International Journal of Knowledge and Language Processing
Volume 6, Number 1, **2015**

www.ijklp.org
KLP International ©2015 ISSN 2191-2734
pp.22–30

The Study of the Automatic Identification of the Parallelism Sentences in *Mencius*

Liang Shehui
International College for Chinese Studies / International Chinese Language Education Institute
Nanjing Normal University
No.122, Ninghai Road, Nanjing, 210097
China
liangshehui@163.com

Received January 2015; revised March 2015

ABSTRACT. By taking the parallelism sentences in Mencius as instances, this article explored the automatic identification of the ancient Chinese figures of speech. This article analyzed the features of the parallelism sentences in detail, designed the corresponding automatic identification algorithm to deal with the parallelism sentences in Mencius, and applied the algorithm into The Analects of Confucius so as to compare the results from the automatic identification of both Mencius and The Analects of Confucius. When processing the two Pre-Qin documents, Mencius and The Analects of Confucius, which are different in styles, the algorithm of the parallelism sentences could always keep a good performance.

Keywords: *Mencius*, parallelism sentence, automatic identification

1. Introduction. *Mencius* is a political comment work, and one of its outstanding features is that the commentary words occupy a large space, in which the parallelism sentence is one of the most typical sentence patterns. In *Mencius*, a lot of important comments are expressed by the form of parallelism sentences.

Using parallelism to explain the reasons can make the comments well organized, full of strength and more convincing, which is conducive for Mencius to state his political views to king Hui of Liang in a more powerful way. There are a good many similar parallelism sentences expressing people's own political opinions in *Mencius*. Excerpting these parallelism sentences from *Mencius* can facilitate our comprehension and learning of the correlative content of this masterpiece in a comparatively new perspective.

Our research aims to use computer to elicit parallelism sentences in *Mencius*, which will not only help us construct a credible database of the parallelism sentences in *Mencius*, but also contribute some efficient methods to obtaining parallelism sentences automatically during processing other texts same or similar in genres, and provide reference to the subsequent automatic identifications of the features of speech.

2. **The Algorithm Design of the Automatic Identification of the Parallelism Sentences**
At present relative researches about automatic identification or acquisition of the parallelism sentences of *Mencius* or other Pre-Qin documents are rare, so are the researches in modern Chinese. Therefore, we considered to self-design an efficient algorithm to elicit the parallelism sentences in *Mencius*, and to apply it to other relative Pre-Qin documents so as to test its reliability and efficiency.

Mencius has about 35,000 Chinese characters, the scale of this number is too small to adopt the statistical methods. Thus, we would adopt rule-based approach to identify the parallelism sentences automatically.

2.1. The Classification of "Clause Segmentation". The boundary of a "clause" in parallelism sentences is not strict. The punctuation marks segmenting the clauses might be a comma, a semicolon or a period (here, including a question mark and an exclamatory mark). The length of the clauses segmented by the punctuation marks respectively varies from a comma to a semicolon, and then to a period in an incremental way, while the length between every two clauses is quite similar. Therefore, it would not deter us to use a unitary algorithm to search parallelism sentences segmented by different punctuation marks.

In order to ensure the completion and accuracy of the search, we firstly segmented the clauses in *Mencius* automatically whose language materials had already been processed by word segmentation, POS tagging and artificial proofreading. There are three criteria of clause segmentation: (1) segmenting the clause by a comma, a semicolon, or a period; (2) segmenting the clause by a comma or a semicolon; (3) only segmenting the clause by a period. We would segment clauses in line with the three criteria so as to collect three piles of language materials, then search the parallelism sentences in every pile, and in the end, by combing all the parallelism sentences, we would get the whole in *Mencius*.

2.2. The Algorithm of the Identification of the Parallelism Sentences. It is generally recognized that parallelism is a kind of rhetorical device, using three or more than three phrases or sentences which are same or similar in meaning, structure and mood, and arranging them in a side-by-side fashion to strengthen the energy of the language.

Explaining the reason by parallelism can make the words well organized; expressing emotions by parallelism can make the rhythm harmonious and embody an exuberance of feeling...In a word, articles with parallelism are often catchy and extremely convincing, and the parallelism can definitely enhance the passages' expression effect.

In summary, the features of the parallelism sentences are:

(1) Every clause is linked with another;
(2) Every clause shares same or similar structures;
(3) Some words are repeated in every clause.

Thus, we considered to design an algorithm searching the parallelism sentences according to these features.

2.2.1. Solution of the Longest Common Subsequence. Since the algorithm of the automatic identification of the parallelism sentences entails the longest common

subsequence, it is necessary to introduce this concept first. The solution of the longest common subsequence has acquired many mature and efficient algorithms, among which the most efficient one is the dynamic programming algorithm. Its idea is to decompose the problem waiting to be solved into several sub-problems, and then, from the bottom to top, to solve the sub-problems step by step so as to obtain the optimal solution in the end. Since the algorithm can store the results after solving the sub-problems, we can avoid repeated calculation when facing these problems again, which may significantly save time and improve the efficiency of the algorithm.

On account of that using the dynamic programming algorithm to solve the longest common subsequence is just a small part of our experiment, we will not go into details.

2.2.2. Connection. The clauses in one parallelism sentence are before and after connected, so searching clauses should be in order. In accordance with the existing clause segmentation language material, we designed a simple traversal algorithm.

ALGORITHM 1: THE TRAVERSAL IDENTIFICATION OF THE PARALLELISM SENTENCES

```
i=0,j=0,P=null;
while(i+2<resource.num){
   if(P==null) P=Si+Si+1+Si+2;
   else P=P+Si;
   if(functionP(Si,Si+1)!=null||functionP(Si,Si+2)!=null) i++;
   else Result=P,  P=null,  j++,  i++; return Result;
}
```

According to Algorithm 1, after traversing the language materials based on the three criteria, and combining them together, we can finally get the required parallelism sentences automatically. The Algorithm 1 is the core algorithm in our automatic identification of the parallelism sentences, containing an important sub-algorithm, P(X, Y), whose function is to judge whether clauses X and Y can compose a part of the parallelism sentence, i.e. whether the clauses, X and Y, can achieve similarity to some extent so as to ensure that the two clauses are a part of a certain parallelism sentence. See the Algorithm 4 for details about this sub-algorithm. The realization of the Algorithm 4 has utilized the other two important features of the parallelism sentences, the structural similarity and some words' repetition.

2.2.3. The Structural Similarity. The structural similarity denotes that clauses in a parallelism sentence have same or similar structural relations with each other. Under the circumstance that we could not conduct a complete analysis of the syntactic structures in *Mencius*, we could only simplify the syntactic structural similarity to the similarity between the sequences of part of speech in sentences. The sequence of part of speech indicates the sequence constructed by the word categories which are arranged in a certain order. Instead of comparing the similarity between the clauses' word structures, comparing the two sequences of part of speech of the clauses can greatly simplify the complexity of the

algorithm and enhance its efficiency. See the concrete description of the structural similarity algorithm below:

ALGORITHM 2: THE CALCULATION OF THE STRUCTURAL SIMILARITY

```
i=0,Score =0,POSi[t],POSj[n];
m=functionM (X,Y);
while(i<m.length){
   if(t-POSi.IndexOf(m.substr(i,1))<4&&n-POSj.IndexOf(m.substr(i,1))<4){
   Score=Score+1;i++;
}
```

There are some points that should be noticed: (1) The algorithm will finally obtain a score which is represented by a number denoting the structural similarity between s_1 and s_2, and will be applied into the Algorithm 1 for the solution of P(X,Y), see the Algorithm 4; (2) M(X,Y) denotes to the longest common subsequence of the strings, X and Y, which is solved by the dynamic programming algorithm as introduced before; (3) In the Step 4, the distance is the distance from the end of the string rather than the initial part, because in most parallelism sentences, the right parts (more close to the end of the string) of the clauses are more similar than the left parts (more close to the initial part of the string); (4) The Step 5 aims at the comparatively long string of part of speech, and the long strings may easily incline to the bigger longest common subsequences. If we trained the algorithm based on this, we would omit a lot of shorter strings of part of speech, which is unexpected. Therefore, we would cut down the score of the sum of the length that is over 60 to balance the difference between both the long and the short strings.

2.2.4. Some words' Repetition. The structural similarity is simulating the feature of the parallelism sentences from the perspective of part of speech, while the feature, some words' repetition, can be tested by the comparison between the strings directly. See the description of the algorithm about some words' repetition below:

ALGORITHM 3: THE CALCULATION OF SOME WORDS' REPETITION

```
i=0,Score=0,Chari[t],Charj[n];
m=functionM (X,Y);
if(m.contain(”,  ”)||m.contain(” ; ”))
   Score=Score+1;
else if(m.contain(”,  ”)||m.contain(”。”))
   Score=Score+1;
while(i<m.length){
   if(t-POSi.IndexOf(m.substr(i,1))<4&&n-POSj.IndexOf(m.substr(i,1))<4){
   Score=Score+1;i++;
}
return m-n>60: score=score-10 ? score;
```

The Algorithm 3 is quite similar to the Algorithm 2 (actually the latter one is proceeded

with perspective of the clausal string of part of speech, while the former one starts with the clause's string. One is special in that the string contains punctuation marks. Hence if m in the Step 3 simultaneously consists a comma and a semicolon, or a comma and a period, then s_1 will be more similar to s_2. Therefore, the score should plus 1.

2.2.5. The Identification of the Parallelism Sentences. The important sub-algorithm, P(X,Y) introduced in the Algorithm 1 above can be utilized to make a judgment on whether the clauses, X and Y, can construct a component of a parallelism sentence, which is realized below:

ALGORITHM 4: P(S1,S2) JUDGING WHETHER S_1 AND S_2 CAN CONSTRUCT A PARALLELISM SENTENCE

```
score1=functionS(S1,S2), score2 = functionW (S1,S2);
L1=S1.Length,L2=S2.Length;
if(L1<15||L2<15)    return false;
if(L1>2*L2||L2>2*L1)    return false;
if(S1.findlastof(" 。 ! ? ")==S1.Length-1 &&
S2.findlastof(", ; ")==S2.Length-1)    return false;
if(score1*P1+score2*P2>=P3)    return true;
```

In the Algorithm 4, the 4th,5th, and 6th step are special cases excluding the possibility that s_1 and s_2 can make a parallelism sentence. By mediating the three parameters, p_1, p_2 and p_3, we can adjust the criterion of judging whether s_1 and s_2 can form a parallelism sentence so as to adjust the final accuracy and the recall rate of the identification and elicitation of the parallelism sentences. Combing the algorithms mentioned above together, we can realize a complete algorithm for identifying all the parallelism sentences in *Mencius*:

ALGORITHM 5: THE AUTOMATIC IDENTIFICATION OF THE PARALLELISM SENTENCES IN *MENCIUS*

```
P1=0.1, P2=0.1, P3=0.2, R=0;
While(Read(file)){
   functionFirst(file.lines);}
Calculate(p,r,f);
if(r>R) R=r;
if(P2<1){P2=P2+0.1,  FuntionFive(P2);}
else P2=0.1 ;
if(P1<1){P1=P1+0.1,FuntionFive(P1);}
else P1=0.1,  P2=0.1 ;
if(P3<1){P3=P3+0.1,FuntionFive(P3);}
else Output(parallelism sentences);
```

The Algorithm 5 is the integration of the several former algorithms and has traversed some parameters in order to find out the optimal result of identification of the parallelism sentences. Since this algorithm pays more attention to the recall rate, its accuracy is

comparatively lower. For purpose of improving this situation, we added a process of filtering the search results behind the Algorithm 5.

According to the features of the parallelism sentences and by carefully analyzing the erroneous sentences in the result of the automatic identification, we have generalized 7 principles below:

(1) If in a certain result there are two periods and a question mark, then it will not be recognized as a parallelism sentence and should be deleted;

(2) If in a certain result there are two periods and a semicolon, then it will not be recognized as a parallelism sentence and should be deleted;

(3) If in a certain result there are only two periods, i.e. if the result is composed of two sentences, then it will not be recognized as a parallelism sentence and should be deleted;

(4) If in a certain result there is a period and two question marks, then it will not be recognized as a parallelism sentence and should be deleted;

(5) If in a certain result there are two semicolons before and after, and in this result the middle clause has a period, then it will not be recognized as a parallelism sentence and should be deleted;

(6) If this result is merely a sentence (ended up by a period, a question mark or an exclamatory mark), and the clause ahead is ended up by a comma or a semicolon, and the last punctuation mark is a question mark, then this result will not be recognized as a parallelism sentence and should be deleted;

(7) If a result only has commas without periods or question marks, then it will not be recognized as a parallelism sentence and should be deleted.

Filtering the results found by the Algorithm 5 based on these principles, we can be close to solving the problem that the accuracy is comparatively low.

3. Experiments and the Analysis of the Results.

3.1. The Result of the Manual Annotation. On the basis of the word segmentation and POS tagging of *Mencius*, we have made a lot of efforts to manually annotate the parallelism sentences as the standard answer to the automatic elicitation experiment of the parallelism sentences.

After the manual annotation, there are totally 74 parallelism sentences in *Mencius*, and the distribution is shown in Table 1. According to the data in Table 1, the distribution of the parallelism sentences is not uniform, in which the piece of *Mencius* containing the most parallelism sentences has such sentences 4 times more than the piece having the least parallelism sentences.

.TABLE 1 THE DISTRIBUTION OF THE PARALLELISM SENTENCES IN *MENCIUS*

Name of the Piece	The Number of the Parallelism Sentences	Name of the Piece	The Number of the Parallelism Sentences
King Hui of Liang I	9	Li Lou II	2
King Hui of Liang II	4	Wan Zhang I	6
Kung Sun Chou I	7	Wan Zhang II	7

Kung Sun Chou II	3	Kao Tzu I	4
Teng Wen Kung I	4	Kao Tzu II	5
Teng Wen Kung II	3	Chin Hsin I	3
Li Lou I	10	Chin Hsin II	7

3.2. The Result and Analysis of the Automatic Identification of the Parallelism Sentences in *Mencius*. We divided the 14 texts of *Mencius* into a training set, a development set and a test set in proportion to 6:4:4, and conducted close and open tests respectively, among which, the texts composing the training set are King Hui of Liang (I and II), Kung Sun Chou (I and II) and Teng Weng Kung (I and II); the texts constructing the development set are Li Lou (I and II) and Wan Zhang (I and II); the texts making up the test set are Kao Tzu (I and II) and Chin Hsin (I and II).

According to the parameters' adjustment of the Algorithm 5 and the filtering principles, we have gained a comparatively ideal experimental result. And the optimal parameters are: p_1=0.1; p_2=0.1; p_3=0.5.

From the result we can know that the recall rate of the parallelism sentences in *Mencius* is high by and large, most chapters even achieved at 100%. Although going through the filtering of the principles, the accuracy could not give a better performance, and the possible reasons are listed below:

First, the limitation of the length of the work. The number of manually annotated answers of *Mencius* is only 74, and the number of the parallelism sentences identified automatically is only 127 as well, which indicates that the smaller the sample size is, the easier the algorithm would make errors due to the statistical sense. For instance, in Wang Zhang First, there are only 6 parallelism sentences, and when we find 3, the recall rate would be only 0.5, and the accuracy rate would also be low.

Second, the limitation of the algorithm. Our algorithm mainly focus on the form features of the parallelism sentences, and we filter the results mainly based on the principles of the punctuation marks. This could bring about some problems. One is that some parallelism sentences may share little similarity in forms but semantically, they have a very close connection, so the algorithm which only depends on the form principles can not find out all the target sentences. Another one may be that the language materials might have its own errors, which will also cause the deviation of the result. For example, in some part of the language materials, there is a clause which contains an erroneous punctuation mark at its end; thus it will be different from other clauses, and in line with our algorithm, this situation will be excluded from the parallelism sentences.

However, of course, the clauses in *Mencius* are quite similar in form, and their character numbers are also akin to each other, possibly giving rise to a relatively lager interference, which is also one of the reasons.

Although the result of the identification is not perfect, the method we proposed has efficiently identified the parallelism sentences in *Mencius* automatically, reflecting the positive exploration to the automatic identification of parallelism sentences of the Pre-Qin documents.

3.3. The Result and Analysis of the Automatic Identification of the Parallelism Sentences in *The Analects of Confucius*. In order to test the universality of our method in other similar texts, we conducted the automatic identification experiment (by the same method) on another Pre-Qin document, *The Analects of Confucius*. It is worthwhile to notice that after training and traversing the work, we obtained the optimal parameters which still are: p_1=0.1; p_2=0.1; p_3=0.5.

In *The Analects of Confucius*, there are 42 manually annotated parallelism sentence answers. There are 4 texts that have no annotated answers, i.e. we did not think they have the parallelism sentences in accordance with our definition, and they are Li Ren, Xian Jin, Wei Zi and Zi Zhang.

The experimental result showed that the performance of the automatic identification of the parallelism sentences in *The Analects of Confucius* is better than the performance in *Mencius*, and we deemed that the reasons may be:

(1) The mean length of the sentences in *The Analects of Confucius* is shorter than the one in *Mencius*, and the sentences in *The Analects of Confucius* are easier to search. The mean length of the sentences in *Mencius* is 21.8, and the mean length of the clauses is 7.0, while the mean length of the sentences in *The Analects of Confucius* is 11.4, and the mean length of the clauses is 5.2. Thus, no matter the mean length of the sentences or clauses, the statistical data in *Mencius* are far higher than the data in *The Analects of Confucius*, which, to some extent, makes the automatic identification of the parallelism sentences in *Mencius* more difficult.

(2) In *Mencius*, the types of the parallelism sentences are overly various and the difference among these sentences is bigger. The parallelism in *Mencius* can be inside a sentence, between sentences and even between paragraphs, while the parallelism in *The Analects of Confucius* is comparatively pure and simple. In addition, there are four texts in *The Analects of Confucius* having no parallelism sentence.

It can be seen that even when confronting *The Analects of Confucius* different in genres from *Mencius*, our algorithm of automatic identification of the parallelism sentences can still keep a good performance.

4. Conclusions. Starting with the single text, *Mencius*, and by generalizing the features of the parallelism sentences in *Mencius*, making rules and designing algorithms, we proposed an efficient method for automatic identification of the parallelism sentences, and applied it to another important Pre-Qin document, *The Analects of Confucius*, and we also obtained an ideal result.

However, this method has its own deficits. For example, it depends heavily on the quality of the language materials and ignores the semantic factors of the sentences. Our future work will mainly focus on the compensation of these problems and conducting experiments on larger scales of language materials so as to provide more reliable statistical information. The method of the automatic identification of the parallelism sentences should not only meet the demands from the Pre-Qin documents, but also be more efficiently utilized in other ancient Chinese literature and even modern Chinese language materials.

5. Acknowledgment. This Work is supported by Program of Natural Science Research of Jiangsu Higher Education Institutions of China (Grant No.14KJB520023), National Social Science Foundation of China (Grant No. 15BYY096).

REFERENCES

[1] Cao Hongli, *Studies on Mencius*. Jiangsu Ancient Books Publishing House Co. Ltd, Nanjing, 1997.

[2] Zhou Wende, Yang Xiaolian, *Database of Mencius,* Bashu Pbulishing House, Chengdu, 2002.

[3] Li Shengmei, *Texual Characteristics of Parallelism*. Journal of Nanchang University, Vol.36, No.5, pp.127-133, 2005.

[4] Yu Zhizhong, *The Research on Narrative Type Prose of Suzhe*, Inner Mongolia University for the Nationalities, 2011.

[5] Zheng Cuiling, *Analysis and Implementation on Longest Common Subsequence Algorithm*, Jounal of Wuyi University, Vol.29 No.2, pp. 44-48, 2010.

[6] Yang Bojun, *The Interpretation of Analects of Confucius*, Zhonghua Book Company, Beijing, 1980.