowledge, they must be loaded into a knowledge tree. Only if categories are loaded into a tree, the pattern matching algorithm can perform searching for proper answers for user inputs.

Given the following categories:

```
<topic name="population">
  <category>
    <pattern> WHAT IS THE POPULATION OF AMERICA</pattern>
    <template>More than 250 million.</template>
  </category>
  <category>
    <pattern> WHAT IS THE POPULATION OF *</pattern>
    <template>The population of <star/> is larger than 100,000.</template>
  </category>
<topic>
<category>
  <pattern>WHAT IS THE BIRTHDAY OF BILL GATES</pattern>
  <template>October 28, 1955.</template>
</category>
<category>
  <pattern>HELLO *</pattern>
  <template>Hi there! <srai><star/></srai></template>
</category>
```

These four categories will be loaded into a knowledge tree like Figure 1.3.

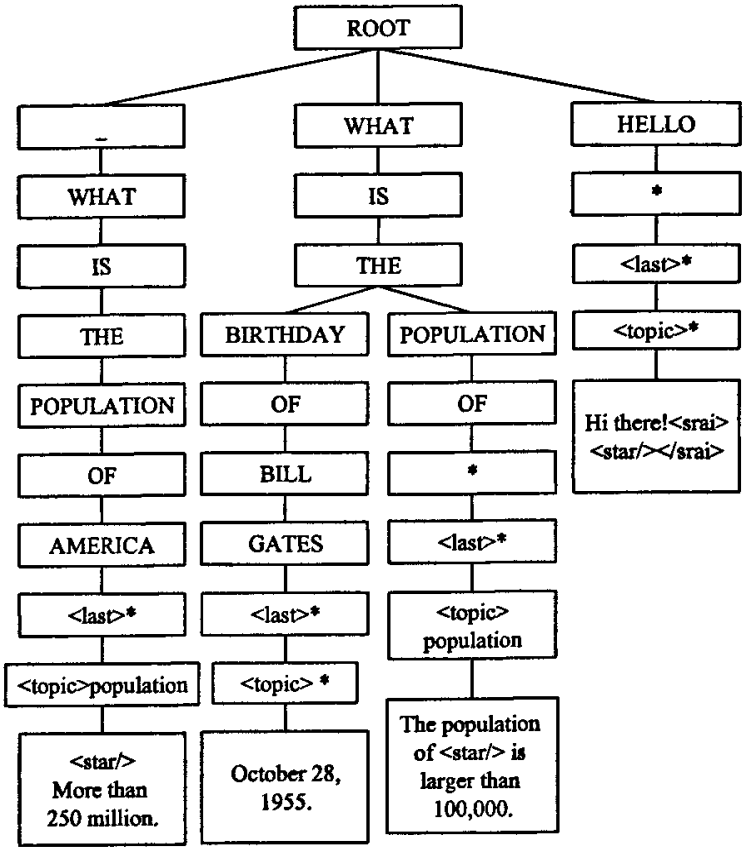

Figure 1.3 A Sample Knowledge Tree

When a user inputs a sentence, an input normalization process including substitution, sentence-splitting, and pattern-fitting normalizations is performed. After user input is normalized, input pattern is generated, then the pattern matching algorithm combines the input pattern, the <that> pattern, and the <topic> pattern into a single "path" or sentence named "input path" into this form: "PATTERN <that> THAT <topic> TOPIC" and treats the tokens <that> and <topic> like ordinary words.

When the input path is generated, the pattern matching is performed to search the proper template for this input path. The search algorithm is described as follows:

Given an input path starting with word X, and a node of the knowledge tree:

1) Does the node contain the key _? If so, search the sub tree rooted at the child node linked by _. Try all remaining suffixes of the input path following X to see if one matches. If no match was found, try:

2) Does the node contain the key X? If so, search the sub tree rooted at the child node linked by X, using the tail of the input (the suffix of the input path with X removed). If no match was found, try:
3) Does the node contain the key *? If so, search the sub tree rooted at the child node linked by *. Try all remaining suffixes of the input path following X to see if one matches. If no match was found, go back up to the parent of this node and put X back on the head of the input path.

For completeness there should also be a terminal case: If the input is null (no more words) and the node contains the <template> key, then a match was found. Halt the search and return the matching node. If the root node contains a key "*" and it points to a leaf node, then the algorithm is guaranteed to find a match.

Note that the matching is word-by-word, not category-by-category, and in the pattern matching algorithm, the "_" has first priority, an atomic word match second priority, and a "*" match lowest priority at every node. The matching algorithm is a highly restricted version of depth-first search, also known as backtracking. Figure 1.4 illustrates this algorithm.

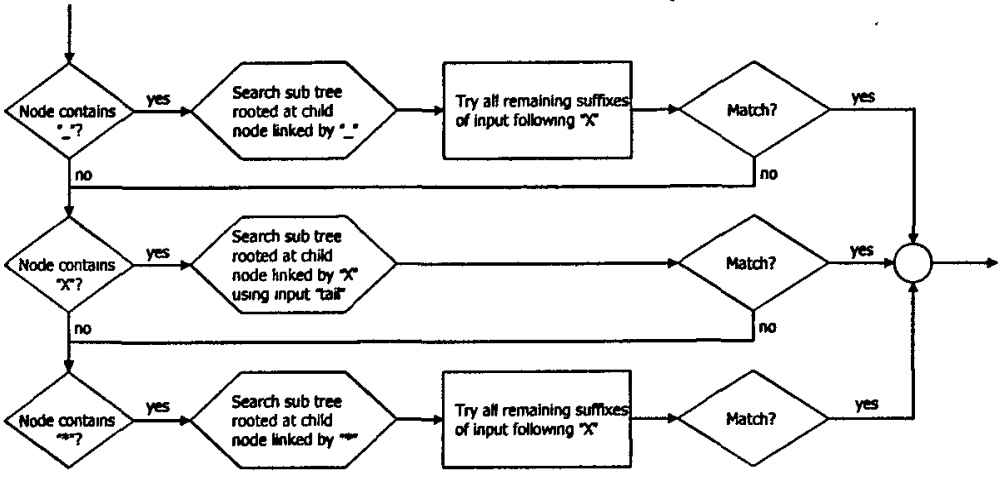

Figure 1.4 An Illustration of the Pattern Matching Algorithm

So, when a user inputs "Hello, what is the population of China", an input path "HELLO WHAT IS THE POPULATION OF CHINA<last>*<topic>*" is generated accordingly, and output chatbot response "Hi there! The population of China is larger than 100,000." But if a user inputs "Hello, what is your favorite movie?" the chatbot could not give any responses with the knowledge tree consists of only four categories.

There are more than 50,000 categories in the current public domain ALICE "brain", it takes people over several years to add them. This shows that knowledge is a key issue to chatbots that use <stimulate, response> pairs as chatbot knowledge source.

Based on above analysis, it shows that whatever kinds of templates employed by current chatbots such as ELIZA and ALICE, etc., all their knowledge are manually hand-coded, which is time consuming, and restricts adaptation to new discourse domains and new languages. In this paper, an approach is proposed to extract chat knowledge from online discussion forums, which can support to construct a chatbot for a certain domain efficiently and quickly.

1.3 AskBill Chatbot

AskBill Chatbot[6] is a powerful chatbot system researched and developed by NLC group of Microsoft Research Asia. AskBill is a system demonstrates natural human-machine conversations. It is supported by automatic chat knowledge extraction, Question Answering (QA) and internet resources.

AskBill uses a template-based chat knowledge representation similar to ALICE but with more extended features. The chat knowledge base of AskBill consists of two parts: manually hand-coded daily dialogue and commonsense knowledge such as greetings, appreciations, etc. and automatically extracted domain knowledge such as movie, sport, etc. In addition to these two components, a web based QA engine was built to enable the chatbot to answer factoid and definition questions by searching the web and extracting the answers. A sentence-level paraphrasing module is also built to support different ways of conveying the same information, so that different sentences which mean the same thing and accordingly, could share the same answer. Figure 1.5 illustrates the architecture of AskBill.

[6] The author took part in the project of AskBill Chatbot led by Dr. Ming Zhou during the visiting to Microsoft Research Asia.

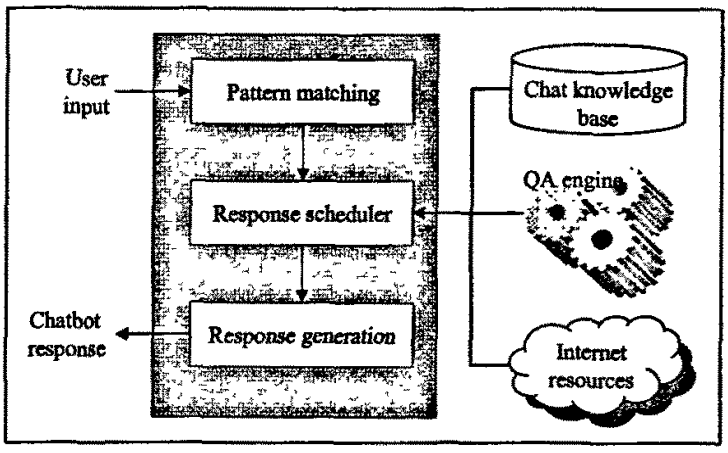

Figure 1.5 AskBill Structure

Figure 1.6, Figure 1.7, Figure 1.8, and Figure 1.9 show four typical scenarios of conversation between AskBill Chatbot and human. Figure 1.6 shows a common conversation with human using knowledge that comes from the daily dialogue and commonsense knowledge base encoded manually. Figure 1.7 shows the scenario of answering factoid questions by AskBill Chatbot supported by a QA engine. Figure 1.8 shows the scenario of performing information seeking which can help people to find internet resources conveniently and quickly. Figure 1.9 shows some chat transcripts in the movie domain, note that the knowledge used by the chatbot is extracted automatically from an online discussion forum. To extract domain knowledge automatically from online discussion forums is the main focus of this paper.

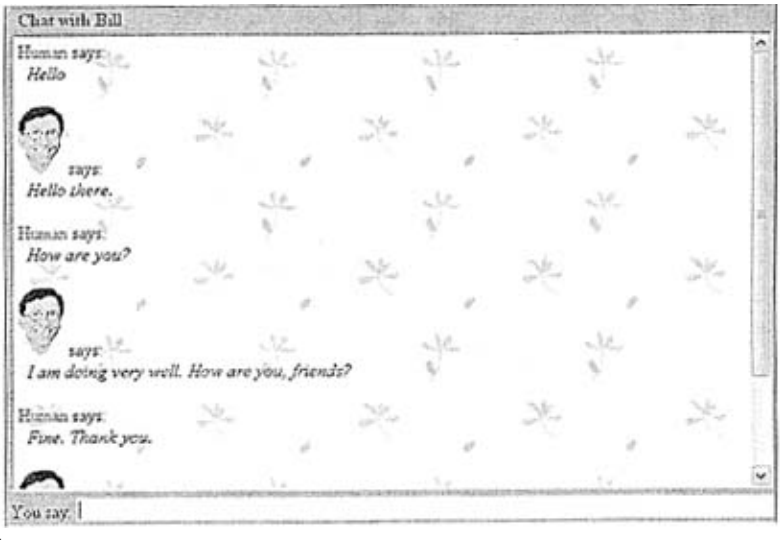

Figure 1.6 Common Conversations

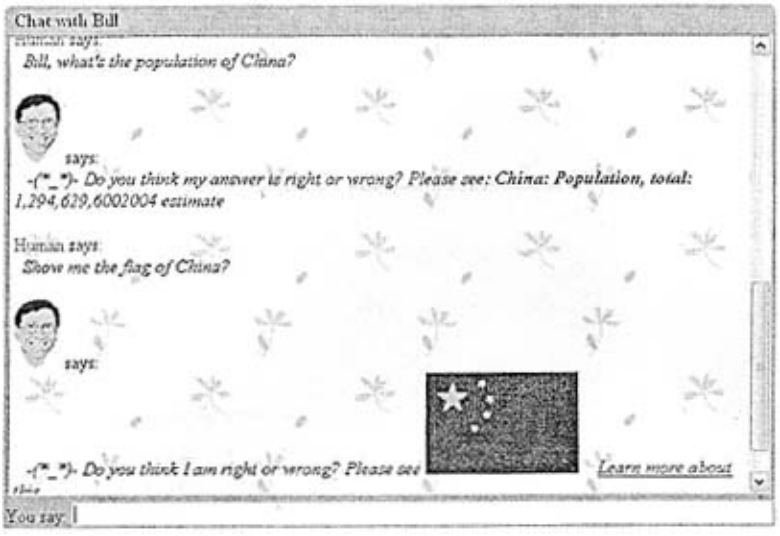

Figure 1.7 Factoid Questions

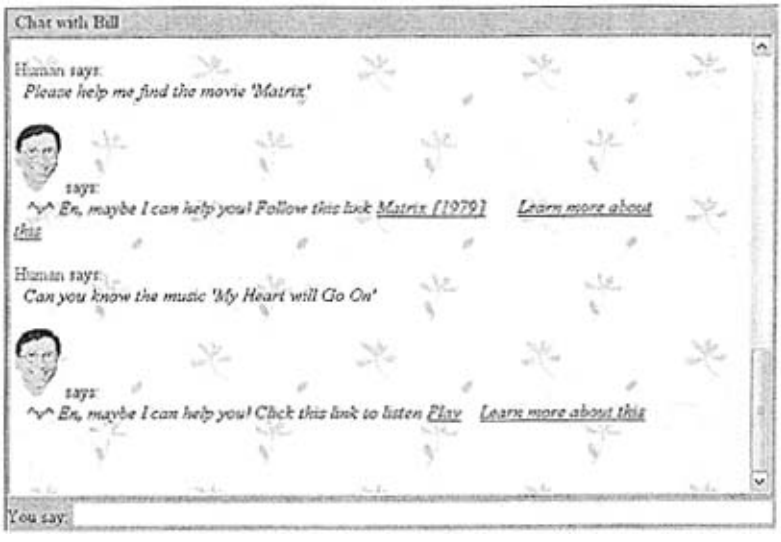

Figure 1.8 Information Seeking

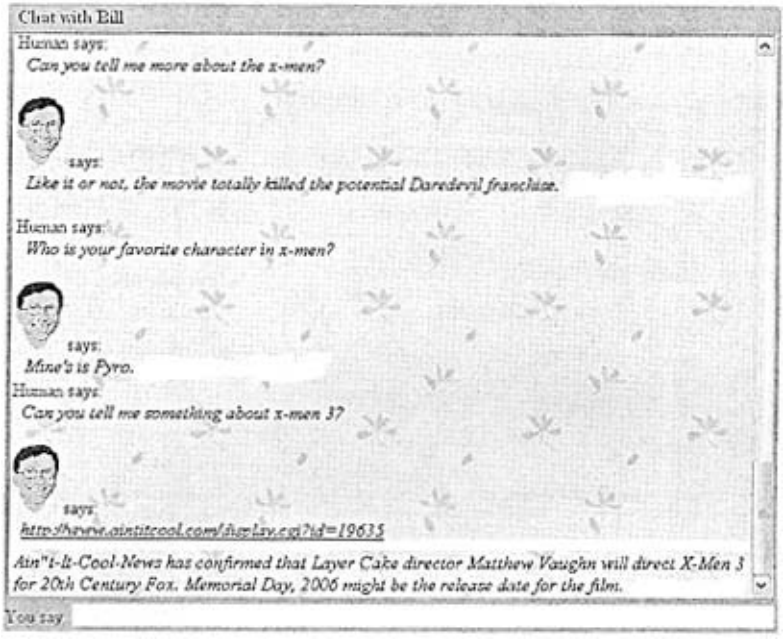

Figure 1.9 Conversations in Movie Domain

Figure 1.10 gives the overview of AskBill Chatbot. AskBill's knowledge is stored in xml files written by ABML (AskBill Markup Language) format which is similar to AIML used by ALICE but with more extended features. Figure 1.11 shows a sample ABML file that can perform real-time information seeking. Figure 1.12 shows a sample ABML file that can handle factoid questions by utilizing a QA engine.

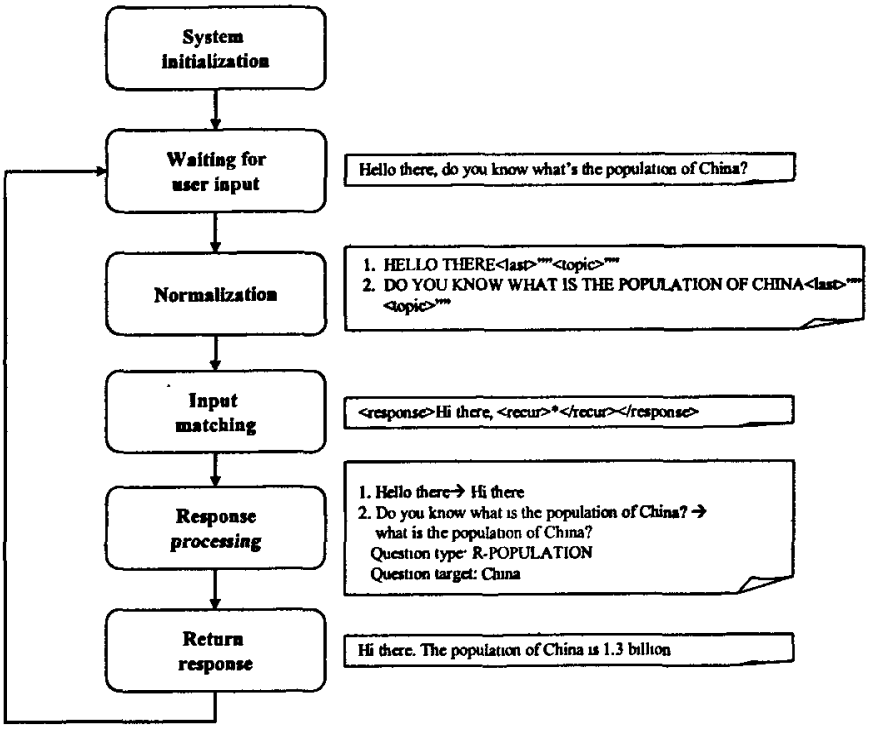

Figure 1.10 Overview of AskBill

```
<?xml version="1.0" encoding="utf-8" ?>
<abml>
    <category>
        <input>WHAT IS THE WEATHRE LIKE TODAY IN *</input>
        <response>The weather of <star/> now is <search name="weather"/></response>
    </category>
</abml>
```

Figure 1.11 A Sample ABML File for Using Internet Resources

```
<?xml version="1.0" encoding="utf-8" ?>
<abml>
    <category>
        <input>WHAT IS THE POPULATION OF * AND *</input>
        <response>
            <invisible>
                <question value= "WHAT IS THE POPULATION OF &lt;star
index=&quot;1&quot;/> AND &lt;star index=&quot;2&quot;/>"/>
                <qtype value="PEOPLE\POPULATION" />
                <qtarget index="1" value="&lt;star index=&quot;1&quot; />" />
                <qtarget index="2" value="&lt;star index=&quot;2&quot; />" />
            </invisible>
            The Population of <star index="1" /> and <star index="2" /> is <answer/>.
        </response>
    </category>
</abml>
```

Figure 1.12 A Sample ABML File for QA

There are three types of categories in ABML: atomic categories, default categories, and recursive categories which are similar to AIML, Figure 1.13 shows the three types of categories.

1. Atomic Category

```
<category>
  <input>HELLO</input>
  <response>Hi there!</response>
</category>
```

2. Default Category
(pattern contains wildcard _ , *, %)

```
<category>
  <input>10 *</input>
  <response>It is ten.</response>
</category>
<category>
  <input>_ METERS</input>
  <response>It is to long.</response>
</category>
<category>
  <input>10 % METERS</input>
  <response>About 10 meters.</response>
</category>
```

3. Recursive Category (response contains recursive tag <recur>)

```
<category>
  <input>HELLO *</input>
  <response>Hi there, <recur>*</recur></response>
</category>
<category>
  <input>WHAT IS YOUR FAVORITE MOVIE</input>
  <response>It is The Lord of the Rings.</response>
</category>
```

Figure 1.13 Three Types of Categories in ABML

Before pattern matching algorithm can perform searching for proper answers for user inputs, the ABML files must be loaded into a knowledge tree, this is similar to ALICE, but the pattern matching algorithm is a little different since a wild card % is defined in ABML specification to match zero or one word.

Given an input path starting with word X, and a node of the knowledge tree:

1) Does the node contain the key _? If so, search the sub tree rooted at the child node linked by _. Try all remaining suffixes of the input path following X to see if one matches. If no match was found, try:

2) Does the node contain the key X? If so, search the sub tree rooted at the child node linked by X, using the tail of the input (the suffix of the input path with X removed). If no match was found, try:
3) Does the node contain the key %? If so, search the sub tree rooted at the child node linked by %, using the raw input starting with X first, if no match was found, then trying to use the tail of the input (the suffix of the input path with X removed). If no match was found, try:
4) Does the node contain the key *? If so, search the sub tree rooted at the child node linked by *. Try all remaining suffixes of the input path following X to see if one matches. If no match was found, go back up to the parent of this node and put X back on the head of the input path.

In the pattern matching algorithm, the "_" has first priority, an atomic word match second priority, a "%" match third priority, and a "*" match lowest priority at every node. If the input is null (no more words) and the node contains the <response> key, then a match was found. Halt the search and return the matching node. Figure 1.14 shows the pattern matching algorithm of AskBill.

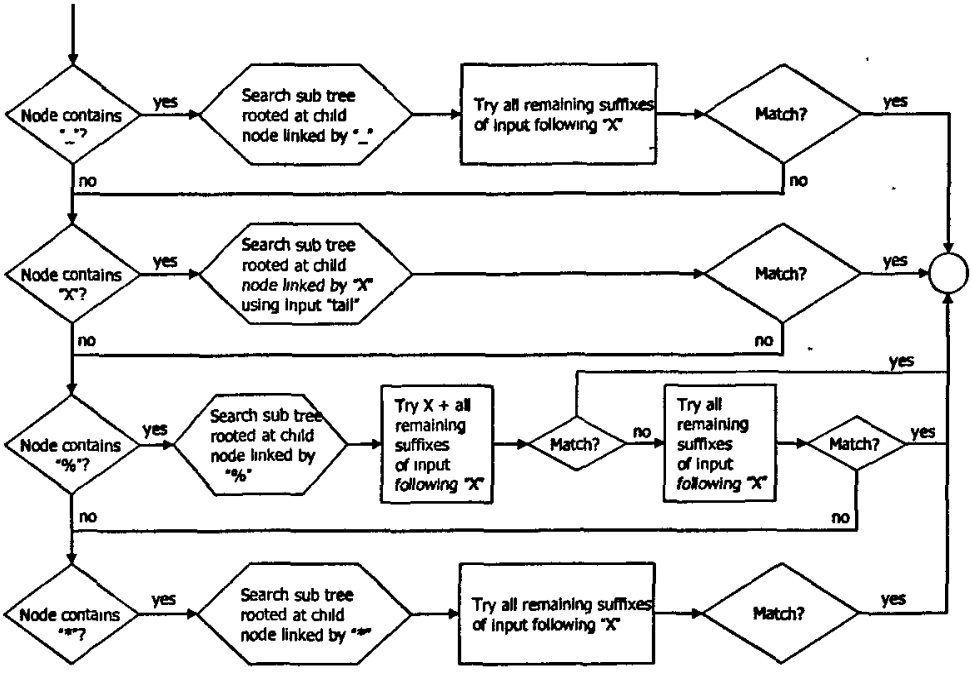

Figure 1.14 Pattern Matching Algorithm of AskBill

Besides encoded QA templates in ABML, another QA engine is built enabling AskBill to answer factoid and definition questions. The response scheduler component of AskBill is responsible for detecting types of users' inputs, if QA type questions are detected, then the questions are sent to the QA engine to acquire the corresponding answers. Both methods can enhance the performance of AskBill, making AskBill be unique as a QA agent, e.g., when a user input "When was Bill Gates born?", AskBill could give the right answer "Bill Gates was born on October 28, 1955.". More details about a definitional QA engine as a component of the project of AskBill is described in [6].

Different sentences which mean the same thing may be used by different people when chatting with AskBill, and accordingly should share the same answer, e.g., "When was Bill Gates born?" and "What is the birthday of Bill Gates?" should share the same answer "October 28, 1955.". To support different ways of conveying the same information, a sentence-level paraphrasing module is built for AskBill. To get paraphrases, a baseline method is employed to search the web using pattern seeds (each pattern seed consists of interrogatives and focus word, e.g., "when earthquake"), and then extract sentences (such as "When did the Los Angeles earthquake happen?") that contain pattern seeds from the search results, finally, extract similar sentences, rank them and filter the wrong ones. A more efficient way to generate paraphrase templates is well studied in [7] which is also a component of the project of AskBill.

1.4 Organization of This Paper

The rest of this paper is organized as follows. Chapter 2 analyzes the problem and outlines the characteristics of online discussion forums with the explanations of the challenges of extracting stable <thread-title, reply> pairs. Important related research topics such as text categorization and learning to rank (ranking) are introduced, too. Chapter 3 presents the proposed cascaded framework for extracting chatbot knowledge from online discussion forms. Experimental results and an end to end evaluation on the chatbot knowledge extraction approach are reported in Chapter 4. Chapter 5 presents comparison of the approach with other related work. The conclusion and the future work are provided in Chapter 6.

2 Chat Knowledge Acquisition

2.1 Chat Knowledge Extraction from Online Discussion Forums

Most existing chatbots knowledge bases implementation contains a set of templates, and templates currently used in chatbots, however, are hand-coded. Therefore, the construction of chatbot knowledge bases is time consuming, and difficult to adapt to new domains and new languages. Furthermore, the coverage and update speeds of chat knowledge are narrow and slow. For example, if the chat knowledge of a chatbot is encoded for the sports domain using English, when movie domain or Chinese dialogue to the chatbot is required, the work that experts manually encode the knowledge into chatbot has to be repeated anew, even more, the topic coverage of chatbot knowledge can not be guaranteed since it is an impossible task for experts to encode a complete knowledge base manually. To solve the problems of chatbots knowledge construction, an approach of extracting chatbot knowledge from online resources such as online discussion forums is studied and proposed in this paper.

Furthermore, current chatbot systems can not deal with real time questions such as "What is the price of Microsoft's stock?" or "What is the weather like in Beijing?" because the answers of these questions are changed dynamically. Another problem raises when factoid or definitional questions are asked, e.g., a chatbot can not give any responses if user asks "When was Bill Gates born?" One may suggest encoding this kind of questions into chatbot knowledge base, but to enumerate all the possible questions is impossible. Taking "When was X born?" as an example again, since there are so many people in the world, it is hardly to replace "X" by enumerating. Since it is impossible to encode answers of these kinds of questions as fixed knowledge into the chatbot knowledge base, methods such as extracting chatbot knowledge at runtime must be explored. To solve the real time questions problem, online resources could be used by searching the web and extracting the right answers to the questions. To solve the problem arising from factoid and definitional questions, a web based QA engine can be integrated into a chatbot to efficiently support the answering factoid and definition questions.

In this paper, the ways of generating answers to real time questions, factoid and definitional questions are not the focus. Instead, this paper focuses on extracting domain knowledge automatically from online discussion forums.

An online discussion forum is a web community that allows people to discuss common topics, exchange ideas, and share information in a certain domain, such as sports, movies, and so on. Creating threads and posting replies are major user behaviors in forum discussions. Large repositories of archived threads and reply records in online discussion forums contain a great deal of human knowledge on many topics. In addition to rich information, the reply styles from authors are diverse. The high-quality replies of a thread, if mined, could be of great value to the construction of a chatbot for certain domains.

By "chatbot knowledge extraction" throughout this paper, it means extracting the pairs of <input, response> from online resources.

Chat knowledge extraction from online discussion forums has two related research areas: automatic chat knowledge acquisition and online communities mining for Question Answering (QA) and summarization.

Based on the study of the literature, there is no published work describing the use of online communities like forums for automatic chatbot knowledge acquisition. Existing work on automatic chatbot knowledge acquisition is mainly based on human annotated datasets, such as the work by Shawar and Atwell [8] and Tarau and Figa [9]. Shawar and Atwell develop a program to convert a machine readable text (linguistically annotated dialog corpus) into the format of the chatbot knowledge. With the annotation of the text, the extraction of chat knowledge becomes a lot easier. Tarau and Figa describe a Prolog-based conversational agent by using WordNet lexical knowledge base and existing common sense knowledge ontology from FrameNet as knowledge sources. Their approaches are helpful to construct commonsense knowledge for chatbots from well structured corpora, but are not capable of extracting knowledge for specific domains from unannotated resources such as online discussion forums.

Notably, there is some work on knowledge extraction from web online communities to support QA and summarization. Nishimura *et al.* [10] develop a knowledge base for a QA system that answers type "how" questions. Shrestha and McKeown [11] present a method to detect <question, answer> pairs in an email conversation for the task of email summarization. Zhou and Hovy [12] describe a summarization system for technical chats and emails about Linux kernel. They utilize adjacent pairs (AP) to identify initiating and responding correspondences from different

participants in one chat log. Xi *et al.* [13] describe an effective ranking function to predict the most relevant messages to queries in newsgroup search based on the thread tree structure. Kim *et al.* [14] describe a segmentation algorithm to identify how the content of a discussion board grows through generalizations and specializations of topics. Tuulos *et al.* [15] show that social network structures inferred using a few basic heuristics can be used to enhance the topic identification of a chat room when combined with a state-of-the-art topic and classification models. Agrawal *et al.* [16] use social graph analysis to partition newsgroup users into opposite camps with respect to the topic discussed, they use quotation links to infer social network between individuals. These researchers' approaches utilize the characteristics of their corpora and are best fit for their specific tasks, but they limit each of their corpora and tasks, so they cannot directly transform their methods to the chatbot knowledge extraction task.

2.2 Online Discussion Forums

Plenty of forums exist on the web focusing on different domains and topics. As a preliminary investigation, a search to the Yahoo! Directory[7] with the keyword "forum" was performed, 8,599 sites distributed in 145 categories were found are forums.

An online discussion forum is a type of online asynchronous communication systems which are dominated by a design theme that refers to as "threaded discussion" [17]. A forum normally consists of several discussion sections. Each discussion section focuses on a specific discussion theme and includes many threads. People can initiate new discussions by creating threads, or ask (answer) questions by posting questions (replies) to an existing section. In a section, threads are listed in chronological order. Within a thread, information such as thread title, thread starter, and number of replies are presented. The thread title is the title of the root message posted by the thread starter to initiate discussion. One can access a thread from the thread list and see the replies listed in chronological order, with the information of the authors and posting times. Figure 2.1 shows the rough structure of most online discussion forums.

[7] http://dir.yahoo.com/

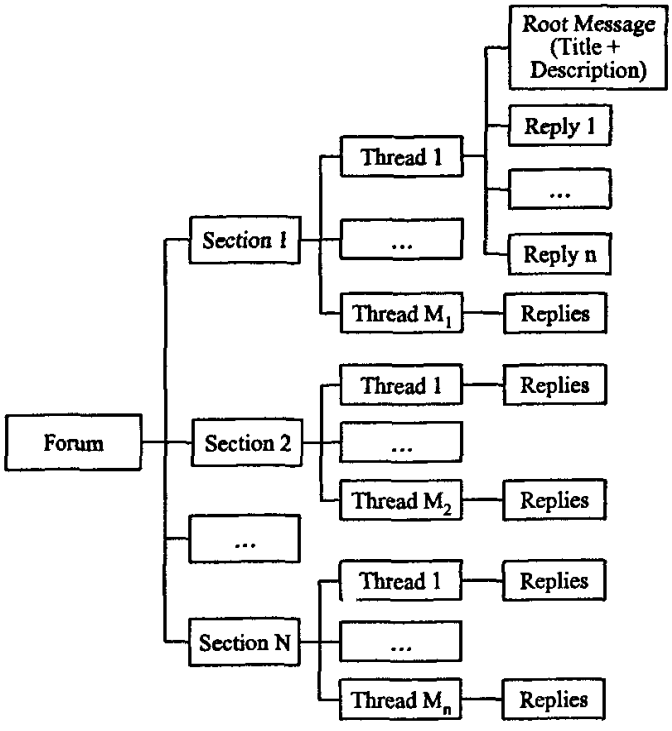

Figure 2.1 Online Discussion Forum Structure

Plenty of forums exist on the web focusing on different domains and topics; this makes forums be reasonable as sources to extract chatbot knowledge and ensures the different domains and language requirements on chat knowledge. Moreover, compared with other types of web communities such as newsgroups, online discussion forums are better suited for chatbot knowledge extraction for the following reasons:

1) In a thread within a forum, the root message and its following up replies can be viewed as <input, response> pairs, with same structure of chat template of a chatbot.
2) There is popular, rich, and live information in an online discussion forum.
3) Diverse opinions and various expressions on a topic in an online discussion forum are useful to extract diverse <input, response> pairs for chatbots.

2.3 Challenges

However, because of the nature of a forum, it is difficult to extract high-quality <input, response> pairs that meet chatbot requirements:

1) Replies are often short, elliptical, and irregular, and full of spelling, usage, and grammar mistakes which results in noisy text.

2) Not all of replies are related to root messages.
3) A reply may be separated in time or place from the reply to which it responds, leading to a fragmented conversational structure. Thus, adjacent replies might be semantically unrelated.
4) There is no evidence to reveal who has replied to which reply unless the participants have quoted the entire entries or parts of a previously posted reply to preserve context [18, 19].

2.4 Strategy and Approach Employed

Due to technical limitations of current chatbots in handling dialogue management, this paper assumes that pairs of <input, response> for a chatbot should be context independent, which means that the understanding inputs and responses will not rely on the previous <input, response>.

The chatbot knowledge extraction task from a forum can be modeled as extracting pairs of <thread-title, high-quality replies> of each thread. In other words, a thread title is used to model the user input of a chatbot and high-quality replies of this thread are used to model the chatbot responses. But it is not an easy task to extract chatbot knowledge from online discussion forums because of the nature of a forum. To overcome sorts of difficulties mentioned in Section 2.3, lexical and structural information from different replies within threads are analyzed in the experiments, as well as user behaviors in discussions.

Therefore, to extract valid pairs of <input, response> from a forum, it first needs to extract relevant replies to initial root messages. In this process, replies that are relevant to the previous replies rather than to the initial root message are ignored and the replies logically directly relevant to the thread title are extracted. The replies to the initial root message, in spite of being relevant, may have different qualities. To select high-quality replies, a ranking SVM is employed to rank the replies. Finally, the pairs of the title of the root message and the extracted Top-*N* replies are used as the chatbot knowledge.

2.5 Related Work

2.5.1 Text Categorization

Text categorization (also known as text classification or topic spotting) is a process of classifying documents with regard to a group of one or more existent categories according to themes or concepts present in their contents. The automated categorization

(or classification) of texts into topical categories has a long history, dating back at least to the early '60s. Until the late '80s, the most effective approach to the problem seemed to be that of manually building automatic classifiers by means of knowledge-engineering techniques, i.e. manually defining a set of rules encoding expert knowledge on how to classify documents under a given set of categories. In the '90s, with the booming production and availability of online documents, automated text categorization has witnessed an increased and renewed interest, prompted by which the machine learning paradigm to automatic classifier construction has emerged and definitely superseded the knowledge-engineering approach.

Now, methods that utilize machine learning techniques are commonly used to transform text categorization into a viable task to automate text classification, allowing it to be carried out fast, into concise manner and in broad range. A number of statistical classification and machine learning techniques has been applied to text categorization, including regression models [20], nearest neighbour classifiers [20], decision trees [21], Bayesian classifiers [21], Support Vector Machines (SVM) [22], rule learning algorithms [23], relevance feedback [24], voted classification [25], and neural networks [26]. Within the machine learning paradigm, a general inductive process (called the learner) automatically builds a classifier (also called the rule, or the hypothesis) by "learning", from a set of previously classified documents, the characteristics of one or more categories. The advantages of this approach are a very good effectiveness, a considerable savings in terms of expert manpower, and domain independence [27].

In text categorization, the categories are just symbolic labels, no additional knowledge (either of a procedural or of a declarative nature) of their "meaning" is assumed available to help in the process of building the classifier. In particular, this means that the "text" constituting the category label (e.g. Sports in a news categorization task) is not to be used. Another thing is that the attribution of documents to categories should, in general, be realized on the basis of the *semantics* of the documents, and not on the basis of *metadata* (e.g. publication date, document type, publication source, etc.). That is, the categorization of a document should be based solely on *endogenous* knowledge (i.e. knowledge that can be extracted from the document itself) rather than on *exogenous* knowledge (i.e. data that might be provided for this purpose by an external source) [27].

Most of the research in text categorization has been devoted to binary problems, where a document is classified as either *relevant* or *not relevant* with respect to a

predefined topic. However, there are many sources of textual data, such as Internet News, electronic mail and digital libraries, which are composed of different topics and which therefore pose a multi-class categorization problem. In multi-class problems, it is often the case that documents are relevant to more than one topic. For example, a news article may well be relevant to several topics. The common approach for multi-class, multi-label text categorization is to break the task into disjoint binary categorization problems, one for each class. To classify a new document, one needs to apply all the binary classifiers and combine their predictions into a single decision. The end result is a ranking of possible topics. The main drawback with this approach is that it ignores any correlation between the different classes [28].

Text categorization has been used in a number of different applications, such as *document organization (e.g., filing of newspaper articles under appropriate sections)*, text filtering (e.g., junk mail identification), word sense disambiguation (refers to the activity of finding, given the occurrence in a text of an ambiguous (i.e. polysemous or homonymous) word, the sense this particular word occurrence has), hierarchical categorization of web pages (e.g., automatic categorization of web pages or sites into Yahoo!-like hierarchical catalogues [29]) etc.

Manual construction through knowledge-engineering techniques is the main approach to classify documents automatically in the '80s. This approach requires manually building an expert system capable of taking categorization decisions. The typical example of this approach is the CONSTRUE system [30], other examples of this "manual" approach to the construction of text classifiers are [31, 32].

Since the early '90s, machine learning approach has gained prominence and becomes the dominant one to the construction of automatic document classifiers. In this approach, a domain expert classified and annotated a set of documents manually as training instances beforehand, and then extracting characteristics of each document which could "prove" a document to be classified as category c_i, these characteristics are also called features. After this process, a general inductive process (also called the learner) automatically builds a classifier for a category c_i based on the extracted features of previous annotated documents. Once a classifier is built, it could classify a new document into category c_i or other categories. Machine learning based classification approach is a supervised learning method, since the learning process is driven, or supervised, by the knowledge of the categories to which the training instances belong. Figure 2.2 shows the structure of most text categorization methods.

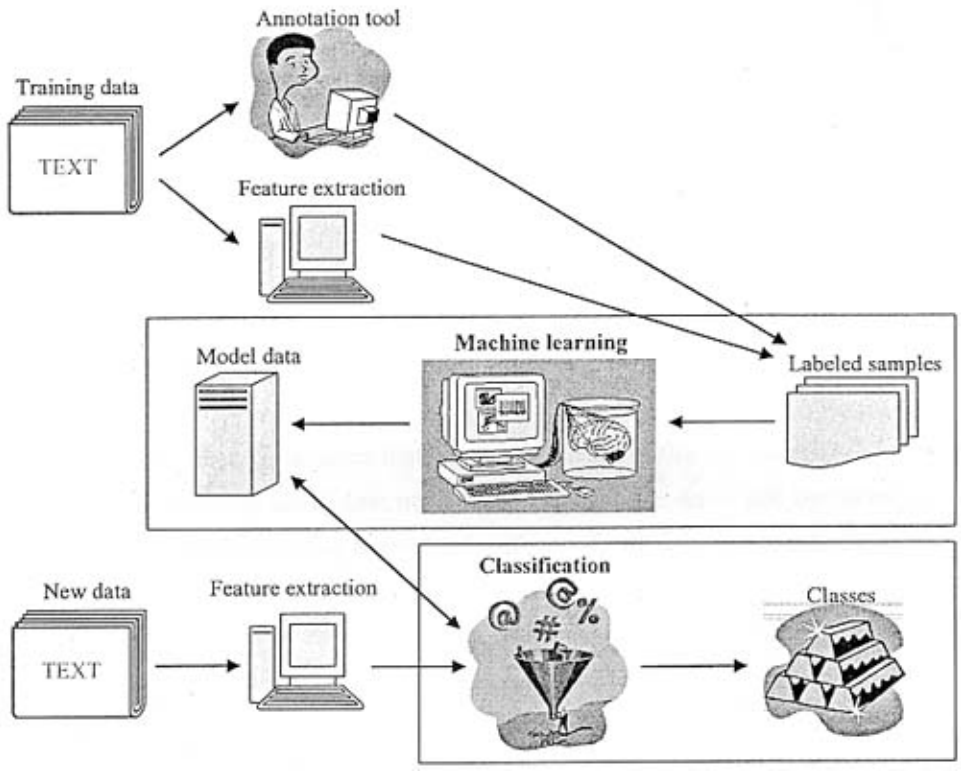

Figure 2.2 Structure of Typical Text Categorization Methods

The advantages of machine learning approach over the manual one are evident: researchers only need to focus on selecting features and the proper method to build a classifier automatically, not writing classification rules manually. So when the original set of categories is updated, or the system is ported to a completely different domain, all that is needed to do is to train a new classifier from a different set of manually classified documents by machine learning methods, neither the domain expert nor the knowledge engineer is needed to write any rules. Furthermore, the performance of the machine learning methods achieves impressive levels, making it be widely studied and applied.

2.5.1.1. Training Set and Testing Set

In general, the machine learning approach relies on an pre-annotated corpus $C_a = \{d_1,...,d_s\}$ of documents under the same set of categories $C = \{c_1,...,c_m\}$. This means that the corpus comes with a *correct decision matrix* as be shown in Table 2.1.

Table 2.1 Correct Decision Matrix of Training and Testing Set [27]

	Training set				Testing set			
	d_1	...	...	d_n	d_{n+1}	...	...	d_s
c_1	ca_{11}	...	...	ca_{1n}	$ca_{1(n+1)}$	...	...	ca_{1s}
...	...	...	...	...	...	...	...	...
c_i	ca_{i1}	...	...	ca_{in}	$ca_{i(n+1)}$	...	...	ca_{is}
...	...	...	...	...	...	...	...	...
c_m	ca_{m1}	...	...	ca_{mn}	$ca_{m(n+1)}$	...	...	ca_{ms}

A value of 1 for ca_{ij} indicates that the expert annotates file d_j to category c_i, while a value of 0 indicates that file d_j does not belong to category c_i annotated by the expert. A document d_j is called a *positive example* of c_i if ca_{ij} =1, otherwise, it is called a *negative example* of c_i.

Normally, to evaluate the performance of constructed classifier, the pre-annotated corpus C_a is typically divided into two sets, named *training set* $C_{training} = \{d_1,...,d_n\}$ and *testing set* $C_{testing} = \{d_{n+1},...,d_s\}$ separately. The training set is used to train a classifier and the testing set is used to test the performance of the trained classifier. In the training process, only the documents in the training set can be used, documents in the testing set are not allowed to participate in the training process, if this condition were not satisfied, the experimental results obtained would probably be unrealistically good [33]. In the testing process, each document in the testing set will be fed to the classifier, and then a category given by the classifier will be compared to the category annotated by the expert, finally match results of all the documents in the testing set will tell the performance of the trained classifier. Note that the training and testing set are not necessarily of equal size. This approach is called the *train-and-test* approach. An alternative approach is the *n-fold cross-validation* approach [33], whereby n different classifiers are trained by partitioning corpus C_a into n disjoint sets $C_{training\text{-}1},..., C_{training\text{-}n}$, and then iteratively applying the train-and-test approach on pairs of training and testing set $< C_{training\text{-}i}, C_a - C_{training\text{-}i}>$. The trained n classifiers are different from each other because they have been generated from n different training sets, are then processed in some way to yield the final classifier. This approach is usually adopted when the corpus C_a is small and the training process cannot afford to lose the information present in the testing set. But it is hardly the case because corpus C_a is usually large in most text

categorization applications [27].

In the train-and-test approach, it is often the case that in order to optimize a classifier, its internal parameters should be tuned by testing which values of the parameters yield the best performance. In order to make this optimization possible while at the same time safeguarding scientific standards, the set $\{d_1,...,d_n\}$ maybe further split into a "true" training set $T_r = \{d_1,...,d_j\}$, from which the classifier is trained, and a *validation set* $V_a = \{d_{j+1},...,d_n\}$ (also called a *hold-out set*), on which the repeated tests of the trained classifier aimed at parameter optimization are performed.

2.5.1.2. SVM

Automated text categorization is a supervised learning task, defined as assigning category labels (predefined) to new documents based on the likelihood suggested by a training set of labeled documents. An increasing number of learning approaches have been studied and applied, such as Decision tree (Dtree), Naïve Bayes (NB), k-Nearest Neighbor (kNN), and Support Vector Machines (SVM) etc., the SVM classifier will be shortly reviewed here.

SVM is a relatively new learning approach to text categorization. It is based on the Structural Risk Minimization principle for which error-bound analysis has been theoretically motivated [34, 35]. The goal of SVM is to produce a model which predicts category of data instances in the testing set which are given only the values of features.

Given a training set of example-label pairs (x_i, y_i), $i = 1, \ldots, l$ where $x_i \in R^n$ and $y_i \in \{1,-1\}$, the SVM requires the solution of the following optimization problem [35]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C\sum_{i=1}^{l} \xi_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0 \qquad \text{Equation 2.1}$$

The training vectors x_i are solely used in inner products which can be replaced by a kernel function $K(x_i,x_j)$ that obeys Mercer's condition: $K(x_i,x_j) \equiv \phi(x_i)^T \phi(x_j)$. This replacement can be interpreted as mapping the data vectors into a high dimensional feature space before using a hyperplane classification there. Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. There are some existing well

known kernel functions, including:

1) Linear: $K(x_i, x_j) = x_i^T x_j$.
2) Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.
3) Radial basis function (RBF): $K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2), \gamma > 0$.
4) Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$.

For simplicity, a case in a two-dimensional space with linearly separable data points is shown here, but the idea can be generalized to a high dimensional space and to data points that are not linearly separable [34]. A decision surface in a linearly separable space is a hyperplane.

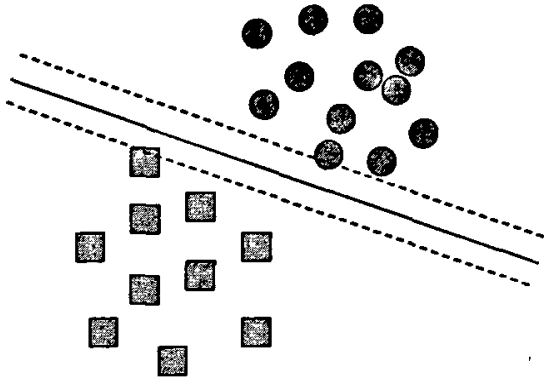

Figure 2.3 A Decision Line (solid) with a Smaller Margin [34]

The solid line in figures 2.3 shows a possible decision surface, it separates the two groups of data correctly. The two dashed lines parallel to the solid one show how much one can move the decision surface without causing misclassification of the data. The distance between each set of those parallel lines is referred to as "the margin". Then the aim of SVM becomes to find the decision surface that maximizes the margin between the data points in a training set.

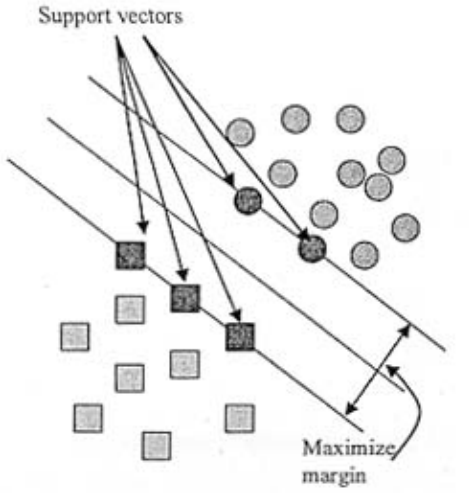

Figure 2.4 The Decision Line with the Maximal Margin [34]

Figure 2.4 shows the optimal decision surface with the maximal margin. Note that the decision surface is determined only by the data points which have exactly the same distance $\frac{1}{\|\vec{w}\|}$ from the decision plane. Those points are called the support vectors, which are the only effective data points in the training set. This makes SVM theoretically unique and different from many other methods, such as kNN, NB etc. where all the data points in the training set are used to optimize the decision function.

2.5.1.3. Evaluation Metrics

The experimental evaluation of text categorization classifiers is a very important issue. Through evaluation, the performances such as the capability of taking the right categorization decisions, etc. of a classifier are perspicuous and clear, making analysis of a classifier become more precise. Many measures have been proposed and used; each of them can evaluate some aspects of the performance of a classifier. Some commonly used evaluation methods in the literature will be described here including precision, recall, F-measure, etc.

A. Precision and Recall

The use of precision and recall to evaluate the performance of a classifier is adapted from the evaluation of classical Information Retrieval notions. Precision is defined as the ratio of correct assignments by the classifier divided by the total number of the classifier's assignments. Recall is defined to be the ratio of correct assignments by the classifier divided by the total number of correct assignments.

Given a category c_i, the golden standard set which was annotated by experts, and

the output results of the classifier, a true-false table as shown in Table 2.2 could be obtained:

Table 2.2 The True-False Table of the Classifier's Output

		Golden standard	
		Positive	Negative
Classifier output	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Let TP_i denote the number of True Positive samples, FP_i denote the number of False Positive samples, FN_i denote the number of False Negative samples, and TN_i denote the number of True Negative samples, then precision and recall of c_i can be formulated as:

$$precision_i = \frac{TP_i}{TP_i + FP_i}$$
$$recall_i = \frac{TP_i}{TP_i + FN_i} \qquad \text{Equation 2.2}$$

Note that the precision and recall are mutually exclusive, to obtain a high precision usually means sacrificing recall and vice versa. Normally a good classifier is required a proper trade-off between precision and recall. If the precision and recall are tuned to have an equal value, then this value is called the break-even point of the classifier. Figure 2.5 shows the relationship between precision and recall.

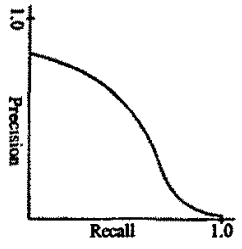

Figure 2.5 The Relationship Curve between Precision and Recall

Equation 2.2 gives the definition of precision and recall for individual category. For evaluating performance of the whole category set, there are two different methods,

namely macro-averaging and micro-averaging. Macro-averaging scores are calculated by first computing the measure scores per category, and then averaging over the results of the different categories. While Micro-averaging values are obtained by globally summing over all individual decisions together. The important distinction between the two types of averaging is: Micro-averaging gives equal weight to every document, while Macro-averaging gives equal weight to each category. It is understood that the micro-averaging scores tend to be dominated by the classifier's performance on common categories, and that the macro-averaging scores are more influenced by the performance on rare categories.

If there are m different categories, the Micro-averaging and Macro-averaging can be defined as:

Micro-averaging:

$$precision = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + FP_i}$$

$$recall = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + FN_i}$$

Equation 2.3

Macro-averaging:

$$precision = \frac{\sum_{i=1}^{m} precision_i}{m}$$

$$recall = \frac{\sum_{i=1}^{m} recall_i}{m}$$

Equation 2.4

B. F-measure

Another evaluation metric that combines precision and recall together is the F-measure:

$$F_\beta = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall}$$

Equation 2.5

where β is a parameter allowing different weight of precision and recall, it may be seen as the relative degree of importance attributed to precision and recall. β = 1 means that precision and recall have the same weight. F_1 is the most widely used one in text categorization task.

C. Accuracy and Error

Besides precision and recall, accuracy and error can also be used to evaluate the performance of a classifier, but they are not widely used as precision and recall. The

reason for this is that, the typically large value of their denominator makes them much more insensitive to a variation in the number of correct decisions than precision and recall. Equation 2.6 shows the definition of accuracy and error.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
$$error = \frac{FP + FN}{TP + FP + TN + FN}$$

Equation 2.6

2.5.2 Ranking

Learning to rank (ranking for short) is an important research topic in machine learning, because many issues in Information Retrieval and data mining can be formalized as ranking problems. Learning to rank aims at learning a scoring function that could order (sort) given objects according to some rules. Learning to rank is widely used in many research areas, e.g., in Information Retrieval, the user types a query, and the system calculates the relevance scores of documents with respect to the query and returns the documents in descending order of the relevance scores. The relevance scores can be calculated by a ranking function constructed with machine learning; in economic, some models often uses learning to rank [36]; and in statistical area, ranking models are commonly used, too [37].

In learning to rank a number of categories are given and a total order is assumed to exist over the categories. Labeled instances are provided. Each instance is represented by a feature vector, and each label denotes a rank. Existing methods fall into two categories. They are referred as "point-wise training" and "pair-wise training". In point-wise training, each instance (and its rank) is used as an independent training example. The goal of learning is to correctly map instances into intervals. In pair-wise training each instance pair is used as a training example and the goal of training is to correctly find the differences between ranks of instance pairs [38].

Learning to rank is related to classification and regression, but differs from both of them. The output space of classification is discrete and disordered, e.g., the output space of binary classification is {+1, -1}, where +1 and -1 are just category labels, and there are not any orders between them. The output space of regression is a set of real number or a range which contains infinite elements; order and measurement between these elements are defined. Compared to classification, the output space of learning to rank also contains finite elements, but orders are defined between elements, e.g., given an

output space {relevant, partially relevant, partially irrelevant, irrelevant}, the order between them is: relevant > partially relevant > partially irrelevant > irrelevant. Compared to regression, there are not any measurements between the elements in the output space of learning to rank, e.g., in the space of real number, the distance between element a and b can be computed as $|a-b|$, but in the above output space of learning to rank, measurements are not defined between relevant, partially relevant, partially irrelevant, and irrelevant, hence no distance can be computed for them. Figure 2.6 illustrates the comparison between them.

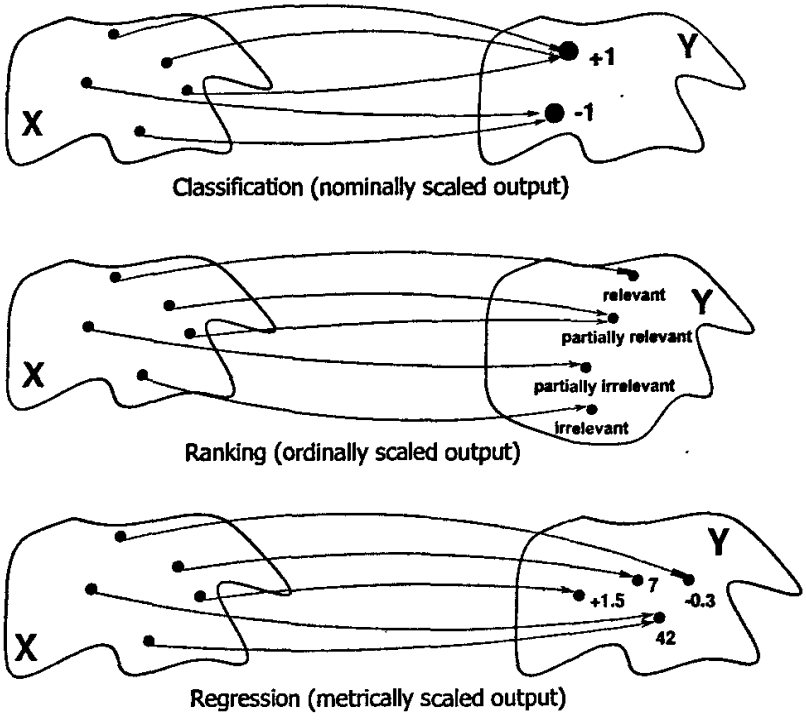

Figure 2.6 Comparison of Output Space between Classification, Ranking and Regression

2.5.2.1 Ranking SVM

Ranking SVM is a typical method of learning to rank. Ranking SVM formalizes learning to rank as a problem of classifying instance pairs into two categories (correctly ranked and incorrectly ranked).

Assume that there exists an input space $X \in R^n$, where n denotes number of features. There exists an output space of ranks (categories) represented by labels $Y = \{r_1, r_2, \cdots, r_q\}$ where q denotes number of ranks. Further assume that there exists a total order between the ranks $r_q \succ r_{q-1} \succ \cdots \succ r_1$, where $\succ$ denotes a preference relationship. A set

of ranking functions $f \in F$ exists and each of them can determine the preference relations between instances [38]:

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow f(\vec{x}_i) > f(\vec{x}_j) \quad \text{Equation 2.7}$$

Suppose that a set of ranked instances $S = \{(\vec{x}_i, y_i)\}_{i=1}^{l}$ are given from the space $X \times Y$. The task here is to select the best function f^* from F that minimizes a given loss function with respect to the given ranked instances.

Herbrich et al. [39] propose formalizing the above learning problem as that of learning for classification *on pairs of instances.*

First, assume that f is a linear function.

$$f_{\vec{w}}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle \quad \text{Equation 2.8}$$

where $\vec{w}$ denotes a vector of weights and $\langle \cdot, \cdot \rangle$ stands for an inner product. Plugging Equation 2.8 into Equation 2.7, it obtains:

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow \langle \vec{w}, \vec{x}_i - \vec{x}_j \rangle > 0 \quad \text{Equation 2.9}$$

Note that the relation $\vec{x}_i \succ \vec{x}_j$ between instance pairs $\vec{x}_i$ and $\vec{x}_j$ is expressed by a new vector $\vec{x}_i - \vec{x}_j$. Next, take any instance pair and their relation to create a new vector and a new label. Let $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ denote the first and second instances, and let $y^{(1)}$ and $y^{(2)}$ denote their ranks, then Equation 2.10 will be obtained:

$$\left(\vec{x}^{(1)} - \vec{x}^{(2)}, z = \begin{cases} +1 & y^{(1)} \succ y^{(2)} \\ -1 & y^{(1)} \prec y^{(2)} \end{cases} \right) \quad \text{Equation 2.10}$$

From the given training data set S, a new training data set S' containing l labeled vectors can be created.

$$S' = \{\vec{x}_i^{(1)} - \vec{x}_i^{(2)}, z_i\}_{i=1}^{l} \quad \text{Equation 2.11}$$

Next, S' is taken as classification data and a SVM model could be constructed that

can assign either positive label $z = +1$ or negative label $z = -1$ to any vector $\vec{x}^{(1)} - \vec{x}^{(2)}$.

Constructing the SVM model is equivalent to solving the following Quadratic Optimization problem:

$$\min_{\vec{w}} (\vec{w}) = \frac{1}{2}\left\|\vec{w}\right\|^2 + c\sum_{i=1}^{l}\xi_i$$
$$\text{subject to } \xi_i \geq 0,\ z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle \geq 1 - \xi_i \quad i = 1,\cdots,l \qquad \text{Equation 2.12}$$

Note that the optimization in Equation 2.12 is equivalent to that in Equation 2.13, when $\lambda = \frac{1}{2C}$ [40].

$$\min_{\vec{w}} \sum_{i=1}^{l}\left[1 - z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle\right]_+ + \lambda\left\|\vec{w}\right\|^2 \qquad \text{Equation 2.13}$$

where subscript "+" indicates the positive part. The first term is the so-called empirical Hinge Loss and the second term is regularizer.

Suppose that $\vec{w}^*$ is the weights in the SVM solution. Geometrically $\vec{w}^*$ forms a vector orthogonal to the hyperplane of Ranking SVM. $\vec{w}^*$ can be utilized to form a ranking function $f_{\vec{w}^*}$ for ranking instances.

$$f_{\vec{w}^*}(\vec{x}) = \left\langle \vec{w}^*, \vec{x} \right\rangle \qquad \text{Equation 2.14}$$

2.5.2.2 Evaluation Measures

There are some measures for evaluating the results of ranking methods such as precision, recall, Mean Reciprocal Rank (MRR), Mean Average Precision (MAP) [41], Normalized Discounted Cumulative Gain (NDCG) [42, 43], etc., each of them can evaluate some aspects of the performance of a ranking methods.

A. MRR

MRR is an evaluation method which mainly measures the position of the first correct answer in the retrieved answers. The calculation formulas for reciprocal answer rank (RR) and MRR are as follows:

$$RR_i = \frac{1}{rank_i} \qquad \text{Equation 2.15}$$

$$MRR = \frac{1}{N}\sum_{i=1}^{N} RR_i \qquad \text{Equation 2.16}$$

where N represents the number of queries in the testing set. $rank_i$ represents the rank (i.e., the position of the first correct answer) of the ith query. If the correct answer is not included in the list of answers, its rank is set at infinite.

B. MAP

MAP is widely used in IR and is based on the assumption that there are two categories: positive (relevant) and negative (irrelevant) in ranking of instances (documents). MAP calculates the mean of average precisions over a set of queries. Given a query q_i, average precision is defined as the average of precision after each positive (relevant) instance is retrieved. Given a query q_i, its average precision ($AvgP_i$) is calculated as:

$$AvgP_i = \sum_{j=1}^{M} \frac{p(j) * pos(j)}{number\ of\ positive\ instances} \qquad \text{Equation 2.17}$$

where j is the rank, M is the number of instances retrieved, $pos(j)$ is a binary function to indicate whether the instance in the rank j is positive (relevant), and $p(j)$ is the precision at the given cut-off rank j.

$$p(j) = \frac{\text{number of positive instances in top } j \text{ positions}}{j} \qquad \text{Equation 2.18}$$

C. NDCG

Unlike MAP, NDCG takes into account degrees of relevance. NDCG is based on the assumption that there are more than two ranks for relevance ranking while MAP is based on the assumption that there are two categories: relevant and irrelevant. To calculate this measure it is necessary to go through a number of steps to determine the gain, the cumulative gain, the discounted cumulative gain and the ideal discounted cumulative gain.

The Gain (G) of each document is its relevance score. Thus,

$$G[n] = Score_{doc\_i} \qquad \text{Equation 2.19}$$

where $Score_{doc\_i}$ is the score of the ith document in the retrieved list.

Cumulative gain (*CG*) is calculated as follows.

$$\begin{aligned} CG[1] &= G[1] \\ CG[i] &= CG[i-1] + G[i] \quad if\, i > 1 \end{aligned} \qquad \text{Equation 2.20}$$

Discounted cumulative gain (*DCG*) is similar to cumulative gain but a discount factor is applied to reflect the decreased utility of documents retrieved further down the ranking. Usually base $b = 2$ is used for the discount factor;

$$\begin{aligned} DCG[i] &= CG[i] \quad if\, i < b \\ DCG[i] &= DCG[i-1] + G[i]/\log_b i + 1 \quad if\, i >= b \end{aligned} \qquad \text{Equation 2.21}$$

Normalized discounted cumulative gain (*NDCG*) is the ratio of the discounted cumulative gain and the ideal discounted cumulative gain. The way to compute the ideal discounted cumulative gain (*IDCG*) is similar to calculating discounted cumulative gain, but using an ideal ranking. The ideal ranking arranged documents in descending order of relevant scores. It starts with all the highest scores, followed by lower scores and then all lowest ones.

$$NDCG[i] = \frac{DCG[i]}{IDCG[i]} \qquad \text{Equation 2.22}$$

3 Cascaded Hybrid Model for Chat Knowledge Acquisition

3.1 Task Definition and Model Description

An input online discussion forum F contains discussion sections $s_1,s_2,...,s_k$. A section consists of T threads $t_1,t_2,...,t_u$. Each thread t is a sequence of replies $t=\{r_0,r_1,r_2,...,r_n\}$, where r_0 is the root message posted by the thread starter and r_i is the ith $(i\geq 1)$ reply. A reply r is posted by a participant p at a specific moment m with content c. A thread t can be modeled as a sequence of triplets:

$$t=\{r_0,r_1,r_2,...,r_n\}$$
$$=\{(p_0,m_0,c_0),(p_1,m_1,c_1),(p_2,m_2,c_2),...,(p_n,m_n,c_n)\}$$

RR is defined as a direct reply r_j $(j\geq 1)$ to the root message r_0 where r_j is not correlated with the other reply $r_{j'}(j'\geq 1 \wedge j'\neq j)$ in the thread.

Therefore, chatbot knowledge (*CK*) can be viewed as the pairs of <input, response> that fulfill the following constraints:

$$CK=\{(input,response)\}$$
$$=\{(thread\text{-}title,\ high\text{-}quality\ RR)\}$$

A thread title is used to model the user input of a chatbot and *RRs* of this thread are used to model the chatbot responses. The high-quality pairs of <thread-title, *RR*> will be selected as chatbot knowledge.

A high-quality pair of <thread-title, *RR*> for the chatbot should meet the following requirements:

1) The thread-title is meaningful and popular.
2) The *RR* provides descriptive, informative and trustworthy content to the root message.
3) The *RR* has high readability, neatly short and concise expressive style, clear structure.
4) The *RR* is attractive and can capture chatter's interest.
5) Both thread-title and *RR* should have NO intemperate sentiment, no obscene words and exclusive personal information.
6) Both thread-title and *RR* should have proper length.

In this paper, identifying the qualified thread-title is not the focus. Instead, the focus is to select qualified *RR*. Figure 3.1 illustrates the structure of the cascaded model. The first pass (on the left-hand side) applies an SVM classifier to the candidate *RR* to identify the *RR* of a thread. Then the second pass (in the middle) filters out the *RR* that

contains intemperate sentiment, obscene words and personal information with a predefined keyword list. The *RR* which is longer than a predefined length is also filtered out. Finally the *RR* ranking module (on the right-hand side) is used to extract the descriptive, informative and trustworthy replies to the root message.

Figure 3.1 Structure of Cascaded Model

3.2 *RR* Identification

Since it only needs to distinguish *RRs* from *non-RRs*, hence the task of *RR* identification can be viewed as a binary classification problem of distinguishing *RR* from *non-RR*. The approach used here is to assign a candidate reply $r_i (i \geq 1)$ an appropriate class y (+1 if it is an *RR*, -1 or not).

Table 3.1 Features for *RR* Classifier

1	**Structural features**
1-1	Does this reply quote root message?
1-2	Does this reply quote other replies?
1-3	Is this reply posted by the thread starter?
1-4	# of replies between same author's previous and current reply
2	**Content features**
2-1	# of words
2-2	# of content words of this reply
2-3	# of overlapping words between thread-title and reply
2-4	# of overlapping content words between thread-title and reply
2-5	Ratio of overlapping words
2-6	Ratio of overlapping content words between thread-title and reply
2-7	# of domain words of this reply
2-8	Does this reply contain other participants' registered nicknames in forum?

Here SVM is selected as the classification model because of its robustness to over-fitting and high performance [27, 34, 44].

SVMlight [45] is used as the SVM toolkit for training and testing. Table 3.1 lists the feature set to identify *RR* for a pair of <thread-title, reply>.

In this paper, both structural and content features are selected. In structural features, quotation maintains context coherence and indicates the relevance between the current reply and the quoted root message or reply, as discussed in [18, 19]. Two quotation features (feature 1-1 and feature 1-2) are employed in the classifier. Feature 1-1

indicates that the current reply quoting the root message is relevant to the root message. On the contrary, feature 1-2 indicates the current reply might be irrelevant to the root message because it quotes other replies. Features 1-3 and 1-4 are selected based on the observation of behaviors of posting replies in forums. The thread starter, when participants reply to the starter's thread, usually adds new comments to the replies. Therefore, the added replies gradually diverge from the original root message. If a participant wants to supplement or clarify his previous reply, he can add a new reply. Therefore, the participant's new reply is often the supporting reason or argument to his previous reply if they are close to each other.

Content features include the features about the number of words and the number of content words in the current reply, the overlapping words and content words between the root message and the current reply. In this work, words that do not appear in the stop word list[8] are considered as content words. Feature 2-7 estimates the specialization of the current reply by the number of domain specific terms. To simplify the identification of domain specific terms, here words that do not appear in a commonly used lexicon (consists of 73,555 English words) are extracted as domain specific simply. Feature 2-8 estimates a reply's pertinence to other replies, because some participants might insert the registered nicknames of other participants and sometimes add clue words such as "P.S." to explicitly correlate their replies with certain participants.

3.3 *RR* Ranking

Further, after the *RRs* have been identified, non-eligible *RRs* are filtered out with a keyword list with 33 obscenities, 62 personal information terms (terms beginning with "my", such as my wife, my child) and 17 forum specific terms (such as Tomatometer, Rotten Tomato, etc.). Replies with more than N words are eliminated because people may become bored in chatbot scenarios if the response is too long. In the experiments, N is set as 50 based on the observation.[9]

After that, an analysis to the resulting *RRs* set is carried out. For some *RRs*, there is certain noise left from the previous pass, while for other *RRs*, there are too many *RRs* with varied qualities. Therefore, a way is needed to extract the top-N *RRs* which have highest quality, while the noisy *RRs* could be ignored. Based on analyses and comparison, to meet these requirements, the task can be treated as a ranking problem to

[8] http://dvl.dtic.mil/stop_list.html
[9] 50 is the average length of 1,200 chatbot responses which preferred by three chatters through sample experiments.

select the high-quality *RRs*.

The ranking SVM [46] is employed to train the ranking function using the feature set in Table 3.2.

Table 3.2 Features for *RR* Ranking

1	**Feature of the number of being quoted**
1-1	# of quotations of this reply within the current thread
2	**Features from the author of a reply**
2-1	# of threads the author starts in the forum
2-2	# of replies the author posts to others' threads in the forum
2-3	The average length of the author's replies in the forum
2-4	The longevity of participation
2-5	# of the author's threads that get no replies in the forum
2-6	# of replies the author's threads get in the forum
2-7	# of threads the author is involved in the forum
2-8	The author's total influence in the forum
2-9	# of quotations of the replies that are posted by the author in current thread
2-10	# of quotations of all the replies that are posted by the author in the forum

The number of being quoted of a reply is selected as a feature (feature 1-1) because a reply is likely to be widely quoted within a thread as it is popular or the subject of debate. In other words, the more times a reply is quoted, the higher quality it may have. This motivates us to extract the quoted number of all the other replies posted by an author within a thread (feature 2-9) and throughout the forum (feature 2-10).

"Author reputation" is also taken into account when assessing the quality of a reply. The motivation is that if an author has a good reputation, his reply is more likely to be reliable. Here the author behavior related features are used to assess his "reputation." An earlier work investigates the relationship between a reader's selection of a reply and the author of this reply, and found that some of the features raised from authors' behavior over time, correlate to how likely a reader is to choose to read a reply from an author [47]. Features 2-1 to 2-7 are author behavior related features in the forum. Feature 2-8 models how many people have chosen to read the threads or replies of an author in the forum by using the measurement of the influence of participants. This is described in detail in [48].

3.4 Summary

To formulate the chatbot knowledge acquisition task, a formal definition on chat knowledge extraction from online discussion forums is given. After that, a cascaded

model is proposed. To filter out *non-RRs*, a binary classification model is proposed, effective features with respect to *RR* identification task are selected. After irrelevant replies are removed in the first pass, a ranking model is then used to extract high-quality *RRs*. After analyzing the characteristics of the remaining *RRs* carefully, salient features such as author related features are taken into account for *RR* ranking. With this cascaded model, different features can be optimally employed at different passes. The next chapter will report the result and evaluation of each pass. An end to end evaluation is also carried out to understand to what extent that the proposed approach could contribute to the chatbot construction in practice.

4 Experimental Results

4.1 Data for Experiments

In the experiments, the *Rotten Tomatoes forum*[10] is used as test data. It is one of the most popular online discussion forums for movies and video games. The *Rotten Tomatoes forum* discussion archive is selected because each thread and its replies are posted by movie fans, amateur and professional filmmakers, film critics, moviegoers, or movie producers. This makes the threads and replies more heterogeneous, diverse, and informative.

For research purposes, the discussion records are collected by crawling the *Rotten Tomatoes Forum* over the time period from November 11, 1999 to June 15, 2005. The downloaded collection contains 1,767,083 replies from 65,420 threads posted by 12,973 distinctive participants, so there are, on average, 27.0 replies per thread, 136.2 replies per participant, and 5.0 threads per participant. The number of thread titles in question form is 16,306 (24.93%) and in statement form is 49,114 (75.07%). Part of these discussion records are used in the experiments.

4.2 *RR* Identification Results

To build the training and testing dataset, 53 threads are randomly selected and manually tagged from the *Rotten Tomatoes* movie forum, in which the number of replies was between 10 (min) and 125 (max). There were 3,065 replies in 53 threads, i.e., 57.83 replies per thread on average. Three human experts were hired to manually identify the relevance of the replies to the thread-title in each thread. Experts annotated each reply with one of the three labels: a) *RR*, b) *non-RR* and c) *Unsure*. Replies that received two or three *RR* labels were regarded as *RR*, replies with two or three *non-RR* labels were regarded as *non-RR*. All the others were regarded as *Unsure*.

After the labeling process, 1,719 replies (56.08%) were *RR*, 1,336 replies (43.59%) were *non-RR*, 10 (0.33%) were *Unsure*. Then 10 unsure replies and 60 replies with no words were removed. 35 threads were randomly selected for training (including 1,954 replies) and 18 threads for testing (including 1,041 replies). The baseline system used the number of replies between the root message and the responding reply [12] as the feature to classify *RRs*.

[10] http://www.rottentomatoes.com/vine/

Table 4.1 provides the performance using SVM with the feature set described in Table 3.1.

Table 4.1 *RR* Identification Result

Feature set	Precision	Recall	F-score
Baseline	73.24%	66.86%	69.90%
Structural	89.47%	92.29%	90.86%
Content	71.80%	85.86%	78.20%
All	90.48%	92.29%	91.38%

With only the structural features, the precision, recall and f-score reached 89.47%, 92.29%, and 90.86%. Content features, when used alone, the precision, recall and f-score are low. But after adding content features to structural features, the precision improved by 1.01% while recall stayed the same. This indicates that content features help to improve precision.

Root message
Title: *Recommend Some Westerns For Me?*
Description: *And none of that John Wayne sh*t.*

1. *The Wild Bunch It's kickass is what it is.*
2. *Once Upon a Time in the West*
3. *Does Dances With Wolves count as a western? Doesn't matter, I'd still recommend it.*
4. *White Comanche This masterpiece stars .....*
5. *Here's some I'm sure nobody else .....*
6. *for Dances with Wolves.*
7. *: understands he's a minority here: .....*
8. *Open Range is really good. Regardless .....*
9. *One of the best films I've ever seen.*
10. *The Good the Bad and the Ugly .....*

Figure 4.1 A Sample of *RRs*

Figure 4.1 presents some identified *RRs* listed in chronological order for the root message with the title, "*Recommend Some Westerns For Me?*" and description for the title, "*And none of that John Wayne sh*t.*".

4.3 Extract High-quality *RR*

To train the ranking SVM model, an annotated dataset was required. After the non-eligible *RRs* were filtered out from the identified *RRs*, three annotators labeled all of the remaining *RRs* with three different quality ratings. The ratings and their

descriptions are listed in Table 4.2.

Table 4.2 *RR* Rating Labels

Rating	Description
Fascinating	This reply is informative and interesting, and it is suitable for a chatbot
Acceptable	The reply is just so-so but tolerable
Unsuitable	This reply is bad and not suitable for a chatbot

After the labeling process, there were 568 (71.81%) *fascinating RRs*, 48 (6.07%) *acceptable RRs*, and 175 (22.12%) *unsuitable RRs* in the 791 *RRs* of the 35 training threads. And in the 511 *RRs* of the 18 test threads, there were 369 (72.21%) *fascinating RRs*, 25 (4.89%) *acceptable RRs*, and 117 (22.90%) *unsuitable RRs*.

Table 4.3 MAP Score

Feature set	Baseline	All
MAP	82.33%	86.50%

MAP is used as the metric to evaluate *RR* ranking here. The baseline ranked the *RRs* of each thread by their chronological order. Table 4.3 shows the performance of the ranking function with the feature set in Table 3.2. The ranking function achieved high performance (MAP score is 86.50%) compared with the baseline (MAP score is 82.33%). Content features such as the cosine similarity between an *RR* and the root message were also tried, and result showed that they could not help to improve the ranking performance. The MAP score was reduced to 85.23% when the cosine similarity feature was added to the feature set.

4.4 Chat Knowledge Extraction with Proper *N* Setting

The chat knowledge extraction task requires that the extracted *RRs* should have high quality and high precision. After getting the ranked *RRs* of each thread, the Top-*N* *RRs* were selected as chatbot responses. The baseline system just selected Top-*N* *RRs* ranked in chronological order. Figure 4.2 shows the comparison of the performances of the approach and the baseline system at different settings of *N*.

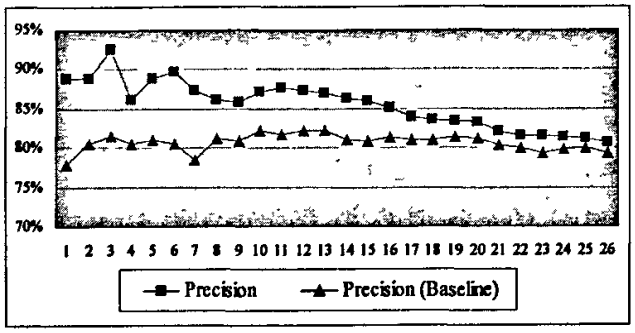

Figure 4.2 Precision at Different *N*

Figure 4.3 shows the Top-*N* (*N*=6, *N* can be adjusted to get proper equilibrium between quantity and quality of *RRs* when extracting chatbot knowledge) *RRs* after ranking the *RRs* in Figure 4.1. As an instance, Top-6 high-quality *RRs* from each thread were uniformly extracted. Altogether 108 <thread-title, reply> pairs were generated from 18 threads. Among these extracted pairs, there were 97 fascinating pairs and 11 wrong pairs, which showed that 89.81% of the extracted chatbot knowledge was correct.

Input: *Recommend Some Westerns For Me?*
Chatbot responses:
6. for Dances with Wolves.
11. Young Guns! & Young Guns 2!
2. Once Upon a Time in the West
9. One of the best films I've ever seen.
27. I second the dollars trilogy and also Big Hand .....
18. Classic Anthony Mann Westerns: The Man from Laramie (1955) .....

Figure 4.3 Top-6 *RRs*

4.5 End to End Evaluation

4.5.1 Evaluation Method

To understand to what extent that the proposed approach could contribute to the chatbot construction in practice, two experiments were carried out. The first one was designed to test the effectiveness of the automatic chat knowledge extraction approach against the hand-coded method. The second experiment evaluated the effectiveness of the proposed approach for the construction of a chatbot.

4.5.2 Automatic Extraction vs. Hand-coded

Two human annotators with some background on movies were asked to write 18

topics and write 6 responses for each of them. The observing of the authoring process showed that it was not as easy for them to think out 18 topics and it took them about 2 hours to finish the task. The coverage of their proposed topics was much narrower than the extracted topics. The responses of each topic contained many repetitive and obsolete ones. Then the automatically extracted 18 threads and 6 replies per thread were showed to them, and let the two human annotators to proofread them, this process was finished within 10 minutes. Then a small high quality chat knowledge base was built quickly by using their proofreaded replies and the 18 threads.

4.5.3 A Pilot Chatbot for Movie Domain

A chatbot was built based on the AskBill architecture described in Chapter 1 for the movie domain by simply feeding <thread-title, reply> pairs extracted from a part of the *Rotten Tomatoes Forum*.

The chat knowledge base for the movie domain was built in this way. First, the high quality pairs of <thread-title, reply> from the *Rotten Tomatoes Forum* were extracted. The repetitive pairs were removed. Then the left <thread-title, reply> pairs were converted to the format of chatbot knowledge representation markup language. A sample of chat knowledge representation is given in Figure 4.4.

Using the chat knowledge acquisition approach proposed, 11,147 <thread-title, reply> pairs were extracted from the randomly selected 2,000 threads (3.06% of the total 65,420 threads) from the *Rotten Tomatoes Forum*, then they were imported into the chat knowledge without any human post editing. As the result, a pilot chatbot that could simply chat on movie on these 2,000 topics was built efficiently.

```
<topic value = "Movie::General Discussion">
<category>
  <input>Recommend Some Westerns For Me</input>
  <response>
    <random>
      <list rank ="1"> for Dances with Wolves.</list>
      <list rank ="2"> Young Guns! & Young Guns 2! </list>
      <list rank ="3"> Once Upon a Time in the West </list>
      <list rank ="4"> One of the best films I've .....</list>
      <list rank ="5"> I second the dollars trilogy .....</list>
      <list rank ="6"> Classic Anthony Mann .....</list>
    </random>
  </response>
</category>
</topic>
```

Figure 4.4 Chat Knowledge Representation

A careful check was conducted with the extracted pairs in order to understand the qualities and analyze the problems. First, checking whether the titles of the threads were qualified for a chatbot was carried out. After checking, in the 2,000 threads, there were 1,892 threads had qualified titles and 108 threads were not suitable as input for a chatbot because the titles of these threads were nonsense or meaningless such as "*YAET*" and "*Best....*". Then a checking to the *RRs* of the threads that had valid titles was carried out, as a result, there were 9,487 pairs that could be used as chat knowledge. In the invalid pairs, 518 replies were not self inclusive, 97 replies actually proposed questions but their thread titles were questions, and other invalid pairs contained incomplete sentences and could not be fully understood. Removing these pairs will be the future work.

5 Comparison with Related Work

Previous works have utilized different datasets for knowledge acquisition for different applications. Shrestha and McKeown [11] use an email corpus. Zhou and Hovy [12] use Internet Relay Chat and use clustering to model multiple sub-topics within a chat log. This work is the first to explore using the online discussion forums to extract chatbot knowledge. Since the discussions in a forum are presented in an organized fashion within each thread in which users tend to respond to and comment on specific topics, it only needs to identify the *RRs* for each thread here. Hence, the clustering becomes unnecessary. Furthermore, a thread can be viewed as <input, response> pairs, with the same structure of chat template of a chatbot, making a forum better suited for the chatbot knowledge extraction task.

The use of thread title as input means that identifying relevant replies to the root message (*RRs*) must be carried out, much like finding adjacent pairs (APs) in [12] but for the root message. They utilize AP to identify initiating and responding correspondence in a chat log since there are multiple sub-topics within a chat log, while *RR* is used to identify relevant response to the thread-title in this paper. Similarly, an SVM classifier is applied to identify *RRs* but more effective structural features are employed. Furthermore, high-quality *RRs* are extracted with a ranking function.

Xi *et al.* [13] use a ranking function to select the most relevant messages to user queries in newsgroup searches, and in which the author feature is proved not effective. In this work, the author feature also proves not effective in identifying relevant replies but it is proved effective in selecting high-quality *RRs* in *RR* ranking. This is because irrelevant replies are removed in the first pass, making author features more salient in the remaining *RRs*. This also indicates that the cascaded framework outperforms the flat model by optimally employing different features at different passes.

6 Conclusions and Future Work

This paper has presented an effective approach to extract <thread-title, reply> pairs as knowledge of a chatbot for a new domain. The contribution can be summarized as follows:

1) Perhaps for the first time, this work proposes using online discussion forums to extract chatbot knowledge.
2) A cascaded framework is designed to extract the high-quality <thread-title, reply> pairs as chabot knowledge from forums. It can optimally use different features in different passes, making the extracted chatbot knowledge of higher quality.
3) It shows through experiments that structural features are the most effective features in identifying *RR* and author features are the most effective features in identifying high-quality *RR*.

Compared with manual knowledge construction methods, the approach proposed is more efficient in building a specific domain chatbot. In the experiment with a movie forum domain, 11,147 <thread-title, reply> pairs were extracted from 2,000 threads within two minutes. It is simply not feasible to have human experts encode a knowledge base of such size.

As future work, the author plan to improve the qualities of the extracted *RRs*. The method of selecting valid thread titles and extracting completed sentences from the extracted *RRs* is an area for exploration. In addition, the author is also interested in extracting questions from threads so that <question, reply> pairs can be used to support QA style chat.

In this paper, the extracted <thread-title, reply> pairs are directly fed into the chatbot knowledge base. But there is much room to improve quality in the future. For example, the chat templates can be generalized by clustering similar topics and grouping similar replies, and coherence among the consecutive chat replies can be improved by understanding the styles of replies.

Acknowledgements

This master thesis is the result of so much appreciated help that had come to me during my master studies directly or indirectly. Without the guidance, advices, ideas, help, and challenges from my two supervisors and my mentor in Microsoft Research Asia, and without the warmth, care, and compassion of friends and family, my work would be non-existent or meaningless.

First I would like to thank my supervisor, Prof. Dan Yang, who gave me invaluable advice, excellent suggestions, constants support and helpful discussion during my study. Our discussions and his constructive comments have greatly improved this work. He is also thanked for giving me great and constant support to be an intern at Microsoft Research Asia.

I would also like to thank my associate supervisor, Assoc. Prof. Junhao Wen, for the patient guidance, encouragement and advice he has provided throughout my study time as his student. He is also thanked for being a cherished friend who shares the experience of study and living with me, and for challenging me to strive for excellence.

Part of the work is finished when I was visiting Microsoft Research Asia. My special and deep appreciation goes to my mentor in Microsoft Research Asia, Dr. Ming Zhou, for providing divine inspiration, great encouragement, dedicated guidance and support throughout my internship. Thank you for your patient and attentive instruction for bringing me into the Natural Language Processing (NLP) research area and opening the door of knowing something about advanced project management in Microsoft and doing some cutting-edge research and engineering work in the NLP field. Also thanks for all of the meetings throughout my internship, leading and guiding, reading endless drafts of proposals until the research paper submitted to ACL and IJCAI came together. In the future, when I am in the position to supervise others, I have a great role model.

I benefit much from the three supervisors. Their enthusiasm and serious attitude toward study and research, and insistent requirements for excellence of their works have made me deeply realize their research states of mind that are far above the secular standards. I will remember their instructions forever, and carry out them with practical acts.

I am very grateful to Dr. Cheng Niu, Yajuan Lv, Zhihao Li and other researchers in NLC group of Microsoft Research Asia for their valuable help on research. I wish to

thank Dwight, Eileen, Xin Ma, Gang Guan, Wen Chen, etc. in Microsoft Research Asia for your kind and warm-hearted help.

Thanks also goes to friends made in Microsoft Research Asia such as Jingjing Liu, Shiqi Zhao, Xiaoyuan Cui, Jun Xu, Yi Chen, Long Jiang, Jun Lang, Yunhua Hu, Wei Yuan, Chengjie Sun, Galen, Litian Tao, Hao Su, etc., completing the paper accepted by IJCAI 2007 would have been all the more difficult were it not for the support and friendship provided by you and the other members of Microsoft Research Asia. I am indebted to all the friends including the ones who are not listed one by one here for your help.

I want to thank Xiyi Zhang, Xi Luo and other teachers in School of Software Engineering for their great help during my study here.

I also express my gratitude for encouragers, classmates, and friends who are not listed here, but your help and encourage give me a lot of passion and make progress step by step on my study.

Finally, I heartily thank my family members. My family members especially my father and mother have been an inspiration throughout my life. They have always supported my dreams and aspirations. I'd like to thank them for all they have done for me. This thesis is a best present I can give to you.

Jizhou Huang

October 18, 2006

In Chongqing University

References

[1] J. Z. Huang, M. Zhou, and D. Yang. Extracting Chatbot Knowledge from Online Discussion Forums. To appear in *Proceedings of International Joint Conference on Artificial Intelligence 2007 (IJCAI-2007).*

[2] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. *Prentice Hall*, 2003, Second Edition.

[3] A. Turing. Computing machinery and intelligence. In *Mind, vol. LIX*, no. 236, October 1950, pp. 433-460.

[4] J. Weizenbaum. ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communications of the ACM*, 9(1):36-45, 1966.

[5] K. M. Colby. Simulation of Belief systems. In *Schank and Colby (Eds.) Computer Models of Thought and Language*, pp.251-286, 1973.

[6] Y. Chen, M. Zhou, and S. L. Wang. Reranking Answers for Definitional QA Using Language Modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp.1081-1088, Sydney, July 2006.

[7] S. Q. Zhao, M. Zhou, and T. Liu. Learning Question Paraphrases for QA from Encarta Logs. To appear in *Proceedings of International Joint Conference on Artificial Intelligence 2007 (IJCAI-2007).*

[8] B. A. Shawar and E. Atwell. Machine Learning from dialogue corpora to generate chatbots. In *Expert Update journal*, 6(3):25-29, 2003.

[9] P. Tarau and E. Figa. Knowledge-based conversational Agents and Virtual Story-telling. In *Proceedings 2004 ACM Symposium on Applied Computing*, 1:39-44, 2004.

[10] R. Nishimura, Y. Watanabe, and Y. Okada. A Question Answer System Based on Confirmed Knowledge Developed by Using Mails Posted to a Mailing List. In *Proceedings of the IJCNLP 2005*, pp.31-36, 2005.

[11] L. Shrestha and K. McKeown. Detection of question-answer pairs in email conversations. In *Proceedings of Coling 2004*, pp.889-895, 2004.

[12] L. Zhou and E. Hovy. Digesting Virtual "Geek" Culture: The Summarization of Technical Internet Relay Chats. In *Proceedings of ACL 2005*, pp.298-305, 2005.

[13] W. S. Xi, J. Lind, and E. Brill. Learning Effective Ranking Functions for Newsgroup Search. In *Proceedings of SIGIR 2004*, pp.394-401, 2004.

[14] J. W. Kim, K. S. Candan and M. E. Donderler. Topic Segmentation of Message Hierarchies for

Indexing and Navigation Support. In *Proceedings WWW 2005, 2005.*

[15] V. Tuulos and H. Tirri. Combining topic models and social networks for chat data mining. In *Proceedings of the Web Intelligence 2004*, pp.206-213. IEEE Computer Society, 2004.

[16] R. Agrawal, S. Rajagopalan, R. Srikant, and Y. Xu. Mining newsgroups using networks arising from social behavior. In *Proceedings of 12th International World Wide Web Conference*, 2003.

[17] P. Resnick, J. Riedl, L. Terveen, and M. Ackerman. Beyond threaded conversation. In *CHI '05 extended abstracts on Human factors in computing systems.* 2005.

[18] K. S. Eklundh and C. Macdonald. The Use of Quoting to Preserve Context in Electronic Mail Dialogues. In *IEEE Transactions on Professional Communication,* 37(4):197-202, 1994.

[19] K. S. Eklundh. To quote or not to quote: setting the context for computer-mediated dialogues. In *S. Herring (Ed.), Computer-Mediated Conversation.* Cresskill, NJ:Hampton Press, 1998.

[20] Y. Yang and J. P. Pedersen. Feature selection in statistical learning of text categorization. In *the 14th Int. Conf. on Machine Learning*, pp.412-420, 1997.

[21] D. Lewis and M. Ringuette. A comparison of two learning algorithms for text classification. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pp.81-93, 1994.

[22] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. 10th European Conference on Machine Learning (ECML), Springer Verlag,* 1998.

[23] W. J. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In *SIGIR'96: Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp.307-315, 1996.

[24] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrival System: Experiments in Automatic Document Processing*, pp.313-323, PrenticeHall Inc., 1971.

[25] S. M. Weiss, C. Apte and F. J. Damerau. Maximizing Text-Mining Performance. In *IEEE Intelligent Systems* 1999.

[26] E. Wiener, J. O. Pedersen and A. S. Weigend. A neural network approach to topic spotting. In *Proc. 4th annual symposium on document analysis and information retrieval*, pp.22-34, 1993.

[27] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys,* 34(1):1-47, 2002.

[28] K. Aas and L. Eikvil. Text categorisation: A survey. *Norwegian Computing Center, Raport NR* 941, June, 1999.

[29] L. D. Baker and A. K. McCallum. Distributional clustering of words for text categorisation. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval (Melbourne, AU, 1998)*, pp.96-103, 1998.

[30] P. J. Hayes, P. M. Andersen, I. B. Nirenburg, and L. M. Schmandt. Tcs:a shell for content-based text categorization. In *Proceedings of CAIA-90, 6th IEEE Confer-ence on Artificial Intelligence Applications (Santa Barbara, US, 1990)*, pp.320-326, 1990.

[31] M. Goodman. Prism: a case-based telex classifier. In *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pp.86-90, 1990.

[32] L. F. Rau and P. S. Jacobs. Creating segmented databases from free text for text retrieval. In *Proceedings of SIGIR-91, 14th ACM International Conference on Research and Development in Information Retrieval (Chicago, US)*, pp.337-346, 1991.

[33] T. M. Mitchell. Machine learning. *McGraw Hill*, New York, US, 1996.

[34] Y. M. Yang and X. Liu. A re-examination of text categorization methods. In *The 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp.42-49, 1999.

[35] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning* 20, pp.273-297, 1995.

[36] R. Herbrich, M. Keilbach, T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Neural networks in economics: Background, applications, and new developments. *Advances in Computational Economics*, 11:169~196, 1999.

[37] L. Fahrmeir and G. Tutz. Multivariate Statistical Modelling Based on Generalized Linear Models. *Springer-Verlag*, 1994.

[38] Y. B. Cao, J. Xu, T. Y. Liu, H. Li, Y. L. Huang, and H. W. Hon. Adapting Ranking SVM to Document Retrieval. *In Proceedings of SIGIR 2006*, pp.186-193, 2006.

[39] R. Herbrich, T. Graepel, and K. Obermayer. Large Margin Rank Boundaries for Ordinal Regression. *Advances in Large Margin Classifiers*, pp.115-132, 2000.

[40] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: data mining, inference and prediction. *Springer-Verlag*, 2001.

[41] R. A. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval, *Addison-Wesley Longman Publishing Co., Inc., Boston, MA*, 1999.

[42] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd International ACM SIGIR Conference*, pp.41-48, 2000.

[43] K. Jarvelin and J. Kekalainen. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4):422~446, 2002.

[44] D. D. Lewis, F. Li, T. Rose, and Y. Yang. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(3):361-397, 2004.

[45] T. Joachims. Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press, 1999.

[46] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pp.133-142, 2002.

[47] A. T. Fiore, S. Leetiernan, and M. A. Smith. Observed Behavior and Perceived Value of Authors in Usenet Newsgroups: Bridging the Gap. In *Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems*, pp.323-330, 2002.

[48] N. Matsumura, Y. Ohsawa, and M. Ishizuka. Profiling of Participants in Online-Community. *Chance Discovery Workshop on the Seventh Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, pp.45-50, 2002.

Appendixes

A. Publications

[1] Jizhou Huang, Ming Zhou and Dan Yang. Extracting Chatbot Knowledge from Online Discussion Forums. To appear in *Proceedings of International Joint Conference on Artificial Intelligence 2007 (IJCAI-2007).*

[2] 黄际洲，黄鑫，黄振. 露天矿生产的车辆安排. 2003 年全国大学生数学建模竞赛.

B. Literature Translation

[1] 黄际洲，文俊浩 译.《Direct 3D 中的 2D 编程》. 重庆大学出版社. 2005 年 1 月 1 日. ISBN: 7-5624-3294-5

[2] 文俊浩，黄际洲，吴红艳 译.《游戏编程 All in One》. 重庆大学出版社. 2005 年 10 月 1 日. ISBN: 7-5624-3375-5.

[3] 黄际洲，刘刚 译.《DirectX 角色扮演游戏编程》. 重庆大学出版社. 2006 年 2 月 1 日. ISBN: 7-5624-3542-1

C. Research Work

[1] Name: AskBill Chatbot project Source: Microsoft Research Asia Time: Feb.2005-Mar.2006