NTT Question Answering System in TREC 2001

Hideto Kazawa, Hideki Isozaki and Eisaku Maeda
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seikacho, Sorakugun, Kyoto, Japan
{kazawa,isozaki,maeda}@cslab.kecl.ntt.co.jp

Abstract

In this report, we describe our question-answering system SAIQA-e (System for Advanced Interactive Question Answering in English) which ran the main task of TREC-10's QA-track. Our system has two characteristics (1) named entity recognition based on support vector machines and (2) heuristic apposition detection. The MPR score of the main task is 0.228 and experimental results indicate the effectiveness of the above two steps in terms of answer extraction accuracy.

1 Introduction

To design a QA system, there are several choices to make. One is about what kind of technology the system should be based on. To date, some research works have attempted the Information Retrieval (IR) approach, assuming that the most relevant passages include the answers of questions. Generally speaking, this IR approach is fast and robust, but is unable to specify the 'exact answer', i.e., what part of the passage is really the answer. Another approach is the Information Extraction (IE) approach, where the system extracts candidate strings from documents and evaluates the validity of the candidates. This approach has the advantage of being able to specify the locations of exact answers although it is usually slow and often fails because of complicated natural language processing.

For TREC-10's QA track, we adopted the IE approach because we think that knowing the exact locations of answers is one of the important goals of QA. It also seems easier to reach the goal with the IE approach. To avoid 'deep' natural language processing, we decided to use only shallow linguistic analysis, i.e., part-of-speech tagging and base noun phrase (NP) chunking. To proceed with this decision, we mainly focused on the following problems.

1. **Learning extraction rules**

   Shallow linguistic analysis only gives 'low level' information and writing extraction rules manually with this information is quite a complicated job. Additionally, the written rules often lack readability and are hard to maintain. Therefore, we applied a machine learning method, support vector machines (SVM), to learn some extraction rules. SVM has shown a high performance in many pattern recognition and natural language processing tasks.

2. **Detecting apposition**

   To answer a certain kind of question, detecting an appositive relation is very important. As we looked further into this issue, however, we found that such detection is not easy because apposition is often determined by long-range constraints in sentences and cannot be identified only by neighborhood information. We therefore created a simple but effective heuristics to detect appositive relations.

This paper is organized as follows. In Section 2, we briefly describe the overall structure of our QA system SAIQA-e (System for Advanced Interactive Question Answering in English). Then, we

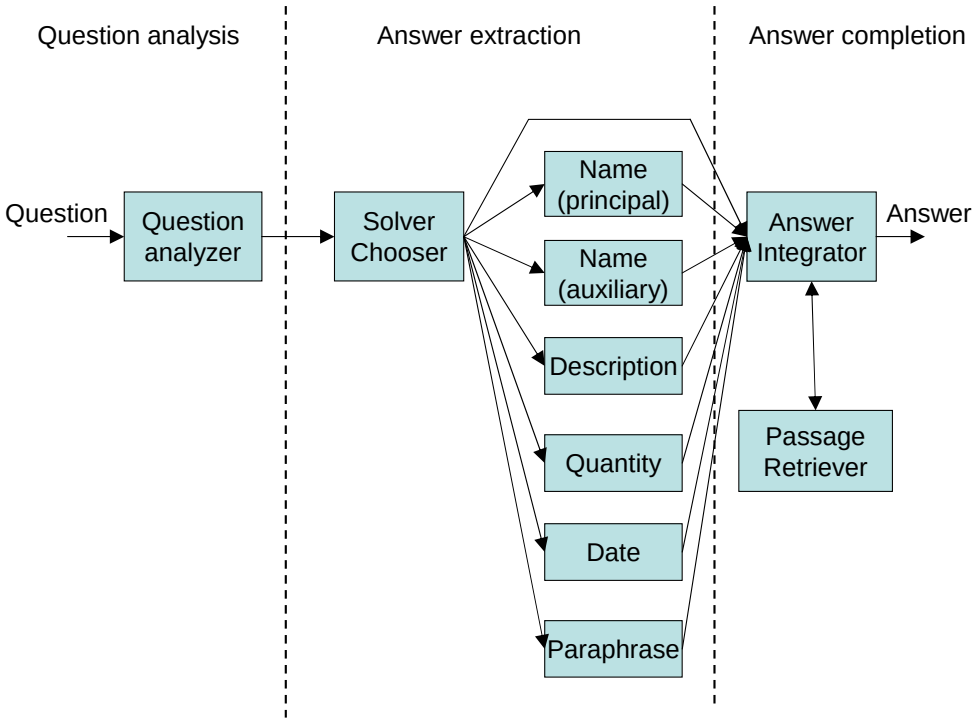

Figure 1: SAIQA-e overview

explain the approaches to our main problems, i.e., learning extraction rules and detecting apposition, in detail in Sections 3 and 4. Finally, in Section 5, we analyze results of SAIQA-e on the main task of TREC-10's QA track.

2 System overview

SAIQA-e first classifies a question into several categories (Figure 1). Then, it passes each question to an answer extraction subsystem, one that is specific to the question's category. Each extraction subsystem extracts candidate answers from a document set and scores them by heuristic measures. Finally, these candidate answers are merged when multiple candidates are located within a 50-byte length. If a question is categorized as 'unknown' or a specific solver does not extract five candidates, the system evokes a passage retrieval system, which extracts 50-byte passages. These passages are added to the candidates.

In the following, we describe the details of each component.

2.1 Question analysis

In the question analysis stage, each question is classified into one of the categories shown in Figure 2. The category is determined by a manually created decision tree and the following features of the question: question words (who, what, when,...), positions of the question words (start/middle/end of the question), first verb, head word of the first NP, head word of the second NP, and the word between the first and the second NPs.

Here, we explain the question categories.

- **Name**

Question type		Example
Name	Person	Who discovered x-rays? (Q945)
	Location	Where is John Wayne airport? (Q922)
	Organization	What is the name of the chocolate company in San Francisco? (Q943)
	Others	What is the Ohio state bird? (Q1001)
Description	Person	Who was Galileo? (Q896)
	Others	What is an atom? (Q897)
Date		When was Lyndon B. Johnson born? (Q1060)
Quantity		What are the speed hummingbirds fly? (Q953)
Paraphrase	Abbreviation	What is the abbreviation for Texas? (Q1172)
	Full name	What does I.V. stand for? (Q1176)
	Nickname	What is Shakespeare's nickname? (Q1294)
Unknown		What 's the easiest way to remove wallpaper? (Q1386)

Figure 2: Question categories

This category has four subcategories: Person (including entities treated like person, such as gods and comic characters), Location, Organization, and Others (including class names like animal species).

- **Description** This category has two subcategories, Person and Others.

  The distinction between Name-Person and Description-Person might be a little confusing, so let us present examples. "Who is the U.S. president?" is a Name-Person question because it asks about the name of a person who is the U.S. president. On the other hand, "Who is George W. Bush?" is a Description-Person question because it requires descriptive information about a person whose name is George W. Bush.

- **Quantity** This category has nine subcategories: Count, Age, Duration, Length, Money, Ratio, Size, Speed, and Others. As you can see, this category is rather broad and contains few related concepts. However, the expressions of these concepts are usually associated with numerical words and accordingly their extraction steps are expected to be similar. Based on this, we grouped these subcategories into one Quantity category.

- **Date** This category has four subcategories: Day, Day of the week, Year, and Others.

- **Paraphrase** This category has three subcategories: Abbreviation, Fullname, and Nickname. The category is created because these expressions are often related to the original expressions in unique ways (for example, an abbreviation follows an original expression in parentheses) and can be identified in a unified fashion.

The questions that are not classified into any above categories are labelled as 'Unknown' questions.

2.2 Answer extraction and evaluation

After the question is categorized, the answers are extracted from a document set and evaluated by heuristic measures. We have several extraction subsystems, each of which is intended to deal with

only restricted types of questions. Each question is passed to the corresponding subsystem, while 'Unknown' questions skip this extraction step and are passed directly to the next answer integration step. Here, we describe the subsystems.

Principal Name Solver

This subsystem deals with Name-Person, Name-Location, and Name-Organization questions. It was separated from the other Name solver because detecting names of person/location/organization in documents is harder than other name detection and we wanted to focus our resources on this problem.

The subsystem first retrieves articles ranked by the fequency of keywords and their proximity. Then, an SVM-trained named entity detection module extracts person/location/organization names from these articles. These names are evaluated by heuristic measures such as the proximity to the keywords.

The issue of SVM learning of named entity detection is discussed in Section 3.

Auxiliary Name Solver

This subsystem deals with Name-Others questions. The extraction and evaluation are similar to Principal Name Solver's, but the name detection rules are manually created. Additionally, the evaluation heuristics are less accurate because Name-Others questions cover such diverse kinds of entities that it is hard to develop accurate category-specific measures such as those used in Principal Name Solver.

Description Solver

This subsystem accepts Description-Person and Description-Others questions. The extraction and evaluation are quite different from the name solvers'.

The subsytem first retrieves all articles including the name of the requested entity. (It is easy to identify the name in the question.) Then, the NPs appositively connected to the name are extracted as the descriptive answers. The answers with the same head are grouped as variant expressions of the same description. Finally, the most specific expressions of the groups are scored by the number of group members. (That is, a more frequent description is considered to be more trustable.)

Apposition detection plays the main role in Description Solver. We discuss this in Section 4.

Quantity/Date Solver

These subsystems deal with Quantity and Date questions. They are almost the same as Auxiliary Name Solver and the differences are in the extraction rules.

Paraphrase Solver

This solver deals with Paraphrase questions and the subsystem is quite different from other solvers.

For example, for Paraphrase-Abbreviation questions (for example, "What is the abbreviation for the United Nations"), it retrieves all articles in which the fullname (United Nations) appears. Then, a regular expression is used to extract all abbreviations from the articles. Finally, a sequence of upper characters in the fullname (UN) is compared to a sequence of upper characters in the abbreviations. This comparison is done approximately so that some missing characters are tolerated and the matching degree is translated into the score of the abbreviation.

2.3 Answer integration

After the answer extraction and evaluation stage, the answers are extended to 50 bytes and merged when the 50-byte passages contain multiple answers. Then, we add 'no answer' in the following manner.

1. If a question is classified into a category other than 'unknown' and the specific solver does not return as many as five answers, then 'no answer' (i.e., 'NIL') is added to the candidates. After that, the output of the passage retrieval system is added.

2. If a question is classified into the 'unknown' category and the passage retrieval system does not return as many as five answers, then 'no answer' (i.e., 'NIL') is added to the candidates.

We set all 'final answers' as '1', because SAIQA-e's outputs have already been sorted according to their relevance and we consider the first-ranked answer as the most trustable one.

3 Named Entity Recognition based on Support Vector Machines

Named entity (NE) recognition systems are useful for determining whether a certain pronoun designates a person or an organization or location. Although we have had our own Japanese NE systems, we did not have any experience on developing English NE systems. Therefore, we decided to develop one by using a corpus-based approach. Since we did not have any training data for English NE tasks, we prepared our own training data.

We employed support vector machines (SVM) for the English NE system. Such a system was proposed by Yamada et al. [YKM01] for Japanese NE recognition. His system is a simple application of Kudo's chunking system [KM01] that shows the best performance for the CoNLL-2000 shared task. We also implemented an SVM-based NE system for Japanese. This SVM-based NE system employs a different approach, but according to our experiments, this system is better than the other Japanese NE systems we have (a C4.5-based rule generation system [Iso01] and a system based on maximum entropy (ME) modelling). In the following sections, we describe our English NE systems.

3.1 Support Vector Machines

First, we introduce SVM briefly. The non-linear SVM classifier [SBS99] uses a decision function for an input vector $\vec{x}$ given by

$$f(\vec{x}) = \mathrm{sign}(g(\vec{x}))$$

where $\mathrm{sign}(y) = -1$ for $y < 0$ and $\mathrm{sign}(y) = 1$ for $y > 0$, and

$$g(\vec{x}) = \sum_{i=1}^{\ell} w_i k(\vec{x}, \vec{z}_i) + b.$$

$k(\vec{x}, \vec{z})$ is called a kernel function. Several kernel functions are known. By considering Japanese NE results, we decided to use a second-order polynomial kernel function $k(\vec{x}, \vec{z}) = (1 + \vec{x} \cdot \vec{z})^2$. The $\vec{z}_i$s are called support vectors that are representatives of training examples. w_is and b are constants determined by the training examples.

3.2 The first NE system

The first English NE system we implemented was a simple variation of ME-based NE systems proposed by Borthwick [Bor99] and Uchimoto [UMM+00]. In this sytem, each word is classified into 21 classes: {PERSON,ORGANIZATION,LOCATION,FACILITY,ARTIFACT} × {SINGLE,BEGIN,MIDDLE,END}

$\cup$ {OTHER}. Here, (PERSON,SINGLE) is a label for a one-word person name like "George." (PERSON,BEGIN) is the first word of a certain multi-word expression for a person's name (e.g., "George" in "George Bush"). (PERSON,MIDDLE) indicates an internal word (e.g., "Walker" in "George Walker Bush"). (PERSON,END) is the last word (e.g., "Bush" in "George Bush"). When a word does not belong to any of the named entities defined above, it is labeled as OTHER.

In ME-based NE systems, the Viterbi algorithm is employed to get the best combination of labels. Since the ME model gives conditional probabilities, this is easy.

However, SVM does not tell us such probabilities. In addition, ordinary SVM can only solve two-class problems. Therefore, we built 21 SVM classifiers, i.e., one SVM for each class. For the application of the Viterbi algorithm, we used the sum of $g(\vec{x})$ instead of the sum of logarithms of probabilities. We used Kudo's TinySVM because of its faster speed over the well-known SVM light [SBS99] for this kind of task.

Since this first NE system classifies every word in a given document, the training data for each class has 10^5-10^6 examples. As a result, its training took a very long time.

In our case, we applied the NE system to the TREC data after the training. It turned out that it was also too slow in the application phase. Because of this slowness, we could not try various combinations of possible features. In addition, we could not improve the QA system, which depends on the NE system. Therefore, we abandoned the first NE system.

3.3 The second NE system

We implemented another NE system in which hand-crafted rules were designed to detect NE candidates (roughly, noun phrases containing capitalized words) and then SVMs classified them into four classes: $C = \{\text{PERSON}, \text{ORGANIZATION}, \text{LOCATION}, \text{OTHER}\}$. For efficiency, we removed two classes, i.e., FACILITY and ARTIFACT, because they had only small numbers of positive training examples and their results were not very good.

In the second NE system, the features for classification include word strings, their memberships in word lists, their part-of-speech tags, word counts, neighbor words, appositive information, information about preceding occurrences in the same documents, and other surface features usually used in other NE systems. Since SVM allows only numerical values in the input, we have to convert features into a set of numerical vector components.

One example is represented by one numerical vector. Suppose an NE candidate's head word is *Washington*. Then, we introduce an axis for the feature *head_word_is_Washington*, and its value is 1. At the same time, the vector's incompatible axes like *head_word_is_University* have 0 as their values. In TinySVM, we have only to enumerate non-zero components.

For each candidate, the outputs of four functions, g_{PERSON}, $g_{\text{ORGANIZATION}}$, g_{LOCATION}, g_{OTHER}, are compared and the function that gives the largest value is chosen as class ($\text{argmax}_{c \in C} g_c(\vec{x})$).

The second NE system was found to be much faster than the first NE system, but it was still too slow for application to all TREC documents. Instead, we embedded the second NE system into the QA system and to be called on demand.

4 Description solver and Apposition detection

To determine which parts of documents contain descriptions of entities is difficult even for humans, but we provisionally adopted the following assumption.

- The description of an entity is expressed as the appositive modifier of the entity.

For example, in the sentence "George W. Bush, the U.S. president, said...", 'George W. Bush' is appositively modified by 'the U.S. president'. Therefore, 'the U.S. president' should be some description of 'George W. Bush'. This assumption makes the detection of an appositive relation the principal task in answering a description question.

Q. category	Num. of Q.	MPR (strict)
Name-{Per,Loc,Org}	131	0.349
Name-Others	76	0.096
Desc-{Per,Others}	128	0.247
Quantity	68	0.219
Date	48	0.119
Paraphrase	14	0.193
Others	27	0.151
Total	492	0.228

Table 1: Results of TREC10 QA track (main task)

In detecting appositive relations, punctuation disambiguation plays an important role. By 'punctuation disambiguation', we mean distinguishing the syntactic roles of commas. For example, in the sentence "When I was a kid, things were simple.", the comma is used as a marker of syntactic movement. On the contrary, in the sentence "George, the son of the former president, is a popular man.", the comma shows an appositive relation between 'George' and 'the son of the former president'. Note that in both examples, the commas are placed between noun phrases. This indicates that we cannot disambiguate this kind of comma usage only from neighbor information and punctuation disambiguation requires 'long-range' information.

We first used some off-the-shelf parsers to detect apposition. Unfortunately, we found that these parsers often failed around commas. We then created several heuristics to disambiguate punctuations and then to identify appositive relations. These heuristics classify punctuations into appositive markers, movement markers, and coordination markers (such as in "cats, dogs and birds").

Here are examples of the heuristics.

1. If a sentence starts with a subordinating conjunction, the leftmost comma in the sentence is a movement marker. (For example, "When I was a kid, TV was not popular.")

2. If a sentence contains the sequence of '(NP ,)+ NP CC NP', these commas are coordination markers.

5 Main task results

Table 1 shows the evaluation returned by NIST for each question category. These categorizations were manually done after the result was submitted.

Name-Person/Location/Organization result in the highest score (0.349) among all categories. This provides moderate but convincing evidence that our machine learning approach in NE recognition improves the answer extraction accuracy. The second highest is Description. Actually, this result was a little surprising for us because the extraction of the description was based on only a simple assumption (See Section 4).

References

[Bor99] Andrew Borthwick. *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York University, 1999.

[Iso01] Hideki Isozaki. Japanese named entity recognition based on a simple rule generator and decision tree learning. In *Proceedings of Association for Computational Linguistics*, pp. 306–313, 2001.

[KM01] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings of NAACL*, 2001.

[SBS99] Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors. *Advances in Kernel Methods*. MIT Press, 1999.

[UMM+00] Kiyotaka Uchimoto, Qing Ma, Masaki Murata, Hiromi Ozaku, Masao Utiyama, and Hitoshi Isahara. Named entity extraction based on a maximum entropy model and transformation rules (in Japanese). *Journal of Natural Language Processing*, Vol. 7, No. 2, pp. 63–90, 2000.

[YKM01] Hiroyasu Yamada, Taku Kudo, and Yuji Matsumoto. Japanese named entity extraction using support vector machines (in Japanese). In *IPSJ SIG Notes*, 2001. NL-142-17.