Extraction of Shallow Language Patterns: An Approximation of Data Oriented Parsing

Samuel W. K. Chan

Dept. of Decision Sciences,
The Chinese University of Hong Kong,
Hong Kong, China
swkchan@cuhk.edu.hk

Abstract. This paper presents a novel approach to extracting shallow language patterns from text. The approach makes use of an attributed string matching technique which is based on two major but complementary factors: *lexical similarities* and *sentence structures*. The technique takes full advantage of a huge number of sentence patterns in a Treebank, while preserving robustness, without being bogged down into a complete linguistic analysis. The ideas described are implemented and an evaluation of 5,000 Chinese sentences is examined in order to justify its statistical significances.

1 Introduction

Identifying language patterns is a fundamental task in natural language processing. Competent speakers of a language hardly ever fail to recognize the language patterns, like verb-object pairs, various phrases structures, semi-idiomatic expressions, and platitudes of that language. Even though these patterns have been found useful in various application areas, including information extraction, textual summarization, and even bilingual alignment, the demands for a highly efficient sentence patterns extraction system are still mounting.

One of the most important sentence patterns is the case frame which can be used to represent the meaning of sentences [10]. In general, a case frame is to be understood as an array of slots, each of which is labelled with a case name, and eventually possibly filled with a case filler, the whole system representing the underlying structure of an input sentence. Certainly, one approach to extracting the case frames is to obtain a full parse of a sentence. However, having a complete parse tree for a sentence is difficult in many cases. An alternative approach is the shallow parsing which tries to circumvent the complexity of full parsing and analyzes a sentence at the level of phrases and the relations between them [1]. As a branch in shallow parsing, data-oriented parsing (DOP) models embody the assumption that humans produce and interpret natural language utterances by invoking representations of their concrete past language experiences, rather than the rules of a consistent and non-redundant competence grammar [3]. DOP models usually maintain large corpora of sentences annotated with syntactic and semantic structures. New input sentences are analyzed by combining partial structures from the corpus. The occurrence frequencies of these

G.I. Webb and Xinghuo Yu (Eds.): AI 2004, LNAI 3339, pp. 574–586, 2004.

structures are employed to estimate which of the resulting analyses are the most suitable patterns for the input sentence.

In this research, a shallow but effective sentence chunking process is designed and developed. This sentence chunking process is to extract all the phrases from the input sentences, without being bogged down into deep semantic parsing and understanding. On the other hand, while several criteria for recognizing sentence patterns, such as case frames, in sentences have been considered in the past, none of the criteria serves as a completely adequate decision procedure. Most of the studies in computational linguistics do not provide any hints on how to map input sentences into case frames automatically, particularly in Chinese [6]. Our second objective is to assign the extracted phrases into their corresponding case roles based on a corpus of utterances annotated with labeled trees or subtrees. One of our primary goals in this research is to design a shallow but robust mechanism which can extract Chinese sentence patterns [14, 15, 4, 5]. Even though the classical syntactic and semantic analysis in Chinese is extremely difficult, if not impossible, to systematize in the current computational linguistics research, this DOP does not require any deep linguistic analysis to be formalized. Consequently, the annotated sentences will give piecemeal the underlying semantic representation, without being mired into the formalism. The organization of the paper is as follows. The related work in DOP and text chunking are first described in Section 2. We have employed a corpus annotated with the labeled trees. The characteristics of the Treebank that supports our approach will also be explained. In this research, each Chinese token will have two attributes, namely Part-of-Speech (*POS*) and Semantic Classes (*SC*). Any input sentence can be viewed as an attributed string. The detailed discussion on how an attributed string matching algorithm can be used in the shallow patterns extraction is shown in Section 3. The system has already been implemented using Java language. In order to demonstrate the capability of our system, an experiment with 5,000 sentences is conducted. It is explained in Section 4 followed by a conclusion.

2 Related Work

A number of systems have been developed to perform shallow parsing. The early approaches in shallow parsing that mainly came out of the Message Understanding Conferences (MUC) have demonstrated their capabilities for extracting noun groups, verb groups, and particles [11]. For example, the FASTUS is a system which is designed for extracting information from free text. One of the applications is to mark text with annotations that indicate items of interest, such as names of people or companies. It also fills up database templates with information that could be then entered into a relational database [2]. Similarly, the SPARKLE (Shallow PARsing for acquisition of Knowledge for Language Engineering) aims to develop shallow parsing technology in four European languages together with corpus-based lexical acquisition techniques, and deploy parsers in multilingual information retrieval and speech dialogue systems (`http://www.informatics.susx.ac.uk/research/nlp/sparkle/sparkle.html`). Their shallow parsing is carried out by a generalized

LR parser, which uses a unification-based phrasal grammar of POS tags. The resulting parses will serve for acquiring lexical information.

Extracting shallow language patterns involves chunking sentences into segments. Motivated by the psycholinguistic evidence which demonstrates that intonation changes or pauses would affect the language understanding processes in humans [12], Abney proposes the concept of text chunking as a first step in the full parsing [1]. A typical chunk of a text is defined as consisting of a single content word surrounded by a constellation of function words, matching a fixed template. Church also uses a simple model for finding non-recursive NPs in sequence of POS tags [9]. Turning the sentence chunking as a bracketing problem, Church calculates the probability of inserting both the open and close brackets between POS tags. Each chunking alternative is ranked and the best alternative is selected. In a somewhat similar vein, using transformation-based learning with rule-template referring to neighboring words, POS tags and chunk tags, Ramshaw and Marcus identify essentially the initial portions of non-recursive noun phrases up to the head, including determiners [18]. These chunks are extracted from the Treebank parses, by selecting NPs that contain no nested NPs. While the above approaches have been proposed to recognize common subsequences and to produce some forms of chunked representation of an input sentence, the recognized structures do not include any recursively embedded NPs. As the result, the resultant fragments bear little resemblance to the kind of phrase structures that are normally appeared in linguistics.

While the state of the art in computational linguistics is to make use of the knowledge encoded in Treebank to analyze sentence structures, two major issues have to be clarified beforehand. First, what formalism do we assume to annotate the corpus utterances? Second, what kinds of trees do we extract from the corpus, and how do we recombine them? In this paper, we address the first issue by adopting the Sinica Chinese Treebank [7]. In contrast to the English and Chinese Penn Treebank which takes a straightforward syntactic approach [16, 21], the Information-based Case Grammar (ICG) in Sinica Chinese Treebank stipulates that each lexical entry contains both semantic and syntactic features. The grammar indicates the way that lexical tokens in the sentences are related to each other. That is, grammatical constraints are expressed in terms of linear order of thematic roles and their syntactic and semantic restrictions. This tree structure has the advantage of maintaining phrase structure rules as well as the syntactic and semantic dependency relations. The latest version of Sinica Treebank (v.2.1), released in early 2004, contains about 55,000 trees with 300,000 words. The Treebank contains a compact bundle of syntactic and semantic information, with more than 150 different types of POS and 50 semantic roles.

On the other hand, while it may be too computationally demanding to have a full syntactic and semantic analysis of every sentence in every text, Sima'an addresses the second issue and presents a *Tree-gram* model, a typical example of DOP, which integrates bilexical dependencies, and conditions its substitutions based on the structural relations of the trees that are involved [19]. The basic ideas of the Tree-gram model are to (i) take a corpus of utterances annotated with labeled trees; (ii) decompose every corpus tree into a set of subtrees; (iii) perform parsing as the union of the best possible subtrees based on a stochastic tree substitution grammar. In this research, we

propose an algorithm in extracting shallow language patterns by matching any input Chinese sentence with the trees in the Treebank using an approximate pattern matching technique. Different from the stochastic tree substitution grammar proposed in the Tree-gram model, our approach, characterized by an optimization technique, looks for a transformation with a minimum cost, or called *edit distance*. While the concept of edit distance is commonly found in the conventional pattern matching techniques [13, 20], we take a step further in applying the technique in data-oriented parsing. The detailed discussion of the algorithm is shown as follows.

3 Shallow Language Patterns Extraction Using Attributed String Matching Algorithm

The algorithm is essentially accomplished by applying a series of edit operations to an input sentence to change it to every tree in the Treebank. Every edit operation has been associated with a cost and the total cost of the transformation can be calculated by summing up the costs of all the operations. This edit distance reflects the dissimilarity between the input sentence and the trees. Instead of analyzing the exact Chinese tokens appearing in the sentence, extended attributes of each token in both input sentence and the trees, with their POS and semantic classes, are used. The closely matched tree, i.e., the one with minimum cost or edit distance, is selected and the corresponding phrase structures and semantic role tags delineated in the tree are unified with the input sentence.

Let two given attributed strings A and B denoted as $A = a_1a_2a_3... a_m$ and $B = b_1b_2b_3... b_n$, where are a_i, b_j the ith and jth attributed symbols of A and B respectively. Each attributed symbol represents a primitive of A or B. Generally speaking, to match an attributed string A with another B means to transform or edit the symbols in A into those in B with a minimum-cost sequence of allowable edit operations. In general, the following three types of edit operations are available for attributed symbol transformation.

(a) *Change*: to replace an attributed symbol a_i with another b_j, denoted as $a_i \rightarrow b_j$.
(b) *Insert*: to insert an attributed symbol b_j into an attributed string, denoted as $\lambda \rightarrow b_j$ where λ denotes a null string.
(c) *Delete*: to delete an attributed symbol a_i from an attributed string, denoted as $a_i \rightarrow \lambda$.

[Definition 1]
An *edit sequence* is a sequence of ordered edit operations, $s_1, s_2,... s_p$ where s_i is any of the following three types of edit operations, *Change*, *Insert*, *Delete*.

[Definition 2]
Let R be an arbitrary nonnegative real cost function which defines a cost $R(a_i \rightarrow b_j)$ for each edit operation $a_i \rightarrow b_j$. The cost of an edit sequence $S = s_1, s_2,... s_p$ to be

$$R(S) = \sum_{i=1}^{p} R(s_i) \tag{1}$$

[Definition 3]
For two strings A ad B with length m and n respectively, $D(i, j)$ denotes the edit distance, which is the minimum number of edit operations, needed to transform the first i characters of A into first j characters of B, where $1 \le i \le m$ and $1 \le j \le n$. In other words, if A has m letters and B has n letters, then the edit distance of A and B is precisely the value $D(m, n)$.

The following algorithm has been proposed for computing every edit distances $D(i, j)$ [13].

[Algorithm A]
$D(0, 0) := 0$;
for $i := 1$ **to** m **do** $D(i, 0) := D(i\text{-}1, 0) + R(a_i \rightarrow \lambda)$;
for $j := 1$ **to** n **do** $D(0, j) := D(0, j\text{-}1) + R(\lambda \rightarrow b_j)$;
for $i := 1$ **to** m **do**
    for $j := 1$ **to** n **do**
    begin
        $m1 := D(i, j\text{-}1) + R(\lambda \rightarrow b_j)$;
        $m2 := D(i\text{-}1, j) + R(a_i \rightarrow \lambda)$;
        $m3 := D(i\text{-}1, j\text{-}1) + R(a_i \rightarrow b_j)$;
        $D(i, j) := \min(m1, m2, m3)$;
    end

Our attributed string matching in case role annotation is to make use of the algorithm above and modify the cost function $R(.)$ for various edit operations. In our approach, each Chinese token has two attributes, i.e., Part-Of-Speech (*POS*) and Semantic Class (*SC*). Let S be an input sentence and the T be a tree in the Sinica Treebank, s_i and t_j be two tokens in S and T with attribute $\langle POS_i, SC_i \rangle$ and $\langle POS_j, SC_j \rangle$ respectively. We define the cost function for a *change* operation $s_i \rightarrow t_j$ to be

$$R(s_i \rightarrow t_j) = u(POS_i, POS_j) + v(SC_i, SC_j) \tag{2}$$

where $u(POS_i, POS_j)$ defines the partial cost due to the difference between the POS of the tokens. The POS tags from the Chinese Knowledge Information Processing Group (CKIP) of Academia Sinica are employed [8]. The tags involve 46 different types of POS which can further refine into more than 150 subtypes. In order to figure out the cost function $u(\cdot,\cdot)$, in our system, all the POS tags are organized into a tree structure using XML with an associated hard-coded cost function. Figure 1 shows a fragment of XML of the nouns (`Na`) which is divided into in-collective (`Nae`) and collective (`Na1`) nouns which are then divided ultimately into in-collective concrete uncountable nouns (`Naa`), in-collective concrete countable nouns (`Nab`), in-collective abstract countable nouns (`Nac`), in-collective abstract uncountable nouns (`Nad`). The cost function $u(\cdot,\cdot)$ will reflect the difference based on the tag `Toll` encoded in the XML as shown in Figure 1. For example, the cost for changing a word having POS from `Naa` to `Nab`,

u(`Naa, Nab`) = `Toll`(`Naa`→`Na11`) + `Toll`(`Na11`→`Nab`) = 1 + 1 = 2

Similarly,

$$u(\mathrm{Naa}, \mathrm{Naea}) = \mathrm{Toll}(\mathrm{Naa}\rightarrow\mathrm{Na11})+\mathrm{Toll}(\mathrm{Na11}\rightarrow\mathrm{Na1})+\mathrm{Toll}(\mathrm{Na1}\rightarrow\mathrm{Na})+\mathrm{Toll}(\mathrm{Na}\rightarrow\mathrm{Nae})+\mathrm{Toll}(\mathrm{Nae}\rightarrow\mathrm{Naea})= 7$$

```
<Na Toll="4" Level="2">
  <Na1 Toll="2" Level="3">
     <Na11 Toll="1" Level="4">
       <Naa Toll="1" Level="5" />
       <Nab Toll="1" Level="5" />
     </Na11>
     <Na12 Toll="1" Level="4">
       <Nac Toll="1" Level="5" />
       <Nad Toll="1" Level="5" />
     </Na12>
   </Na1>
   <Nae Toll="2" Level="3">
     <Naea Toll="1" Level="4" />
     <Naeb Toll="1" Level="4" />
   </Nae>
</Na>
```

Fig. 1. Tree structure of Nouns (Na) based on the CKIP Academia Sinica

The function $u(\cdot,\cdot)$ partially indicates the alignment of the syntactic structure of the input sentence and the sentence appeared in the Treebank. The second term in equation (2) defines the other partial cost due to the semantic differences. In our approach, the lexical tokens in the both sentences are identified using a lexical source similar to the Roget's Thesaurus. The lexical source is a bilingual thesaurus with an *is-a* hierarchy. An *is-a* hierarchy can be viewed as a directed acyclic graph with a single root. Based on the *is-a* hierarchy in the thesaurus, we define conceptual distance *d* between two notional words by their shortest path lengths [17].

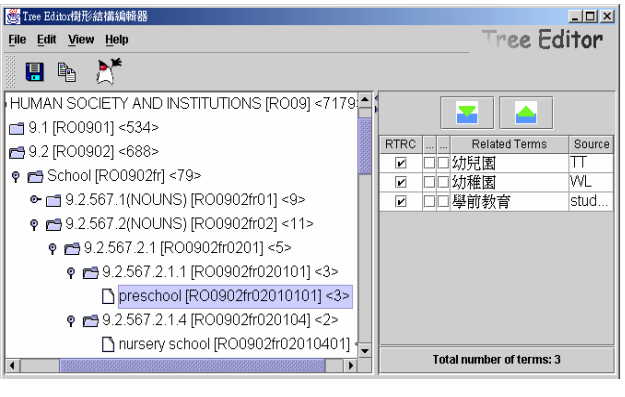

Fig. 2. *is-a* hierarchy in the bilingual thesaurus

Figure 2 shows one of our *is-a* hierarchies in our bilingual thesaurus using our Tree Editor. While the upward links correspond to generalization, the specialization is represented in the downward links. For example, the upward link from 幼稚園 (pre-school) to 學校 (school) indicates that 學校 (school) is more general than 幼稚園 (preschool) and the lexical tokens in the same terminal subclass, such as 幼稚園 (pre-school), 幼兒園 (nursery school), or 幼兒中心 (day-care centre), are of the same meaning. The hierarchies demonstrated in the thesaurus are based on the idea that linguists classify lexical items in terms of similarities and differences. They are used to structure or rank lexical items from more general to the more special. Given two tokens t_1 and t_2 in an *is-a* hierarchy of the thesaurus, the distance d between the items is defined as follows:

$$d(t_1, t_2) = \text{minimal number of } \textit{is-a} \text{ relationships in the shortest path between } t_1 \text{ and } t_2 \tag{3}$$

The shortest path lengths in *is-a* hierarchies are calculated. Initially, a search fans out through the *is-a* relationships from the original two nodes to all nodes pointed to by the originals, until a point of intersection is found. The paths from the original two nodes are concatenated to form a continuous path, which must be a shortest path between the originals. The number of links in the shortest path is counted. Since $d(t_1, t_2)$ is positive and symmetric, $d(t_1, t_2)$ is a metric which means (i) $d(t_1, t_1) = 0$; (ii) $d(t_1, t_2) = d(t_2, t_1)$; (iii) $d(t_1, t_2) + d(t_2, t_3) \geq d(t_1, t_3)$. At the same time, the semantic similarity measure between the items is defined by:

$$v(t_i, t_j) := \begin{cases} d(t_i, t_j) & \text{if } d(t_i, t_j) \leq d_{max} \\ MaxInt & \text{otherwise} \end{cases} \tag{4}$$

where d_{max} is proportional to the number of lexical items in the system and *MaxInt* is a maximum integer of the system. This semantic similarity measure defines the degree of relatedness between tokens. Obviously, strong degree of relatedness exists between the lexical tokens under the same nodes. For the cost of the insert and delete operations, we make use the concept of *collocation* which measures how likely two tokens are to co-occur in a window of text. To better distinguish statistics based ratios, work in this area is often presented in terms of the mutual information, which is defined as

$$MI(t_{j-1}, t_j) = \log_2 \frac{P(t_{j-1}, t_j)}{P(t_{j-1}) \times P(t_j)} \tag{5}$$

where t_{j-1} and t_j are two adjacent tokens. While $P(x, y)$ is the probability of observing x and y together, $P(x)$ and $P(y)$ are the probabilities of observing x and y anywhere in the text, whether individually or in conjunction. Note that tokens that have no association with each other and co-occur together according to chance will have a mutual information number close to zero. This leads to the cost function for insertion and deletion shown in equation (6) and (7) respectively.

$$R(\lambda \rightarrow t_j) = \begin{cases} K \times |z| & \text{if } 0 \geq z > \varepsilon \\ MaxInt & \text{otherwise} \end{cases} \tag{6}$$

where $z = \min\{MI(t_{j-1}, t_j), MI(t_j, t_{j+1})\}$

$$R(t_j \to \lambda) = \begin{cases} L \times \left| MI(t_{j-1}, t_{j+1}) \right| & \text{if } 0 \geq MI(t_{j-1}, t_{j+1}) > \varepsilon \\ MaxInt & \text{otherwise} \end{cases} \tag{7}$$

where K, L, ε are three constants relied on the size of the corpus.

Obviously, the insertion operation will be penalized if the co-occurrence between the newly inserted token and its neighbors is low. Similarly, the deletion operation is most likely to happen if there is a high co-occurrence between the adjacent pairs after the deletion. Using the above cost functions for the three edit operations, the tree in the Treebank with minimum cost is identified to best approximation of the input sentence S and its relevant case roles tags will be adopted. Shallow language patterns, with all the recursively embedded structures, are then extracted based on the case role tags appeared in the Treebank. Experimental results and an illustration of the patterns extracted are shown in the following section.

4 Experimental Results

We have implemented the system using Java JDK1.4.2 under Sun Microsystems. The whole system development is designed under Unified Modeling Language (UML) using Rational Rose. To show the efficiency of the proposed algorithm, a series of experiments are performed.

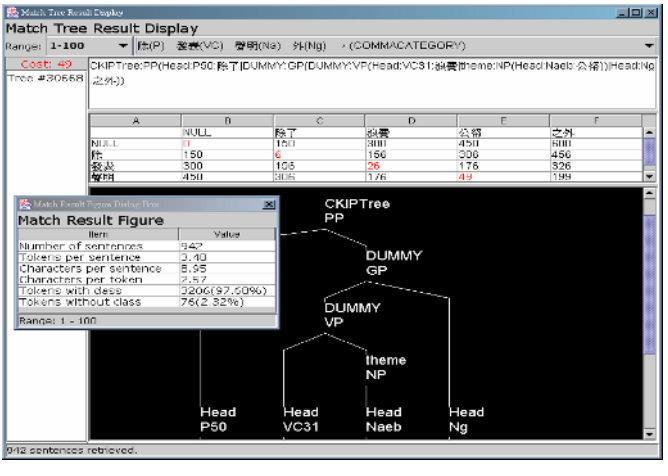

Fig. 3. Graphical User Interface (GUI) in the shallow language patterns extraction

In our experiment, for every input sentence, the best matching tree with minimum edit distance in the Treebank is calculated as shown in Algorithm A. The Information Case Grammar (ICG) of the best matching tree in the Treebank will be adopted. Figure 3 shows the graphical user interface which includes the cost matrix generated and

the corresponding ICG structure of the input sentence. We have tested the system with 5,000 input sentences. The detailed results are shown in Table 1. The average sentence length is around 10.5 characters per sentence. In order to let the readers to visualize the relevance of the edit distance with the underlying tree structure, Figures 4 and 5 show two sentences clearly with edit distance equal to 15 and 64 respectively. The sentence

議員商討總統出兵一事 (S1)

(in English, *The senators discuss the issue on sending troops initiated by the president*)

has a small edit distance, equal to 15, with the tree shown in Figure 4.

Table 1. Analysis of 5,000 sentences in the experiment

Edit distance	# of sentences	Average # of tokens	Average edit distance	% of sentences having incomplete semantic classes
0-25	336	5.24	21.06	2.32
0-50	1556	6.15	34.42	9.01
0-75	2841	6.67	46.94	11.08
0-100	5000	6.62	65.94	11.93

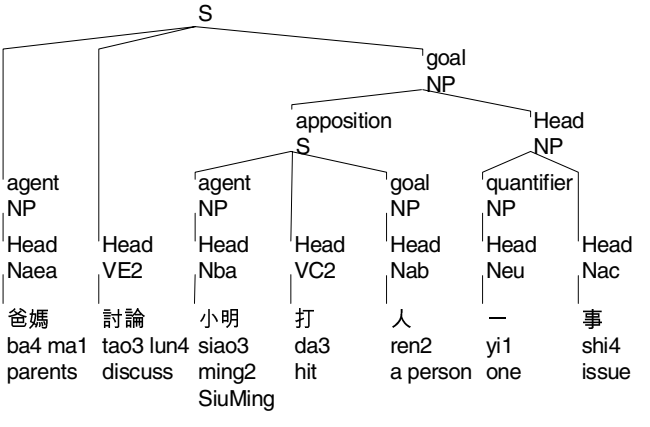

Fig. 4. Tree in the Treebank which closely matches, edit distance equal to 15, with the input sentence shown in (S1)

The sentence is then chunked into phrases, ⟨議員⟩ (*The senators*), ⟨商討⟩ (*discuss*), and ⟨總統出兵一事⟩ (*the issue on sending troops initiated by the president*), which are further tagged with *agent*, *act*, and *goal* respectively by taking the advantage of

annotation in the Treebank. Certainly, the phrase 〈總統出兵一事〉 (*the issue on sending troops initiated by the president*) can be further chunked into more details 〈總統出兵〉 (*sending troops initiated by the president*), 〈一事〉 (*the issue*).

Table 2. Language patterns extracted from the input sentence (S1)

@SP[	議員商討總統出兵一事	*The senators discuss the issue on sending troops initiated by the president*
Agent	〈議員〉	*The senators*
Act	〈商討〉	*discuss*
Goal	@AP[〈總統出兵〉]AP, @NP[〈一事〉]NP	@AP[(*sending troops initiated by the president*)]AP, @NP[(*the issue*)]NP
]SP		
@AP[	總統出兵	*sending troops initiated by the president*
Agent	總統	*the president*
Act	出	*sending*
Goal	兵	*troops*
]AP		

This chunking not only provides the basic semantic tag for each constituent, it also reflects the language patterns of the input sentence. The corresponding pattern extracted is shown in Table 2. As shown in Table 2, the language patterns extracted are indicated by the square brackets together with the explicit semantic tags. While the sentence pattern is marked with `@SP[…]`, the embedded phrases are marked by different tags, such `@AP[…]` for apposition phrase, or `@PP[…]` for position phrase. Similarly, in Figure5, the upper sentence is coming from the Treebank *T* while the lower one represents the input sentences *S*. Due to the similarity, in terms of the edit distance, of the matched pair, the syntactic structure of the sentence from the Treebank is transplanted to the input sentence. As a result, each token in the input sentence will inherit the associated roles from the target sentence. For example, the Chinese sentence shown in Figure 5,

可惜國家財政萬分困難
(in English: *Unfortunately, the national budget is so tight*) (S2)

The sentence is chunked into 〈可惜〉 (*unfortunately*), and 〈國家財政萬分困難〉 (*the national budget is so tight*) which is further chunked into 〈國家財政〉 (*national budget*), 〈萬分〉 (*so*), 〈困難〉 (*tight*). At the same time, 〈國家財政萬分困難〉 (*national budget is so tight*), 〈國家財政〉 (*national budget*), and 〈萬分〉 (*so*) are annotated as the *goal*, *main theme* and *degree* of Sentence S2 respectively.

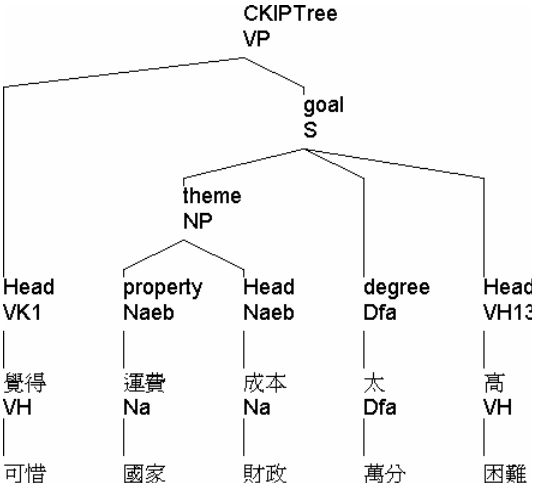

Fig. 5. Sentence with edit distance equal to 64

As with other text analysis, the effectiveness of the system appears to be dictated by recall and precision parameters where recall (R) is a percentage of how many correct case roles can be identified while precision (P) is the percentage of case roles, tackled by the system, which are actually correct. In addition, a common parameter F is used as a single-figure measure of performance as in follows,

$$F = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R} \tag{8}$$

We set $\beta = 1$ to give no special preference to either recall or precision. The recall, precision and F value are 0.84, 0.92 and 0.878 respectively. It is worthwhile to mention that, as shown in Table 1, more than 500 sentences have incomplete semantic classes which mainly come from proper nouns, unknown words, proverbs or even short phrases. While the boundaries between words and phrases in Chinese are not easy to differentiate, the performance, due to the coverage of semantic classes in our thesaurus, does not deteriorate much in our system. This tolerance ability provides the graceful degradation in our case role annotation. While other systems are brittle and working only in all-or-none basis, the robustness of our system is guaranteed even though more than 10% of tokens having their *SC* tags missing.

5 Conclusion

In this paper, we have illustrated a shallow technique in which language patterns are extracted in forms of chunks of phrases or words. The chunks are further tagged with case roles. Although the technique does not require a full syntactic parse to pursue semantic analysis, the recursively embedded phrases can also be identified without pain. While we have demonstrated that it is much easier to work out the approximate

parse of individual actual sentences than to try to determine what all possible manifestations of a certain rule or grammatical constructs are, our linguistic sequence analysis is inspired by the research into bio-molecular sequences, such as DNA and RNA in protein. Bio-molecular scientists believe that *high sequence similarity usually implies significant function or structural similarity*. It is characteristic of biological systems that objects have a certain form that has arisen by evolution from related objects of similar but not identical form. This *sequence-to-structure* mapping is a tractable, though partly heuristic, way to search for functional or structural universality in biological systems. With the support from the results as shown in this paper, we conjecture this *sequence-to-structure* phenomenon appears in our sentences. The sentence sequence encodes and reflects the more complex linguistic structures and mechanisms described by linguists. While our system does not claim to deal with all aspects of language, we suggest an alternative, but plausible, approach to handle the real corpus.

Acknowledgement

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4438/04H).

References

[1] Abney, S. (1991). Parsing by chunks. In Berwick, R., Abney, S. & Tenny, C. (Eds.), *Principle-Based Parsing*. Kluwer Academic.

[2] Appelt, D. E., Hobbs, J. R., Bear,J., Israel, D., Tyson,M. (1993). FASTUS: a finite-state processor for information extraction from real-world text. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, vol. 2, 1172-1178.

[3] Bod, R., Scha, R., & Sima'an, K. (2003). *Data-Oriented Parsing*. Stanford: California, CSLI.

[4] Chan, S.W.K. (2004). Extraction of Textual Salient Patterns: Synergy between Lexical Cohesion and Contextual Coherence. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34, 2, 205-218.

[5] Chan, S.W.K., & Franklin, J. (2003). Dynamic context generation for natural language understanding: A multifaceted knowledge approach. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* 33, 1, 23-41.

[6] Chao, Y.-R. (1968). *A Grammar of Spoken Chinese*. University of California Press.

[7] Chen, F.-Y., Tsai, P.-F., Chen, K.-J., & Huang, C.-R. (2000). Sinica Treebank. [in Chinese] *Computational Linguistics and Chinese Language Processing*, 4, 2, 87-103.

[8] Chen, K.-J., Huang, C.-R., Chang, L.-P., & Hsu. H.-L. (1996). Sinica Corpus: Design Methodology for Balanced Corpora. *Proceedings of the 11th Pacific Asia Conference on Language, Information, and Computation (PACLIC II)*, Seoul Korea, 167-176.

[9] Church, K. (1988). A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of Second Conference on Applied Natural Language Processing*, Austin, Texas.

[10] Fillmore, C.J. (1968). The case for case. In E. Bach & R.T. Harms (Eds.), *Universals in Linguistic Theory*, 1-90. Holt, Rinehart & Winston.

[11] Gaizauskas, R., & Wilks, Y. (1998). Information extraction: Beyond document retrieval. *Journal of Documentation*, 54, 1, 70-105.

[12] Gee, J., & Grosjean, F. (1983). Performance structures: A psycholinguistic and linguistic appraisal. *Cognitive Psychology*, 15, 4, 411-458.

[13] Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press.

[14] Her, O. S. (1990). *Grammatical functions and verb subcategorization in Mandarin Chinese*. The Crane publishing Co.

[15] Li, Y. C. (1971). *An investigation of Case in Chinese Grammar*. Set On Hall University Press.

[16] Marcus, M., Santorini, B. and Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19, 2, 313-330.

[17] Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989) Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19, 1, 17-30.

[18] Ramshaw, L. A., & Marcus, M.P. (1995). Text chunking using transformation-based learning. *Proceedings of the Third Workshop on Very Large Corpora*, 82-94.

[19] Sima'an, K. (2000). Tree-gram parsing: lexical dependencies and structural relations. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 53-60, Hong Kong.

[20] Tsay, Y.T., & Tsai, W.H. (1989). Model-guided attributed string matching by split-and-merge for shape recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 3, 2, 159-179.

[21] Xia, F., Palmer, M., Xue, N., Okurowski, M.E., Kovarik, J., Chiou, F.-D., Huang, S., Kroch, T., & Marcus, M. (2000). Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. *Proceedings of the second International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.