Abstract

This paper describes the framework for a new abstraction method that utilizes event-units written in sentences. Event-units are expressed in Language Information Structure (LIS) form and the projection of LIS from a sentence is performed by a semantic engine. ABEX (ABstraction EXtraction system) utilizes the LIS output of the semantic engine. ABEX can extract events from sentences and classify them. Since ABEX and the LIS form use only limited knowledge, the system need not construct or maintain a large amount of knowledge.

1 Introduction

Automatic abstraction, a major natural language processing application, is difficult to achieve. Luhn[5] developed a very simple extraction method that searches for the keywords in each sentence. This type of abstraction is easy to accomplish, but its quality is poor. Other abstraction methods[6, 3] utilize natural language understanding (NLU). However NLU is still in the development stage. For achieving good practical performance, it is necessary to treat the information expressed in a document uniformly so that it can be analyzed with only a small fixed amount of knowledge.

We propose the LIS form which allows information about events to be uniformly treated. Furthermore, the semantic engine only uses abstract words, this reduces the size of the knowledge. So the semantic engine projects a sentence to a LIS form within a limited amount of knowledge.

In abstraction, classification is the first step. Classification is performed using uniform approach, the LIS form. LIS event representation allow us to select and classify sentences.
we introduce extenisonal roles which allow to be modified by the addition of "constraint".

2.2.2 Word feature-structure

Word has six features. These are semantic feature (DDF) slot, numerical-value slot, date slot, constraint slot, modality slot, and word string slot.

Using the semantic feature, the event feature-structure will be determined during semantic interpretation process. Six classes of semantic features are defined, such as INDIVIDUAL, ELEMENT, THING, ACTION, LOCATION, and TIME. These classes are instantiated to Domain Dependent semantic Features (DDF) when the domain is decided.

The constraint feature restricts the feature-structure of brother words or phrases. Furthermore, the constraint feature determines the relation between word feature-structure and event feature-structure. In Japanese language, a word which has a ACTION DDF usually has the constraint feature that determines the slots of event feature-structure.

The numerical-value slot expresses numerical value of a word; 0, 1, 2, ~ (one), 3 (hundred), 4 (thousand), and so on. The calculation of counting up and down is necessary, so all figures are separated. The numerical-value feature will be expressed as follows. (Our notation of a feature-structure is [feature-name = feature-value].)

\[
\text{numerical-value} = \begin{bmatrix}
\text{first-digit} &=& \ldots \\
\text{second-digit} &=& \ldots \\
\ldots &=& \ldots
\end{bmatrix}
\]

The date slot expresses event occurrence time and is expressed by the Christian era. In the Christian era, days are counted by numbers, so that date slots are calculated using the numerical-value feature. The date slot has a minute slot, a second slot, a hour slot, a day slot, a month slot, and a year slot. Each slot is expressed in numerical value.

The modality slot is classified into three categories; tense, aspect, and mood. Since the tense and aspect are linguistically fixed, we use an ordinary categorization. However, mood is needed to be categorized differently, because the information unit used this system is an event. So we categorized mood as a combination of Brittan's Belief-Desire-Intention model[1] and modal logic. That is, the state of event is expressed by modal logic (necessary operator, possible operator, and negation sign) and the attitude of speakers can be classified into belief, desire, and intention. As an example, the sentence I think it is possible to construct a plan there will be expressed as \text{Belief[Possible E]}, where \text{E} means an event; \text{construct a plan there}, that is, the individual believes that \text{E} is possible.

Furthermore, it is necessary to consider a situation in which information is transferred. In the newspaper, it is created by journalists who get information from other services (person or company information bureau). In this situation, the event and the attitude of the information possessor (IP) is transported to a speaker (SP); journalist.

The journalist then reexpresses the information to reflect his attitude. If the modality of IP and SP are expressed as \text{Modality}_{\text{IP}}(), \text{Modality}_{\text{SP}}(), respectively, information in newspaper is expressed as, \text{Modality}_{\text{IP}}(\text{Modality}_{\text{SP}}(\text{EVENT})).

If the target document is newspaper, the LIS form includes the modality of speaker (\text{Modality}_{\text{IP}}) and the modality of information possessor (\text{Modality}_{\text{IP}}).

2.3 Projection Mechanism

Parsing is done using Morphological analysis and Dependency analysis[4] and yields a syntactic tree for a sentence.

After the parsing, we search a feature-structure dictionary to extract feature-structures of all words related to the domain. To perform semantic analysis with limited knowledge, word feature-structures are prepared only for abstract words. The registration of proper nouns are left to the user. The semantic engine infers the semantic meaning of words or phrases from the system default and user registrations held in the dictionary. This means the semantic engine do not need all knowledge of words for semantic interpretation. Thus only a small amount of words need to be maintained.

After attaching the appropriate word feature-structure to all important words, semantic interpretation can proceed. From the type of propagation for the feature-structure in a parsing tree, there are two types of features. One is the synthesized type whose value is calculated from sons to father relationship of the parsing tree. The other is the inherited type that are calculated from father or brothers.

For DDF features, the grammar is as follows.

\[
E.\text{DDF} ::= \text{indefinite} \\
N.\text{DDF} ::= N_1.\text{DDF} \mathbin{\@} N_2.\text{DDF} \mathbin{\@} \cdots \mathbin{\@} N_n.\text{DDF} \\
N.\text{DDF} ::= \text{individual} \text{[company]} \mathbin{\@} \text{element} \text{[company]} \mathbin{\@} \text{money} \text{[man]} \mathbin{\@} \text{product} \mathbin{\@} \text{action} \text{[company]} \mathbin{\@} \text{location} \text{[time]}
\]

Note: Uncapitalized words mean terminate and capitalized words mean nonterminate. E means EVENT node structure and N means other node structure and N.DDF means DDF feature-value in node 'N'. Symbol 'n' is the number of nodes. Operator @ means unification operator.

For the constraint feature,

\[
E.\text{constraint} ::= \sum_{i=1, \ldots, n} (N_i.\text{DDF} \mathbin{\@} E) \\
N.\text{constraint} ::= \sum_{j=1, \ldots, n} (N_j.\text{DDF} \mathbin{\@} N) \\
N.\text{constraint} ::= \text{feature-structure of brother nodes}
\]
The date and numerical-value features are rather complicated because we have to deal with the semantic meaning of time.

The grammar of the date feature is,

\[
N._{\text{date}} ::= N._{\text{date}} \oplus N._{\text{date}} \oplus \cdots \oplus N._{\text{date}}
\]

\[
N._{\text{date}} ::= \begin{cases} \text{minute} \equiv \cdots \quad & \text{second} \equiv \cdots \quad & \text{hour} \equiv \cdots \quad & \text{day} \equiv \cdots \quad & \text{month} \equiv \cdots \quad & \text{year} \equiv \cdots \end{cases}
\]

The calculation of number and date features is done like a stack. The numerical-value feature has one stack and date feature has six stacks.

For example, for the number 1992, all the numbers, 1, 2, and 9, are expressed as follows,

\[
\begin{align*}
1 & \Rightarrow \text{[numerical-value} = \text{eval (push-number-stack 1)}] \\
2 & \Rightarrow \text{[numerical-value} = \text{eval (push-number-stack 2)}] \\
9 & \Rightarrow \text{[numerical-value} = \text{eval (push-number-stack 9)}]
\end{align*}
\]

Note: Symbol 'eval' means that next form will be evaluated by Common-lisp. Symbol 'push-stack' is the function that puts the argument value on the top of the stack.

The equation for the numerical-value of 1992 is,

\[
1992._{\text{numeral-value}} = \{ \text{[numerical-value} = \text{eval (push-number-stack 1)}], \text{[numerical-value} = \text{eval (push-number-stack 9)}], \text{[numerical-value} = \text{eval (push-number-stack 2)}] \}
\]

which, after evaluation, gives as the value of 1992 as following expressions, counting right to left, first digit being 2, second digit begin 9, and so on.

\[
1992._{\text{numeral-value}} = \begin{cases} \text{first-digit} \equiv 2 \\
\text{second-digit} \equiv 9 \\
\text{third-digit} \equiv 9 \\
\text{fourth-digit} \equiv 1 \end{cases}
\]

If we process the phrase, 1992 \textit{(year 1992)}, the equation becomes,

\[
\text{1992.year} = \{ \text{[year} = \text{eval (if SELF.numerical-value (push-year-stack SELF.numerical-value))]} \}
\]

Note: Nonterminate 'SELF' refers to self feature-structure. Symbol 'push-year-stack' is the function that places the argument value to the year stack in the date feature.

Then and we get the time feature-structure as,

\[
\begin{align*}
[1992 \text{ year} = 1992]
\end{align*}
\]

The grammar for modality is quite simple,

\[
E._{\text{modality}} ::= N._{\text{modality}} \oplus N._{\text{modality}} \oplus \cdots \oplus N._{\text{modality}}
\]

\[
N._{\text{modality}} ::= N._{\text{modality}} \oplus N._{\text{modality}} \oplus \cdots \oplus N._{\text{modality}}
\]

\[
N._{\text{modality}} ::= \text{trust, aspect, and mood}
\]

2.4 An example of the projection process in the semantic engine

This passage comes from The Nikkan-kougyou shinbun (Daily Industrial Newspaper). The headline is “Takada Kiko Co., Ltd. is constructing a new plant to assemble large-scale steel bridges in Wakayama-ken.”

S1: "Takada Kiko announced a land purchase agreement in Shimotsu-machi, Wakayama-ken, where they will construct a new plant giving them additional space and capabilities to fabricate larger scale steel bridge structures."

S2: "Construction of the new plant is expected to start in April, 1993.

S3: "Investment capital is about ¥ 22 billion.

S4: "Operations will begin in April, 1993.

(S1) is made of two events. One is "construction" and the other is "agreement". The two events are connected by "and" (surukoto-de). So sentence (S1) is separated to two events. Sentence S3 does not have an ACTION DDF, so further analysis is not conducted.

Therefore, we can obtain five events from five sentences: S1 to S5.
a Maruzen Oil Co., Ltd. refinery, was purchased from Maruzen Shimotsu Kosan for an estimated ¥10 billion.

Event4 in S2: "来看から新工場建設に着手する。" (Construction on the new plant facility is slated to begin this coming spring.)

Event4 in S4: "平成五年四月桝業の予定。" (Operations will begin in April, 1993.)

After the event separation process concludes, semantic interpretation is commenced. The first stage is attaching a feature structure to each word.

Let's consider the Event4 in S2; "来看から新工場建設に着手する。" (Construction on the new plant facility is slated to begin this coming spring.), there are three Bunsetsu, five independent words, three dependent words. We need only five feature-structures as shown below.

来自;this coming spring—

\[
\begin{array}{l}
\text{string} = \text{来自 (raisyun)}; \text{this coming spring} \\
\text{date} = \text{(push-year-stack (1 + *article-year*)})
\end{array}
\]

工場;plant—

\[
\begin{array}{l}
\text{string} = \text{工場 (kouyou)}; \text{plant} \\
\text{DDF} = \text{element (company)}
\end{array}
\]

建設;construction—

\[
\begin{array}{l}
\text{string} = \text{建設 (kensetsu)}; \text{construction} \\
\text{DDF} = \text{action (company)} \\
\text{constraint} = \begin{cases}
\text{agent} = \text{individual (company)} \\
\text{action} = \text{action (company)} \\
\text{time} = \text{time}
\end{cases}
\end{array}
\]

着手;is slated to.......

\[
\begin{array}{l}
\text{string} = \text{着手 (chakkou)}; \text{is slated to} \\
\text{modality} = \text{aspect = just, before}
\end{array}
\]

する;will—

\[
\begin{array}{l}
\text{string} = \text{する (suru)}; \text{will} \\
\text{modality} = \text{[aspect = future]}
\end{array}
\]

Note: Symbol "agent.DDF" means that if the DDF value is unified to the one node, then variable "agent" is bounded to that node's feature-structure. Variable *article-year* is bounded to the date of year when the article is published.

The example is parsed as shown in figure 1.

Once the parsing is finished, the semantic interpretation process begins. Node n1 will have the feature-structure that is the result of calculation between the feature-structure of "来自 (raisyun)" and "から (kara)", but the word "から (kara)" has no feature-structure so the feature-structure of "来自 (raisyun)" is propagated to node n1. The feature-structures of all nodes are calculated same way.

For the constraint feature, unification was done to all brothers. For example, [agent.DDF = individual (company)] means that one brother node is needed which have the DDF value of agent (company). If there is a node which satisfies the constraint, then the variable _agent_ is bounded to that node feature-structure. If there is no node which satisfies the constraint, then variable _agent_ is unbounded.

Try to think about the constraint feature in "建設 (kensetsu)". There is no node that has agent (company) in DDF, but there are nodes which satisfy the constraint, such as the action, object, and time which are bounded to nodes n3, n2, n1, respectively.

Finally we get the event feature-structure of top node n-top, shown in figure 2.

3 An abstraction using the LIS form

3.1 The basic method of the abstraction

In the abstraction, we utilize classification of the LIS output. First, a sentence is put into the LIS form by the semantic engine.

The LIS output is used to commence the abstraction procedure. To extract information from sentence, we think classification is the best way. The semantic engine analyzes sentence in fixed domain, after the semantic analysis, Sentences are classified whether an event or not, and the system extracts the events which are related to the domain.

Finally, ABEX provides a abstraction. One abstraction proposed here is the classification of event occurrence time and similarity of event. This classification reveals the relationships of each event. Individual event occurrence times will be determined from value of the time feature and the similarity of events is calculated by comparing event feature-structure slots.

The other method is classification by the modality of information. From the view point of Modality, we can classify an event according to the modality of information possessor (IP). If there is no modality in the event, we classify it as 'fact'. Others are classified using modality feature. This classification of the event's modality reveals the attitude of the information possessor.
3.2 An example of the abstraction

Figure 2 shows a typical abstraction result of ABEX. The events are classified by the event occurrence time and the similarity of each event. In this figure, x-axis indicates absolute event occurrence time and y-axis indicates relative similarity of events and circled icons indicate single events.

A typical classification result using the modality of information is shown in Figure 3.

The Event 2 has the modality of an official bulletin and Event 5 has the modality of company intention, so we get the abstraction result shown in Figure 3.

4 Conclusion

We have described a framework for a new abstraction method that utilizes classification. Classification is performed using the output of a semantic engine that is based on LIS form. Since the LIS form takes into account the incompleteness of knowledge, the system requires only a small amount of knowledge to perform the semantic analysis.

First, ABEX utilizes the selectivity of the semantic engine according to the domain and the event. Furthermore, ABEX classify according to the LIS constituents such as, TIME modality and so on. The generation mechanism is poor, but abstraction by classification is an easy way making an abstract. Furthermore, the classification method described here well supports human abstract tasks.

References

[1] M. E. Bratman. Plans and resource-bounded practical reasoning. Computational Intelligence, (4), 1988.
[2] C. J. Fillmore. Toward a modern theory of case. Prentice-Hall, 1969.
[3] U. Hahn. Topic essentials. Coling-86, 1986.
[4] H. Inagaki, S. Miyahara, and F. Obashi. Sentence disambiguation by document oriented preference sets. Coling-90, 1990.
[5] H. P. Luhn. The automatic creation of literature abstract. IBM Journal, Vol.2, 1958.
[6] L. F. Rau. Information extraction and text summarization using linguistic knowledge acquisition. Processing & Management, pages 419-428, 1989.