A Novel Approach for Measuring Chinese Terms Semantic Similarity based on Pairwise Sequence Alignment

Shuo Xu[#], Li-jun Zhu[#], Xiao-dong Qiao[#1], Cun-xiang Xue[#*]

[#]*Information Technology Supporting Centre, Institute of Scientific and Technical Information of China*
No. 15 Fuxing Rd., Haidian District, Beijing 100038, P.R. China

[*]*Department of Information Management, Nanjing University of Science and Technology*
No. 200 Xiaolingwei Street, Xuanwu District, Nanjing 210094, P.R. China

{xush, zhulj, qiaox, xuecx}@istic.ac.cn, [1]Corresponding author

***Abstract*—In this study, we first give a problem formulation for Chinese terms semantic similarity calculation. After that, on closer examination, we find that the traditional approach makes an implicit assumption that the order of corresponding primitive terms for two terms is roughly consistent. In other words, it doesn't consider how the difference in the order affects the quality of correspondence. To overcome this problem, a novel approach based on pairwise sequence alignment is proposed. Finally, an experimental evaluation is conducted, and the result indicates that our approach outperforms or matches at least the traditional one in the majority of cases.**

I. INTRODUCTION

Terms semantic similarity is broadly used in many applications, such as intelligent information retrieval, text clustering/classification, word sense disambiguation, example-based machine translation, etc. At the present time, there are many quantitative methods used to compute terms semantic similarity, which tend to fall into two kinds: one is based on a semantic taxonomy [1]-[8]; the other is based on collocations of words in a corpus [9]-[12], where the former is main focus in this study. A semantic taxonomy is often called a semantic knowledge database (SKD), and a few popular semantic knowledge databases include Tongyici Cilin [13] [14], HowNet [15], WordNet [16], among others.

However, any SKD is not complete and the granularity is usually very fine, that is to say, it is impossible for a SKD to collect all words in real-world applications, especially compound words in science and technology. As a result, semantic similarity between many terms cannot be calculated directly. Before illustrating how to solve this problem, some definitions first are given here. The word in a given SKD is called as a *primitive term* (PT) [7][8]. The word that is not included in the SKD and composed of two or more primitive terms is called as a *combined term* (CT) [7][8]. Primitive term and combined term are collectively known as *term*.

Formally speaking, given a SKD $D = \{PT_1, PT_2, \ldots, PT_K\}$, each element in D is a primitive term, and the word CT defined below is a combined term:

$$CT = PT_{i_1} PT_{i_2} \cdots PT_{i_n},\ CT \notin D,\ PT_{i_j} \in D, j = 1, \ldots, n,\ n \geq 2. \quad (1)$$

For a combined term CT, the position of its primitive term is definite, so each CT can be represented as an ordered list, i.e.,

$$CT \equiv < PT_{i_1}, PT_{i_2}, \cdots, PT_{i_n} > . \quad (2)$$

For the sake of consistence, a primitive term PT is also represented similarly as <PT>. Additionally, in order to make refer the position information of each primitive term easily, we define a rank function R for a term $T = < PT_{i_1}, PT_{i_2}, \ldots, PT_{i_n} >$ ($n \geq 1$) and a primitive term $PT \in D$ as follows.

$$R(T, PT) = \begin{cases} j, & \text{if } PT = PT_{i_j} \\ 0, & \text{otherwise} \end{cases} . \quad (3)$$

Now return to semantic similarity calculation problem, which can be stated formally as follows: Given a SKD $D = \{PT_1, PT_2, \ldots, PT_K\}$, for any two terms $T_1 = <PT_{1,1}, PT_{1,2}, \ldots, PT_{1,m}>$, $T_2 = <PT_{2,1}, PT_{2,2}, \ldots, PT_{2,n}>$, to calculate the semantic similarity between T_1 and T_2, denoted as $Sim(T_1, T_2)$. If both T_1 and T_2 are primitive terms, $Sim(T_1, T_2)$ can be calculated directly according to the work (See section 3) (Type-I problem). Otherwise, the usual procedure is to first establish the correspondence between primitive terms from T_1 and T_2, and then to make a weighted summation according to a certain criterion (See section 4) (Type-II problem).

On closer examination, we find that this traditional method for Type-II problem makes an implicit assumption that the order of corresponding primitive terms for two terms is roughly consistent. However, there are a lot of term pairs in real world applications that do not meet this assumption, e.g., <燃气, 汽车> "gas vehicle" and <汽车, 燃气> "gas for vehicle". Furthermore, the definition of combined terms is not concerned with whether a term is valid or not, so that this may result in very high similarity between valid and invalid terms. By a valid term here, we mean that the term has definite meaning. Otherwise, it is invalid. For instance, <汽车, 车灯> "automotive lamp" is valid, but <车灯, 汽车> "lamp's automobile" is invalid. Therefore, we conjecture that this may have an effect on some applications.

To solve this problem, this paper proposes a novel approach based on pairwise sequence alignment. Since this

978-0-7695-3810-5/09 $26.00 © 2009 IEEE
DOI 10.1109/SKG.2009.34

paper puts focus on semantic similarity calculation for Type-II problem, we adopt Tongyici Cilin as our SKD for simplicity. What needs to explain, this approach is also applicable to the other SKDs. The organization of the rest of this paper is as follows. Tongyici Cilin is briefly described in Section 2; Semantic similarity calculation for Type-I problem is presented in Section 3. The traditional and novel approaches of semantic similarity calculation for Type-II problem are presented in Section 4. In Section 5 an experimental evaluation is conducted, and Section 6 concludes this paper.

II. Description of Tongyici Cilin

Tongyici Cilin (Cilin1 in short) [13] is a Chinese thesaurus published in 1983, which defines a three-level semantic taxonomy tree (a dummy root node needs to be added) for 53,859 words. These words consist of 12 major classes labelled by English upper case letters, 94 medium classes labelled by English lower case letters, and 1428 minor classes labelled by 2-digit numbers. All the words in the same leaf node, i.e., third level node, are regarded as synonyms.

Since some words in Cilin1 become uncommon ones, and many new words are not yet joined in, Tongyici Cilin (extension edition, Cilin2 in short) [14] is created from Cilin1 through expansion and refinement. In Cilin2, 39,099 high frequency words in Cilin1 are reserved and another 38,244 ones are introduced from other resources, such as People's Daily corpus. Cilin2 defines a five-level semantic taxonomy tree (similarly, to add a dummy root node), where the upper three levels are the same as those in Cilin1. The forth level in Cilin2, called synset, corresponds to every paragraph of the third level in Cilin1 and is labelled by English upper case letters. The fifth level in Cilin2, called subsynset, corresponds to every line of the third level in Cilin1 and is labelled by the line number.

TABLE I
Chinese Word Code Table in Cilin2

Position	1	2	3	4	5	6	7	8
Example of the label	B	o	2	1	A	2	6	#\ =\ @
Meaning of the label	Major class	Medium class	Minor class		Synset	Sub-synset		
Level	First	Second	Third		Forth	Fifth		

In addition, the lines of the third level in Cilin1 can be divided further into 3 cases: some lines are synonyms, some are related words, and others only contain one word. In some applications, these need to be treated differently, so three symbols (=, #, @) are utilized. Thus, each word in Cilin2 can be represented by a code of length 8 (See Table I for details). Of course, this correspondence is not one-to-one. That is, each code may correspond to multiple words, e.g., the code "Aa01A01=" corresponds to all elements in {人, 士, 人物, 人士, 人氏, 人选}. Likewise, each word may also correspond to multiple codes, e.g., the word "人" corresponds to all codes in {Aa01A01=, Ab02B01=, Dd17A02=, De01B02=, Dn03A04=}.

For convenience, we further introduce some notation. Define a function *Code* (*PT*) as a set of all codes corresponding to the primitive term *PT*. For example, *Code* (人) = {Aa01A01=, Ab02B01=, Dd17A02=, De01B02=, Dn03A04=}. And let lower case letter *c* represent an element in this set, i.e., $c \in Code(PT)$.

III. Semantic Similarity for Type-I Problem

From the viewpoint of information theory, the similarity between two objects is related to their commonality and differences [17]. Based on this point, semantic similarity between codes c_1 and c_2 can be defined as follows [7][8]:

$$Sim(c_1, c_2) = \frac{2 \times Spd(c_1, c_2)}{Dsd(c_1, c_2) + 2 \times Spd(c_1, c_2)}. \quad (4)$$

where $Spd(c_1, c_2)$ and $Dsd(c_1, c_2)$ is superposed degree and dissimilitude degree between c_1 and c_2, respectively. For a semantic taxonomy tree, such as Cilin2, $Spd(c_1, c_2)$ is the length of path shared by c_1 and c_2, and $Dsd(c_1, c_2)$ is the length of the shortest path between two leaf nodes represented by c_1 and c_2. For a SKD based on Cilin2, it is easy to check that Eq. (4) can be simplified as [8]

$$Sim(c_1, c_2) = Spd(c_1, c_2)/5. \quad (5)$$

Now we can define the semantic similarity between primitive terms PT_1 and PT_2 as follows [6]-[8]:

$$Sim(PT_1, PT_2) = \max_{c_1 \in Code(PT_1)} \max_{c_2 \in Code(PT_2)} Sim(c_1, c_2). \quad (6)$$

IV. Semantic Similarity for Type-II Problem

In this section, we will solve the Type-II problem. That is, given any two terms $T_1 = <PT_{1,1}, PT_{1,2}, \ldots, PT_{1,m}>$, $T_2 = <PT_{2,1}, PT_{2,2}, \ldots, PT_{2,n}>$, to calculate the semantic similarity between T_1 and T_2. Without loss of the generality, let $m \le n \ge 2$. First the problem on the existing method is analysed in subsection *A*, and then a novel approach based on pairwise sequence alignment is presented in subsection *B*.

A. Traditional Approach

The usual procedure [6]-[8] is to first establish a correspondence between primitive terms from T_1 and T_2. That is, a correspondence set *CS* like below needs to be built:

$$CS = \left\{ PT_{1,1} \leftrightarrow PT_{2,j_1}, PT_{1,2} \leftrightarrow PT_{2,j_2}, \cdots, PT_{1,m} \leftrightarrow PT_{2,j_m} \right\}. \quad (7)$$

The strategy for building *CS* is slightly different in the literature. Here we follow the strategy in [8], and corresponding pseudo code is given below.

Algorithm 1: Building the correspondence

Input: Two terms $T_1 = <PT_{1,1}, PT_{1,2}, \ldots, PT_{1,m}>$, $T_2 = <PT_{2,1}, PT_{2,2}, \ldots, PT_{2,n}>$ and a SKD *D*.
Output: Correspondence set *CS* between primitive terms from T_1 and T_2.

1. $CS = \phi$;
2. FOR $i = m$ TO 1, STEP = –1
   // If there is a tie in step 2.1, the one with larger subscript is

preferred.

2.1 $j \leftarrow \arg\max_{PT_{2,j} \in T_2} Sim\left(PT_{1,i}, PT_{2,j}\right)$;

2.2 $CS \leftarrow CS \cup \{PT_{1,i} \leftrightarrow PT_{2,j}\}$;

2.3 $T_2 \leftarrow T_2 - PT_{2,j}$;

END FOR

Note that we adopt the symbols representation associated with a set. For instance, $PT_{2,j} \in T_2$ means that T_2 contains $PT_{2,j}$. $T_2 - PT_{2,j}$ means that $PT_{2,j}$ is removed from T_2. By the way, the running time of the algorithm is in $\mathcal{O}(m \times n)$, and the amount of memory used is in $\mathcal{O}(m+n)$.

Now semantic similarity can be calculated as follows:

$$Sim(T_1,T_2) = \alpha \times \left(\frac{1}{m}+\frac{1}{n}\right) \times \sum_{i=1}^{m} Sim(PT_{1,i}, PT_{2,j_i}) + (0.5-\alpha) \times \frac{m}{n} \times \sum_{i=1}^{m} \left\{ \left[\frac{R(T_1, PT_{1,i})}{\sum_{i'=1}^{m} i'} + \frac{R(T_2, PT_{2,j_i})}{\sum_{j'=1}^{n} j'} \right] \times Sim(PT_{1,i}, PT_{2,j_i}) \right\}, \quad (8)$$

where type value of α is 0.3 and Sim (PT_1, PT_2) is the semantic similarity between primitive terms PT_1 and PT_2.

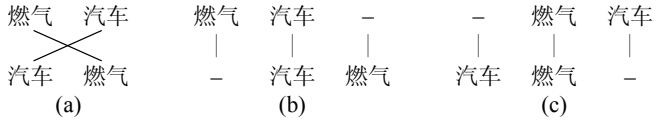

Fig. 1 The correspondence between T_1 and T_2.

Example 1. Let T_1 = <燃气, 汽车> "gas vehicle", T_2 = <汽车, 燃气> "gas for vehicle". According to Algorithm 1, the correspondence between T_1 and T_2 is illustrated in Fig. 1 (a). Then by Eq. (8), the semantic similarity between T_1 and T_2 is

$$Sim(T_1,T_2) = 0.3 \times \left(\frac{1}{2}+\frac{1}{2}\right) \times 2 + 0.2 \times \frac{2}{2} \times \left[\left(\frac{1}{1+2} + \frac{2}{1+2} \right) \times 1 + \left(\frac{2}{1+2} + \frac{1}{1+2} \right) \times 1 \right] = 1.0 .$$ □

Obviously, it is somewhat unacceptable (Zhang [18] also observed this phenomenon, and developed a model for Chinese string similarity based on multi-level features). On closer examination, we find that this approach makes an implicit assumption that the order of corresponding primitive terms for T_1 and T_2 is roughly consistent. In other words, it doesn't consider how the difference in the order affects the quality of correspondence. However, the order is not trivial for two Chinese terms, because Chinese terms formation has a characteristic that the primitive term that expresses centre meaning usually lies in back part of the terms. For the Example 1 above, it seems that the correspondence in Fig. 1 (b) or (c) is more proper, in which the symbol "–" denotes a gap (see further). Next subsection will consider how to construct such correspondence.

B. A Novel Approach based on Pairwise Sequence Alignment

In bioinformatics, a sequence alignment is a way of arranging the sequences of DNA, RNA or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Several versions of this problem occur in practice, depending on whether one is interest in alignments involving the entire sequences or just substrings of them. This leads to the definition of global and local alignments, both of which can be solved efficiently by dynamic programming [19]. For more elaborate and detailed surveys we refer the readers to [20]. It is worth mentioning that the sequence alignment algorithm has been successfully used to generate candidate patterns in auto-construction of conceptual relations [21]. In this paper, we mainly consider global sequence alignments.

Now if we see each primitive term as a nucleotide or amino acid residue and each term as a sequence, by analogy analysis it is not difficult to find that the problem to construct the correspondence similar to Fig. 1 (b) or (c) can be seen as finding a global alignment between two sequences, which is main idea of our approach. Taking the characteristic of Chinese terms formation into consideration, we will build the alignment from back to front, which is contrary to that in bioinformatics. What follows is a brief introduction of Needleman-Wunsch algorithm (NW algorithm in short) [20][22], which performs a global alignment on two sequences.

Main idea of NW algorithm is to find the alignment with the highest score. In order to compute the score of each alignment, we need the scores for aligned primitive terms, i.e., semantic similarity between primitive terms. Additionally, a matrix needs to be allocated, which is often called the F matrix, and its (i, j)-th entry is often denoted $F_{i,j}$ (i along horizontal axis and j along vertical axis). There is one row for each primitive term in T_1 and one column for each primitive term in T_2. As the algorithm progresses, the $F_{i,j}$ will be assigned to be the optimal score for the alignment of the last i primitive terms in T_1 and the last j primitive terms in T_2.

The principle of optimality is then applied as follows.

Basis: $F_{i,n+1} \leftarrow d \times (m-i+1)$, $F_{m+1,j} \leftarrow d \times (n-j+1)$; $i = 1, 2, \ldots, m+1$; $j = 1, 2, \ldots, n+1$.

Recursion: $F_{i,j} \leftarrow \max(F_{i+1,j+1} + Sim(PT_{1,i}, PT_{2,j}), F_{i,j+1} + d, F_{i+1,j} + d)$; $i = m, m-1, \ldots, 1$; $j = n, n-1, \ldots, 1$.

Here d is a gap penalty ($d = -0.05$ in this paper). Once the F matrix is filled, the top left hand corner of the matrix is the maximum score for any alignment. To find which alignment actually gives this score, one can start from the top left cell, and compare the value with the three possible sources to see which it comes from as follows. If there is a tie, Case 1 is preferred, and then Case 2, and last Case 3.

Case 1: IF $F_{i,j} = F_{i+1,j+1} + Sim(PT_{1,i}, PT_{2,j})$, THEN $PT_{1,i}$ and $PT_{2,j}$ are aligned;

Case 2: IF $F_{i,j} = F_{i,j+1} + d$, THEN $PT_{2,j}$ is aligned with a gap;

Case 3: IF $F_{i,j} = F_{i+1,j} + d$, THEN $PT_{1,i}$ is aligned with a gap.

In fact, it is not necessary to output/save the optimal alignment, since the semantic similarity can be calculated during the procedure.

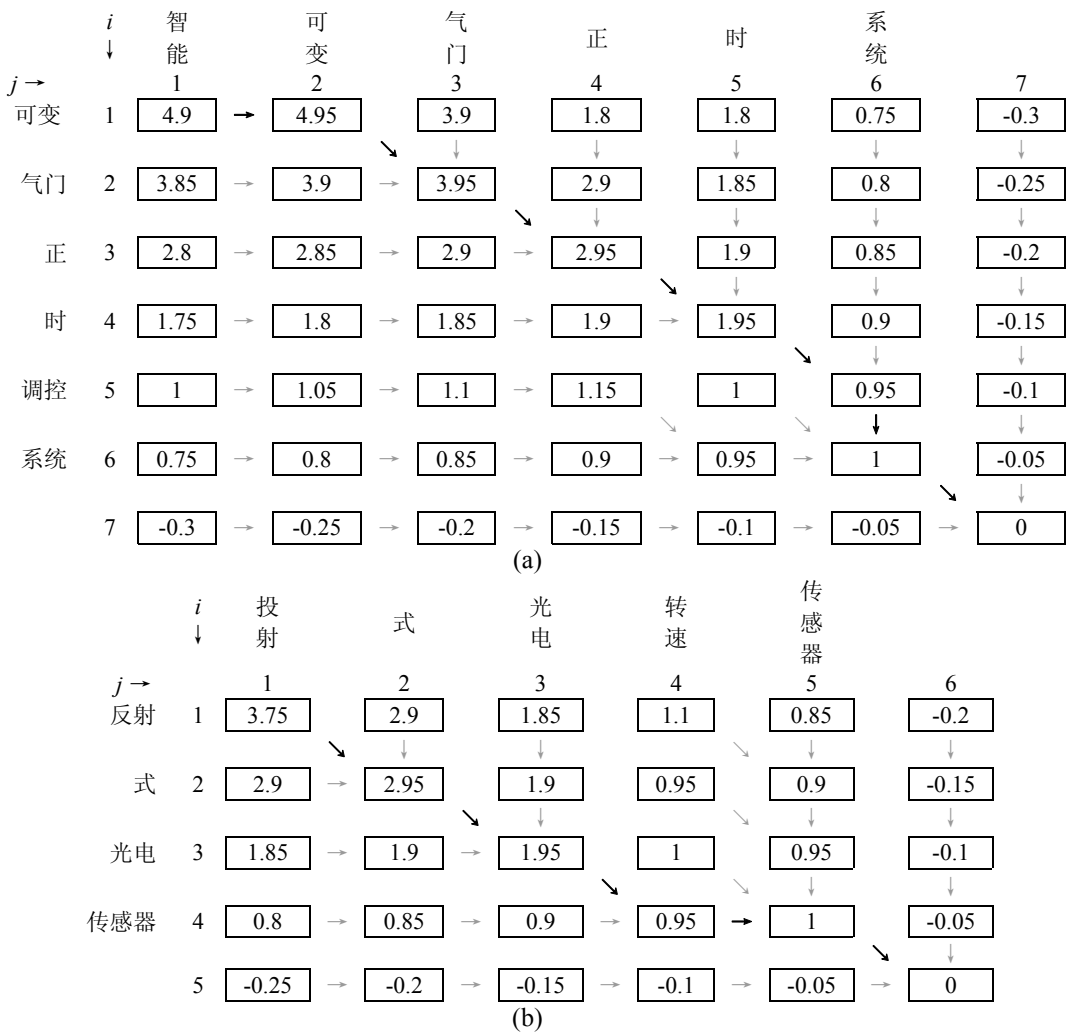

Fig. 2 The matrix F for computing optimal alignments between T_1 and T_2 (a), and between T_3 and T_4 (b).

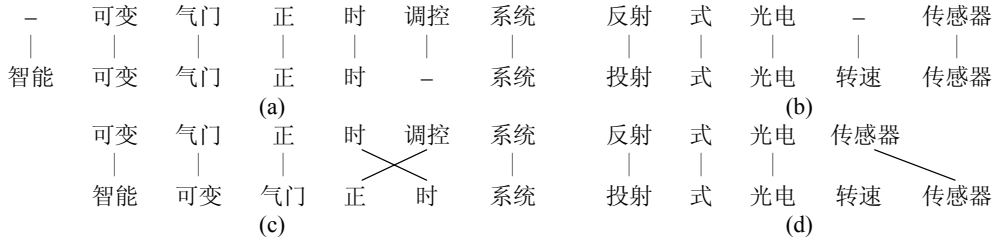

Fig. 3 The optimal alignment between T_1 and T_2 (a), and between T_3 and T_4 (b) by NW algorithm, as well as the correspondence between T_1 and T_2 (c), and between T_3 and T_4 (d) by the traditional approach.

Example 2. Let T_1 = <可变, 气门, 正, 时, 调控, 系统> "variable valve timing regulatory system", T_2 = <智能, 可变, 气门, 正, 时, 系统> "intelligent variable valve timing system", T_3 = <反射, 式, 光电, 传感器> "reflective photoelectric sensor", T_4 = <投射, 式, 光电, 转速, 传感器> "projected photoelectric sensor of speed measuring".

Fig. 2 shows the matrix F corresponding to T_1 and T_2 (a), T_3 and T_4 (b), respectively. The arrow in the right, lower or lower right of each cell indicates which source it comes from. The arrow in black reveals the optimal alignments, which is also illustrated in Fig. 3 (a) and (b) for clearness. What's more, the correspondences between T_1 and T_2 (c), T_3 and T_4 (d) by Algorithm 1 are shown in Fig. 3.

By comparison between (a) and (c) as well as (b) and (d) in Fig. 3, one can easily observe that if the order of corresponding primitive terms for two terms roughly agrees, the correspondences obtained by these two methods are the

same. Otherwise, our approach seems to be superior. This point is validated through experiments in next section. □

In addition, the complexity of NW algorithm is $\mathcal{O}(m \times n)$ both for time and space, viz., the algorithm has quadratic complexity. As regards the space, however, it is possible to improve complexity from quadratic to linear and keep the same generality [22]. The price to pay is an increase in processing time, which will roughly double. Nevertheless, the asymptotic time complexity is still the same. But we keep the space quadratic for faster processing time, because the number of primitive terms of a term is often not much.

Finally, in order to calculate the semantic similarity between T_1 and T_2 from the optimal alignment, we still adopt the Eq. (8), but the primitive terms that are aligned with gaps are excluded.

Example 3. Let's consider the terms in Example 1 again, repeated here for convenience, T_1 = <燃气, 汽车> "gas vehicle", T_2 = <汽车, 燃气> "gas for vehicle". The alignment in Fig. 1 (c) is used, and then semantic similarity between T_1 and T_2 is

$$Sim(T_1, T_2) = 0.3 \times \left(\frac{1}{2} + \frac{1}{2}\right) \times 1 + 0.2 \times \frac{2}{2} \times \left(\frac{1}{1+2} + \frac{2}{1+2}\right) \times 1 = 0.5$$

Thus if a threshold greater than 0.5 is set, one cannot conclude that T_1 is similar to T_2, opposed to the traditional approach. □

V. Experimental Results and Discussions

Currently, there is no internationally agreed standard for assessing the performance of terms (esp., combined terms) semantic similarity calculation. In our opinion, the reason is two-fold: (1) semantic similarity is a very subjective concept, which differs not only from person to person, but also from application to application; (2) To the best of our knowledge, there is no public benchmark dataset related to Chinese terms semantic similarity calculation. Hence, semantic similarities between some terms from our application are given in Table II, mainly in order to give some insights for further study.

TABLE II
SEMANTIC SIMILARITIES BETWEEN SOME TERMS FROM OUR APPLICATION. METHOD1 REFERS TO THE TRADITIONAL APPROACH AND METHOD2 OUR APPROACH.

ID	Term1	Term2	Method1	Method2
1	<燃气, 汽车> "gas vehicle'	<汽车, 燃气> "gas for vehicle"	1.0	0.5
2	<前轮, 驱动> "front-wheel drive"	<驱动, 轮> "driving wheel"	0.9	0.5
3	<电阻, 式, 传感器> "resistive sensor"	<陶瓷, 电容器> "ceramic capacitor"	0.65	0.3611
4	<可变, 气门, 正, 时, 调控, 系统> "variable valve timing regulatory system"	<智能, 可变, 气门, 正, 时, 系统> "intelligent variable valve timing system"	0.4686	0.8429
5	<直流, 电动机, 驱动> "direct current motor drive"	<驱动, 电动机> "drive motor"	0.7444	0.3833
6	<点火, 控制, 计算机> "spark-control computer"	<微机, 控制, 点火, 系统> "ignition system controlled by microcomputer"	0.765	0.255
7	<驱动, 电动机> "drive motor"	<四, 轮, 驱动> "four-wheel drive"	0.5144	0.3611
8	<点火, 提前, 作用, 板> "advance plate"	<离心, 点火, 提前, 机构> "centrifugal advance mechanism"	0.038	0.518
9	<多, 片, 离合器> "multiple disk clutch"	<离合器, 压, 板> "clutch pressure plate"	0.82	0.4867
10	<汽车, 车灯> "automotive lamp"	<车灯, 汽车> "lamp's automobile"	1.0	0.5
11	<防, 抱, 死, 制动, 系统> "anti-lock braking system"	<制动, 防, 抱, 死, 系统> "anti-lock braking system"	1.0	0.8133
12	<动力, 制动, 系统> "dynamic braking system"	<制动, 力, 系统> "braking force system"	0.94	0.7
13	<辅助, 制动, 系统> "brake assist system"	<制动, 辅助, 系统> "brake assist system"	1.0	0.7
14	<柴油, 发动机> "diesel motor"	<汽油, 引擎> "gasoline engine"	1.0	1.0
15	<反射, 式, 光电, 传感器> "reflective photoelectric sensor"	<投射, 式, 光电, 转速, 传感器> "projected photoelectric sensor of speed measuring"	0.785	0.785
16	<磁, 粉, 式, 安全, 联, 轴, 器> "magnetic powder safety coupling"	<销钉, 式, 安全, 联, 轴, 器> "pin type safety coupling"	0.8033	0.8020
17	<阀> "valve"	<释放, 阀> "relief valve"	0.6167	0.6167
18	<释放, 阀> "relief valve"	<机油, 压力, 释放, 阀> "oil pressure relief valve"	0.62	0.62
19	<阀> "valve"	<机油, 压力, 释放, 阀> "oil pressure relief valve"	0.445	0.445
20	<多晶硅, 薄膜, 太阳能, 电池> "polycrystal silicon thin film solar battery"	<非, 晶, 硅, 薄膜, 太阳能, 电池> "amorphous silicon thin film solar battery"	0.7476	0.7476

Since most of terms in real-world applications are combined ones, Chinese terms must be segmented in the first place. In order to make the best of the knowledge in Cilin2, we take Cilin2 as our lexicon. For word segmentation method, forward maximum match (FMM) [23] and backward maximum match (BMM) [23] with manually correcting are applied at the same time. In other words, if the results by FMM and BMM don't agree with each other, the more reasonable one is selected from two results. Of course, there still exist some errors in segmenting, but we let them alone here.

From the results (ID = 1, 2, …, 10) in Table 2, it is not difficult to see that our approach can avoid very well the problem mentioned above, where the term <车灯, 汽车> "lamp's automobile" for ID = 10 is invalid, but the traditional approach gives semantic similarity 1.0. This point agrees with our analysis in the introduction. However, there are always the exceptions due to the complexity of Chinese language phenomenon, e.g., term pairs for ID = 11, 12, 13. These term pairs express the same meaning, but our approach does not give a semantic similarity of 1.0 or close to 1.0. Fortunately, this case is very small relatively according to our preliminary statistics. What's more, because the difference is the order of primitive terms which don't express centre meaning, the semantic similarity obtained by our approach can still be acceptable.

If the order of corresponding primitive terms for T_1 and T_2 is roughly consistent, these two methods give almost exact the results, such as, ID = 15, 16, …, 20 in Table II. Additionally, what needs to explain, the similarity 1.0 does not always mean that the involved terms are equivalent, e.g. T_1 = <柴油, 发动机> "diesel motor", T_2 = <汽油, 引擎> "gasoline engine" (ID = 14). The main reason is that we cannot consider the 8th position (see Table I) in the code of each primitive term when calculating semantic similarity between primitive terms.

There is another interesting phenomenon in Table II. If the threshold of 0.6 is set, some hypernym-hyponym relations can be extracted, such as T_1 = <阀> "valve" and T_2 = <释放, 阀> "relief valve" (ID = 17), T_2 and T_3 = <机油, 压力, 释放, 阀> "oil pressure relief valve" (ID = 18), however, the others cannot, such as T_1 and T_3 (ID = 19). This is mainly caused by the weights in Eq. (8), which are concerned with the number of corresponding primitive terms. Nevertheless, the relationship between T_1 and T_3 can be inferred simply from T_1 & T_2 and T_2 & T_3.

In the end, the incompleteness of Cilin2 is also observed from Table II, say, T_1 = <多晶硅, 薄膜, 太阳能, 电池> "polycrystal silicon thin film solar battery", T_2 = <非, 晶, 硅, 薄膜, 太阳能, 电池> "amorphous silicon thin film solar battery". If Cilin2 includes the primitive term <非晶硅> "amorphous silicon", semantic similarity between T_1 and T_2 may be higher and more intuitive.

VI. Conclusions

In this paper, we mainly consider Chinese terms semantic similarity calculation for Type-II problem, that is, both of involved terms are not primitive terms. After problem formulation, we analyse in detail the traditional approach. It turns out that it does not consider how the difference in the order of corresponding primitive terms for two terms affects the quality of correspondence, which results in poor performance in some cases. By analogy analysis, we think that the procedure of building correspondence can be seen as a sequence alignment problem. Therefore, a novel approach based on sequence alignment is put forward, thus overcoming the underlying problem on the traditional approach.

Acknowledgment

We thank Information Retrieval (IR) laboratory, Harbin Institute of Technology (HIT) for providing us with the Tongyici Cilin (extension edition). This work was funded by the "Research and Implementation of Knowledge Organizing System Integration & Service Architecture", which is sponsored by Key Technologies R&D Program of Chinese 11th Five-Year Plan (2007-2009) under grant number 2006BAH03B03.

References

[1] E. Agirre and G. Rigau, "A Proposal for Word Sense Disambiguation using Conceptual Distance," in *International Conference Recent Advances in Natural Language Processing (RANLP)*, 1995, pp. 258–264, Tzigov Chark, Bulgaria.

[2] Q. Liu and S. J. Li, "Word Similarity Computing based on How-net," in *The 3rd Chinese Lexical Semantics Workshop*, 2002, Taipei. (in Chinese)

[3] K. J. Chen and J. M. You, "A Study on Word Similarity using Context Vector Models," *Computational Linguistics and Chinese Language Processing*, vol. 7, pp. 37–58, 2002.

[4] H. M. Tran and S. Dan, "Word Similarity in WordNet," in *The 13th International Conference on High Performance Scientific Computing*, 2006, pp. 293–302, Hanoi, Vietnam.

[5] X. Y. Liu, Y. M. Zhou, and R. S. Zheng, "Measuring Semantic Similarity in WordNet," in *the 6th International Conference on Machine Learning and Cybernetics*, 2007, pp. 3431–3435, Hong Kong

[6] C. Z. Zhang, "Research on Synonyms Dictionary-based on Recognition of Synonyms," *Journal of Huaiyin Institute of Technology*, vol. 13, pp. 59–62, 2004. (in Chinese)

[7] T. Xia, "Study on Chinese Words Semantic Similarity Computation," *Computer Engineering*, vol. 33, pp. 191–194, 2007. (in Chinese)

[8] W. R. Wang, "Dynamic Construction of the Lexical Knowledge System: Approaches and Tools Implementation," M. Chinese thesis, Institute of Scientific and Technical Information of China, Beijing, China , 2008.

[9] J. Z. Li, C. N. Huang, and E. H. Yang, "An Adaptive Chinese Word Sense Disambiguation Method," *Journal of Chinese Information Processing*, vol. 13, pp. 1–8, 1999. (in Chinese)

[10] S. Lu and S. Bai, "To Calculate the Distance between Words," in *The 6th National Joint Symposium of Computational Linguistics*, 2001, Taiyuan, China. (in Chinese)

[11] I. Dagan, S. Marcus, and S. Markovitch, "Contextual Word Similarity and Estimation from Sparse Data," in *Proceedings of the Annual Meeting the Association for Computational Linguistics (ACL)* , 1993, pp. 164–171.

[12] I. Dagan, L. Lee, and F. C. N. Pereira, "Similarity-based Models of Word Cooccurrence Probabilities. Machine Learning," *Special Issue on Machine Learning and Natural Language*, vol. 34, pp. 43–69, 1999.

[13] J. J. Mei, Y. M. Zhu, Y. Q. Gao, and H. X. Yin, *Tongyici Cilin*, Shanghai Lexicographical Publishing House, Shanghai, China, 1983. (in Chinese)

[14] Tongyici Cilin (Extension Edition). [Online]. Available: http://www.ir-lab.org

[15] HowNet. [Online]. Available: http://www.keenage.com/html/e_index.html

[16] WordNet. [Online]. Available: http://wordnet.princeton.edu

[17] D. Lin, “An Information-Theoretic Definition of Similarity,” in *The 15th International Conference on Machine Learning*, 1998, pp. 296–304.
[18] C. Z. Zhang, “A Model for Chinese String Similarity based on Multi-Level Features,” *Journal of the China Society for Scientific and Technical Information*, vol. 24, pp. 696–701, 2005. (in Chinese)
[19] S. R. Eddy, “What is Dynamic Programming?” *Nature Biotechnology*, vol. 22, pp. 909–910, 2004.
[20] J. C. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*, PWS Publishing, MA, USA, 1997.
[21] Y. Hu, R. Z. Lu, and H. Liu, “Information Retrieval Oriented Auto-Construction of Conceptual Relations,” *Journal of Chinese Information Processing*, vol. 21, pp. 46–50, 2007. (in Chinese)
[22] S. B. Needleman and C. D. Wunsch, “A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins,” *Journal of Molecular Biology*, vol. 48, pp. 443–453, 1970.
[23] X. H. Chen, “Automatic Analysis of Contemporary Chinese using Visual C++,” Beijing Language and Culture University Press, Beijing, China, pp. 90–103, 2000. (in Chinese)