5E-1

The Automatic Sentence Separation for Thai Language

Vises VORASUCHA, Surapant MEKNAVIN, Hozumi TANAKA
Tokyo Institute of Technology

1 Introduction

Unlike any other languages, Thai language use no punctuation. There is no period and comma appear in Thai text. That is Thai text is written exactly in the way it is spoken. The only punctuation appears in Thai text is space. Moreover, Thai language is in the form of SVO, and use no inflection at all. Mixing up with preposition phrases which can be put at the beginning or ending of a sentence, it is difficult to determine where the beginning or ending of a sentence is. It is true that even though Thai people, in some cases, can hardly say where is the beginning of a sentence. To perform an analysis of Thai language on a computer properly, the automatic sentence separation is indispensable.

This research tried to find any algorithm to do the automatic sentence separation for Thai language.

2 Some Observations on Thai Sentence Structure

If we look carefully at Thai text, we can observe some important points which are conditions for the sentence separation. We can roughly summarize these observations as follows.

1. Sentences in a paragraph do not spread over to next paragraph.
2. Sentences must end at a space. (But spaces may be included in a sentence) Space is the only punctuation normally included in Thai text. One thing to be mentioned here is Thai language, like Japanese but different from English, use no space between words. Space is to be used occasionally in a paragraph. Where to put a space depend on the writer sense. There is no clear rule of where to put space into Thai text.
3. Sentences do not end after words like "for example" or "such as", even though there is a space after such words.
4. Usually sentences do not begin with conjunction or verb phrase or preposition phrase.
5. Usually sentences begin with noun phrase

3 Some Significant Features of Sentence Separation Program

We wrote a program for sentence separation. In this section we would like to mention about some important features which are the background of this program.

According to the first observation in the previous section, the program first reads in all of the text in one paragraph and keeps it in a buffer. Next, according to the second observation, we pay the attention to each of spaces appears in that paragraph. Here, we call text exists from a space to next space "Block". We can decide the separation of a sentence by considering the relation between the block before a space and the block following that space. Namely, if there is any relation between two blocks of both sides of a space, then we ignore the space and consider these two blocks to be one block. Else, the sentence ends at that space. Therefore, all of what we have to do is to find out whether any two blocks have any relations to each other or not.

One important point is that we use the method of longest matching. That is the block will be connected to the prior block as long as there are any possibilities to be connected. Once the program get a sentence separated, it will send the sentence to the syntax and semantic analysis program. According to the syntax and semantic analysis, if the sentence is acceptable then the program will begin to separate another new sentence start from where the previous sentence is separated. But if the sentence is not accepted by the analysis program, by the backtrack function of Prolog, the separation program will try to find other possibility to separate the sentence, and make the sentence shorter. The sentence will be cut shorter and shorter until there is any possibility which is accepted by the analysis program.

Another point need to be mentioned is, in our system, we do the separation absolutely before the syntax and semantic analysis. We use the information of the surface structure as much as possible, and based on this information we try to find out at least one possibility of the separation. Information of syntax and semantic copes with the separation program only when the analysis fails and backtracks to the separation program.

4 Types of Block

As mentioned above, "block" is the text that is exist between two spaces. We can divide the type of block into 4 types, according to its function on the sentence separation. In this section we are going to explain about these kinds of block.

1. Forward Block: This is the block that can not stand alone as a sentence. Space which follows this kind of block is ignored, and the block itself is connected to the following block. That is this block have to stand "forward" to other blocks. This kind of block is the block that end with words such as
   - "is", "that", ... (คือ , ว่า)
   - "for example", "such as", ... (ได้แก่ , เช่น)
2. Backward Block: This is also the block that can not stand alone as a sentence. Space before this kind of block is ignored, and the block itself is connected to the previous block. That is this block stands "backward" to other blocks. This kind of block will begin the block with words such as
   - conjunction (exclude "because", "for")
     (เพราะ , เพื่อ)
   - verb (exclude "there is", "happen" which can come at the beginning of a sentence)
     (เกิด , มี)
   - "etc." (เป็นต้น)
3. Special Block: This is the block that are special and cannot include into the above two types. We can recognize one type of this blocks.
   - Block beginning with "because", "for" : For blocks which begin with this kind of conjunction, that is the idiom conjunction that come together in pair, for instance the "not only ... but also ..." in English, we cannot just simply attach these blocks to previous blocks. We have to consider the following block as well. This kind of block may be attached to both previous or following block depends on conditions. Consider the following example which has the

The Automatic Sentence Separation for Thai Language
Vises VORASUCHA, Surapan MEKANAVIN,
Hozumi TANAKA
Tokyo Institute of Technology

meaning of "I cannot buy a car, because I do not have money."

... ฉันไม่ซื้อรถ เพราะฉันไม่มีเงิน ...

```
... I cannot buy a car
    because I do not have money ...
```

This is the case that the "because" block connect to the previous block. The following example is the other case.

... เพราะฉันไม่มีเงิน ฉันจึงไม่ซื้อรถ ...

```
... because I do not have money
    therefore I cannot buy a car ...
```

Therefore, for this kind of block, we have to consider not only the current block but also both previous block and following block of the current block at the same time to determine the separation.

4. Sentence Block: This is the block other than those three types above that can stand alone as a sentence, and need not to connect with other blocks.

Divide the types of block in this way, we can do the separation of Thai sentence easily by considering the relation of each block to the other blocks.

5 Semantically Parallel: The Scope of "example" and "restate"

Even though we divide the block into several types, there are still some problems left. One of them is the case of blocks that are semantically parallel. This is the case of "example" and "restate".

When we say "for example" usually there are some items follow. In our program they are a kind of block, too. All of them are, in some sense, semantically parallel. Unlike other kinds of blocks, we have to find some relations among them and put them together into the same sentence, and to separate the sentence at the appropriate place when the new sentence begin. That is what we call the scope of "example".

The problem is how we can determine the scope of "example". Actually, for us human we can know the scope easily by the meaning of the words and phrases. But to process this on a computer, we need something more simple for computer to understand. After we studied an amount of Thai sentences, we divided the type of "example" scope to be five types as follow only according to syntactical information.

1. "for example" N1 ... Nn S1 ...
2. "for example" N1 ... Nn V1 ... Vm S1 ...
3. "for example" V1 ... Vm S1 ...
4. "for example" S1 S2 ...
5. "for example" ... "etc." ...

For the first three cases, we separate the sentence at the beginning of the first sentence (S1). For the forth case, we separate at the beginning of second sentence (S2). The last case is the case that there is word "etc." included in the text, for this case we separate just after the word "etc." If there are many cases occur at the same time, the program will separate a sentence at the first nearest place which is possible to be separated.

The "restate" case is another case of the semantically parallel block. Consider the following example.

... อริสโตเติ้ล นักวิทยาศาสตร์ชาวกรีก ...

```
... Aristotle the Greece scientist ...
```

In this example, the phrase "the Greece scientist" is a restate phrase of "Aristotle". We consider the case like this to be another semantically parallel, too.

We still do not have any good solution for this case. To separate sentence like this it needs semantic information of the blocks to check whether they are parallel or not. But, to check whether two noun phrase is parallel or not need an extremely high level of semantic analysis. We will left this to be one of our further topics.

6 Comments on Implementation

Implementation is done on LangLAB[1] system, the natural language analysis system in our laboratory. The program is written in Prolog. Because our program works closely with the syntax and semantic analysis program, therefore it is hardly to say what is the percentage of correctness of our algorithm. The correctness of the separation is depend on the correctness of the syntax and semantic analysis.

7 Summaries and Further Topics

We tried to find a way to do automatic sentence separation, and check its practicality by implementing it on several topics still left to be further topics. In this section we are going to remark on them briefly.

7.1 The semantically parallel block

As mentioned in the previous section, the problem of semantically parallel block is difficult to solve. Usually, sentence will be separated between two noun phrases. But for two noun phrases which are semantically parallel, they must be put into the same sentence. It is difficult to determine that the two noun phrases have the semantically parallel relation or not. To do this it need an extremely high level of semantic analysis.

And, even though such a system exists there will be a problem of requirement of semantic analysis before being able to separate a sentence. We mentioned in the previous section that in our system for the sake of efficiency, we postpone the analysis of syntax and semantic to be done after we can separate a sentence. If it is really need to do the semantic analysis before we can separate a sentence, this will greatly lower the efficiency of the program.

7.2 Preposition Phrase

At this moment, a preposition phrase is attached to the previous block unconditionally. Since in Thai language preposition phrase can be placed at the beginning or ending of a sentence[2, 3], therefore when we only pay an attention to a preposition phrase, we cannot tell that the preposition phrase is in the previous sentence or in the next sentence.

In our system, a preposition is always attached to the previous sentence. After perform the semantic analysis, if it is semantically acceptable, the sentence will be separated. But if it is not, system will do the backtrack, and shorten the sentence. This mean that it will cut off the preposition phrase, and automatically put that preposition phrase to be attached to the next sentence.

8 Acknowledgement

Many thanks goes to Somkiat Tangkitvanich. He made many valuable comments on this research. The research will not be as successful as it is without his help.

References

[1] T. Tokunaga, M. Iwayama, T. Kamiwaki, and H. Tanaka. LangLAB : A natural language analysis system. In *COLING '88*, 1988.

[2] V. Vorasucha, and H. Tanaka. Thai Language Analysis Using Feature-Unification. In *The 5th Conference Proceedings of Japan Society for Software Science and Technology*, 1987.

[3] V. Vorasucha, and H. Tanaka. Thai Syntax Analysis Based on GPSG. In *Journal of Japanese Society for Artificial Intelligence*, 3(1):78–85, 1988