An Edit Distance Approach to Shallow Semantic Labeling

Samuel W.K. Chan

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

Abstract. This paper proposes a model of semantic labeling based on the edit distance. The dynamic programming approach stresses on a non-exact string matching technique that takes full advantage of the underlying grammatical structure of 65,000 parse trees in a Treebank. Both part-of-speech and lexical similarity serve to identify the possible semantic labels, without miring into a pure linguistic analysis. The model described has been implemented. We also analyze the tradeoffs between the part-of-speech and lexical similarity in the semantic labeling. Experimental results for recognizing various labels in 10,000 sentences are used to justify its significances.

1 Introduction

Automatic information extraction has received a great deal of attention in the latest development of information retrieval. While a plethora of issues relating to questions of accuracy and efficiency have been thoroughly discussed, the problem of extracting meaning from natural language has scarcely been addressed. When the size and quantity of documents available on the Internet are considered, the demand for a highly efficient system that identifies the semantic meaning is clear. Case frame is one of the most important structures that are used to represent the meaning of sentences (Fillmore, 1968). One could consider a case frame to be a special, or distinguishing, form of knowledge structure about sentences. Although several criteria for recognizing 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 natural language processing (NLP) do not provide any hints on how to map input sentences into case frames automatically. As a result, both the efficiency and robustness of the techniques used in information extraction is highly in doubt when they are applied to real world applications.

Any high level language understanding process, such as semantic labeling, must involve chunking sentences into segments. Motivated by the psycholinguistic evidence which demonstrates that intonation changes or pauses would affect the language understanding processes in humans, Abney (1991) proposes the concept of text chunking as a first step in the full parsing. 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 uses a simple model for finding base non-recursive NPs in sequence of POS tags (Church, 1988). Turning sentence chunking into 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. Using transformation-based learning with rule-template referring to neighboring words, POS tags and chunk tags, Ramshaw & Marcus (1995) identify essentially the initial portions of non-recursive noun phrases up to the head, including determiners. 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 normally appear in our languages.

In this research, we propose a mechanism in shallow semantic labeling as well as sentence chunking by matching any input sentence with the trees in a Treebank through a two-phase feature-enhanced string matching. The objective of this research is twofold. First, a shallow but effective sentence chunking process is developed. The process is to extract all the phrases from the input sentences, without being bogged down into deep semantic parsing and understanding. Second, a novel semantic labeling technique that is based on the syntactic and semantic tags of the latest Treebank is being constructed (CKIP, 2004). One of our primary goals in this research is to design a shallow but robust mechanism which can annotate sentences using a set of semantic labels. The annotation will provide piecemeal the underlying semantic labels of the sentence. The organization of the paper is as follows. In our approach, each word in sentences has two attributes, i.e. part-of-speech (POS) and semantic classes (SC). Any input sentence is first transformed into a feature-enhanced string. A two-phase feature-enhanced string matching algorithm which is based on the edit distance is devised. Section 2 shows how the algorithm can be applied in the semantic labeling using 65,000 parse trees in a Treebank. The system has already been implemented using Java language. In order to demonstrate the capability of our system, an experiment with 10,000 sentences is conducted. A detailed evaluation is explained in Section 3 followed by a conclusion.

2 Two-Phase Feature-Enhanced String Matching Algorithm

In this section, we will first outline the concepts of edit operations which are essential components of our feature-enhanced string matching algorithm. The two-phase shallow semantic labeling will be discussed thoroughly in Section 2.2.

2.1 Edit Operations

Our 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 cost reflects the dissimilarity between the input sentence and the trees. Instead of analyzing the exact words in the sentence, extended attributes of each word 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 called *edit distance*, is selected and the corresponding phrase structures and semantic labels delineated in the tree are unified with the input sentence.

Let two given feature-enhanced strings A and B denoted as $A = a_1a_2a_3\ldots a_m$ and $B = b_1b_2b_3\ldots b_n$, 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 a feature-enhanced 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 a feature-enhanced string, denoted as $\lambda \rightarrow b_j$ where λ denotes a null string.

(c) *Delete*: to delete an attributed symbol a_i from a feature-enhanced string, denoted as $a_i \rightarrow \lambda$.

[Definition 1]

An *edit sequence* is a sequence of ordered edit operations, $s_1, s_2, \ldots s_p$ 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 $S = s_1, s_2, \ldots s_p$ to be

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

[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 $D(m, n)$. Wagner & Fischer (1974) had proposed the following algorithm for computing every edit distances $D(i, j)$.

```
[Algorithm]
D(0, 0) := 0;
for i := 1 to len(A) do D(i, 0):=D(i-1, 0)+R(a_i→λ);
for j := 1 to len(B) do D(0, j):=D(0, j-1)+R(λ→b_j);
for i := 1 to len(A) do
    for j := 1 to len(B) do
    begin
        m1 := D(i, j-1) + R(λ→b_j);
        m2 := D(i-1, j) + R(a_i→λ);
        m3 := D(i-1, j-1) + R(a_i→b_j);
        D(i, j) := min (m1, m2, m3);
    end
```

Our feature-enhanced string matching in shallow semantic labeling is to make use of the algorithm above and modify the cost function $R(.)$ for various edit operations.

2.2 Shallow Semantic Labeling as Two-Phase Feature-Enhanced String Matching

Our labeling is defined as a two-phase feature-enhanced string matching using the edit operations. For every input sentence, a coarse-grained syntactic matching is conducted in our first phase of matching. The matching relies on a set of coarse-grained but global part-of-speech (POS) tags. The major objective of this phase is to shortlist all the potential trees among 65,000 parse trees in the CKIP Treebank, which are relevant to the input sentence, without getting bogged down into computational complexity with other linguistic details. The second phase of the matching is followed to compute the dissimilarity measure between the input sentence and every short-listed candidate that is identified in the first phase. Detailed POS and semantic class (SC) tags will be employed. As a result, a candidate tree which has the minimum dissimilarity with the input sentence will be identified. The underlying semantic labels and phrases of the candidate tree are used to determine the shallow language patterns of the input sentence. The details of the two-phase matching are explained in the following.

2.2.1 Coarse-Grained Syntactic Matching

In the first phase of matching, each word is represented by its corresponding POS. Let S be an input sentence and the T be a tree in a Treebank, s_i and t_j be two tokens in S and T with attribute POS_i and POS_j respectively. We define the cost function for the *change* operation $s_i \rightarrow t_j$ to be

$$R(s_i \rightarrow t_j) = u(POS_i, POS_j) \quad \textbf{(2)}$$

where $u(POS_i, POS_j)$ defines the cost due to the difference between the POS of the two tokens. The POS tags from the Chinese Knowledge Information Processing Group (CKIP) of Academia Sinica are employed (Chen *et al.*, 1996). The tags are subdivided into 46 major POS classes which are further refined into more than 150 subtypes. However, in this coarse-grained matching, only the major POS classes will be considered. To figure out the cost function $u(\cdot,\cdot)$ in the coarse-grained matching, all the major POS tags are organized into a hierarchical structure with an associated hard-coded cost function. Figure 1 shows the structure of notional words and describes the relative distances between the adjectives (`A`), verbs (`V`), status-verbs (`VH`), measure-words (`Nf`), nouns (`N`), position-words (`Ng`), time-words (`Nd`) and place-words (`Nc`). All notional words have definite meanings in the language. The cost function is based on their interchangeability, the degree of flexibility in placement in the syntax, and the similarity of their acceptable modifiers. For example, Chinese verbs and adjectives share a lot of common features syntactically, i.e. both can be predicates or modified by adverbs and the word, *not*. All these features fail to appear in nouns. The abbreviations in bracket indicate the original POS tags marked by the CKIP. The corresponding tree structure of the XML is shown in Figure 2. The cost function $u(\cdot,\cdot)$ reflects the difference based on the tag `toll` encoded in the XML as shown in Figure 1. The function also indicates the degree of alignment between the syntactic structure of the input sentence and the trees in the Treebank. Although two feature-enhanced strings with the same POS sequence do not imply they will share the same syntactic structure, this coarse-grained syntactic matching shortlists the potential trees by imposing a necessary, even not sufficient, constraint on its syntactic structure and limits the potential search space in the subsequent stage of semantic matching.

```
<Head toll="5">
  <NodeB toll="2">
    <NodeC toll="2">
      <Adjective toll="5"/>
      <Verb toll="5"/>
    </NodeC>
    <Status-Verb toll="7"/>
  </NodeB>
  <NodeD toll="2">
    <Measure-Word toll="7"/>
    <NodeE toll="2">
      <Noun toll="5"/>
      <NodeF toll="2">
        <Position-word toll="3"/>
        <NodeG toll="1">
          <Time-word toll="2"/>
        ...
</Head>
```

Fig. 1. XML illustrating the relative distances between 8 different types of POS

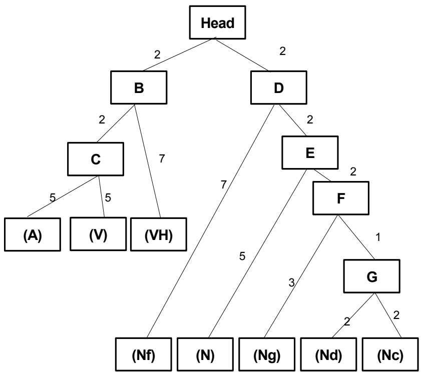

Fig. 2. Corresponding tree structure of the XML shown in Fig.1

2.2.2 Computation of Semantic Dissimilarity

What this second phase matching basically does is to make a detailed comparison between the input sentence and the short-listed trees in its earlier stage. In this phase, each Chinese token has two attributes, i.e. a detailed part-of-speech (*POS*) and semantic class (*SC*). Similar to the approach in Section 2.2.1, we define the cost function for the *change* operation $s_i \rightarrow t_j$ to be

$$R(s_i \rightarrow t_j) = f\big(u(POS_i, POS_j), v(SC_i, SC_j)\big) \qquad \textbf{(3)}$$

where the function *f* is the dissimilarity function relied on two major components. The first component $u(POS_i, POS_j)$ defines the partial cost due to the difference between the detailed POS of the words. The detailed POS tags are organized in XML format, similar to the approach demonstrated in Figure 1. For example, the further breakdown of the nouns (`Na`) which are divided into in-collective (`Nae`) and collective (`Nal`) nouns. The collective nouns are then subdivided into in-collective concrete uncountable nouns (`Naa`), in-collective concrete countable nouns (`Nab`), in-collective abstract countable nouns (`Nac`), and in-collective abstract uncountable nouns (`Nad`). The second term in Eqn. (3) defines another partial cost due to the semantic differences. In our approach, the words in the input sentences and the trees are identified using a bilingual thesaurus similar to the Roget's Thesaurus. The *is-a* hierarchy in the bilingual thesaurus, shown the underlying ontology, can be viewed as a directed acyclic graph with a single root. While the upward links correspond to generalization, the specialization is represented in the downward links. The ontology demonstrated in the thesaurus is 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. Based on the *is-a* hierarchy in the thesaurus, we define the conceptual distance *d* between two notional words by their shortest path lengths.

Given two words t_1 and t_2 in an *is-a* hierarchy of the thesaurus, the semantic distance d between the tokens is defined as follows:

$d(t_1, t_2) =$ minimal number of *is-a* relationships in the shortest path between t_1 and t_2 **(4)**

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:

$$v(t_i, t_j) := \begin{cases} d(t_i,\ t_j) & \text{if} \quad d(t_i,\ t_j) \leq d_{max} \\ MaxInt & \text{otherwise} \end{cases} \qquad \textbf{(5)}$$

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 the words. Obviously, strong degree of relatedness exists between the lexical tokens under the same nodes. On the other hand, for the cost of the *insert* and *delete* operations, we make use the concept of *collocation* that measures how likely two words 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 pointwise mutual information (*MI*), which is defined as

$$MI(t_{j-1}, t_j) = \log \frac{P(t_{j-1}, t_j)}{P(t_{j-1}) \times P(t_j)} \qquad \textbf{(6)}$$

where t_{j-1} and t_j are two adjacent words (Manning & Schütze, 1999). While $P(t_{j-1}, t_j)$ is the probability of observing t_{j-1} and t_j together, $P(t_{j-1})$ and $P(t_j)$ are the probabilities of observing t_{j-1} and t_j 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 *MI* value close to zero. This leads to the cost function for insertion and deletion shown in Eqns. (7) and (8) respectively.

$$R(\lambda \to t_j) = \begin{cases} k \times e^{-z} & \text{if } z > \varepsilon > 0 \\ MaxInt & \text{otherwise} \end{cases} \qquad \textbf{(7)}$$

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

$$R(t_j \to \lambda) = \begin{cases} l \times e^{-MI(t_{j-1}, t_{j+1})} & \text{if } MI(t_{j-1}, t_{j+1}) > \varepsilon > 0 \\ MaxInt & \text{otherwise} \end{cases} \qquad \textbf{(8)}$$

where k, l, ε are three constants relied on the size of the active corpus.

Obviously, the insertion operation will be penalized if the co-occurrence between the newly inserted word 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 types of edit operations, the tree in the Treebank with minimum cost is being chosen to be the best approximation of the input sentence and its associated semantic labels will be adopted. Shallow language patterns are then extracted based on the recursive structures and semantic labels appeared in the Treebank. The experimental results of the semantic labeling are shown in the section below.

3 Experimental Results

As mentioned in Eqn. (3), several approaches have been used to define the dissimilarity function *f* by combining the semantic differences and the detailed POS tags in our second phase feature-enhanced string matching. In our evaluations, five different types of dissimilarity function *f* are applied. They are

(i) $f_1(u, v) = u(POS_i, POS_j)$

(ii) $f_2(u, v) = v(SC_i, SC_j)$

(iii) $f_3(u, v) = u(POS_i, POS_j) + v(SC_i, SC_j)$

(iv) $f_4(u, v) = \max\,(u(POS_i, POS_j)\,,\ v(SC_i, SC_j))$

Dissimilarity function $f_1(u, v)$ provides a detailed version of our coarse-grained syntactic matching. Detailed POS tags are used as the dissimilarity measure in the labeling. Similarly, $f_2(u, v)$ considers only the semantic class of the words. The other two combine both syntactic and semantic features in defining the dissimilarity measures. We have tested our shallow semantic labeling with 10,000 sentences with the Treebank. Since this research is concerning with shallow semantic labeling, we have no incentive to match the trees/subtrees in the Treebank with very complicated structures. The average sentence length is around 13.7 characters per sentence.

Table 1. Sentence analysis in the experiment. Edit distance is defined as a minimum cost in transforming the input sentence with the closest sentence pattern in the Treebank.

Dissimilarity function *f*	Range of Edit distance	% of sentences	Average edit distance	% of sentences w/ incomplete info.
$f_1(u, v)$	0-25	13.9	19.2	2.1
	26-50	16.3	40.5	2.1
	51-75	19.7	63.6	4.7
$f_2(u, v)$	0-25	11.3	19.3	2.3
	26-50	15.6	41.4	3.8
	51-75	17.7	65.2	6.4
$f_3(u, v)$	0-25	24.1	17.9	2.2
	26-50	31.6	38.2	4.1
	51-75	22.7	62.3	6.9
$f_4(u, v)$	0-25	20.5	19.6	1.9
	26-50	22.4	40.9	3.8
	51-75	26.9	58.2	7.3

Table 1 summarizes the results of our system evaluation. The third and fourth columns in the table are number of sentences in each range of edit distance and their average edit distances. The edit distance is defined as a minimum cost in transforming the input sentence with the closest sentence pattern in the Treebank. In other words, the smaller the distance, the higher similarity they have. If it is considered as a good match where the edit distances are equal to or less than 50, then it can be observed, in Table 1, that the dissimilarity functions f_3 and f_4 all produce higher percentage of sentences with lower edit distance. This reflects both the information from syntactic tags and semantic classes provide useful clues in our shallow semantic labeling. Our experiments are not conducted with perfect information. It is worthwhile to mention that, as shown in right-most column of Table 1, more than 530 sentences have incomplete information which mainly comes from proper nouns, or out-of-vocabulary (OOV) words. Both of them have neither defined POS nor semantic class. All these information will be annotated with a default value which will certainly induce errors in our labeling. While it is inevitable to have OOV words in any real corpus, the performance, due to the coverage of POS and semantic classes, does not deteriorate much in our system. The labeling is still feasible over the sentences with OOV words. This tolerance ability provides the graceful degradation in our shallow semantic labeling. While other systems are brittle and working only in all-or-none basis, the robustness of our system is guaranteed. At the same time, while real text tends to have grammatical mistakes and error-prone, these problems can be tackled with an acceptable tolerance in our system. In our second evaluation, we have tested our algorithm in recognizing several major semantic labels that appear in our sentences. The semantic labels include `theme`, `goal`, `property`, `range`, `agent`, and `predication`. 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 labels can be identified while precision (P) is the percentage of labels, tackled by our system, which are actually correct. In addition, a common parameter F is used as a single-figure measure of performance which combines recall (R) and precision (P) as in follows,

$$F = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R} \quad \textbf{(9)}$$

We set $\beta = 1$ to give no special preference to either recall or precision. The recall, precision and F-score for the semantic labels in dissimilarity function f_3 are shown in Table 2.

Table 2. Evaluation of some semantic labels in the dissimilarity function f_3. An elementary subtree spans only on a sequence of words while a derivation subtree contains at least one branch of elementary subtree.

	Elementary Subtree			**Derivation Subtree**		
Semantic Label	***R***	***P***	***F*-score**	***R***	***P***	***F*-score**
`theme`	0.79	0.82	0.805	0.88	0.85	0.865
`goal`	0.80	0.79	0.795	0.78	0.76	0.770
`property`	0.89	0.91	0.900	0.78	0.83	0.804
`range`	0.94	0.92	0.930	0.93	0.91	0.920
`agent`	0.92	0.87	0.894	0.92	0.85	0.884
`predication`	0.76	0.81	0.784	0.80	0.78	0.790
`all labels`	0.81	0.83	0.821	0.78	0.81	0.801

As shown in the last row in Table 2, the precision and recall of all semantic labels are calculated by considering all the semantic labels that appear in the sentences, rather than by averaging the measures for individual semantic labels. It is worth noting that the greatest differences in performance are the recall while the precision remains relatively steady in most semantic labels. One possible explanation is that the low recall rates in some labels are due to less complete coverage of linguistic phenomena. In addition, we define an elementary subtree that spans only on a sequence of words, as well as a derivation subtree that contains at least one branch of elementary subtree. It may be expected the F-score of the derivation subtrees will be much worse than its counterpart, however, Table 2 shows surprisingly the differences in the overall accuracy in two main types of subtrees are not significant. An explanation is that we have approached chunking as well as assigning the most salient semantic label to the chunks based on the POS and semantic tags. Even though there may be some misclassification in the terminal nodes, this will not hinder the system to tag the semantic labels in the longer chunks. In other words, the longer chunks are less error prone in our semantic labeling. This shallow semantic labeling technique produces an output that abstract away the details but retains the core semantic structure of the actual sentence. To further evaluate our system, we compare its performance with other approaches, even though the relevant experimental results in Chinese language are not easy to obtain in the literature. We compare our system with those systems participated in the CoNLL, the Conference on Natural Language Learning. The training data used in the CoNLL-2005 consists of sections from the Wall Street Journal with information on predicate-argument structures (Carreras & Màrquez, 2005). Table 3 shows the performance comparison with the top five participating systems in the conference. Even though our system cannot compare head-to-head with their models since their training and test data is totally disparate with our experiment, this paper has suggested one of the alternatives in semantic labeling, with *F*-score over 0.8 shown in Table 2. Certainly, further explorations are needed to improve the system performance.

Table 3. Overall precision (*P*), recall (*R*) and *F*-scores of the top five systems in CoNLL-2005. While the second column, ML-Method, illustrates the related machine learning techniques, the third column shows whether they involve any post-processing method.

System	**ML-Method**	**Post**	***P***	***R***	***F***
Punyakanok *et al* (2005)	Winnow-based Network	No	0.823	0.768	0.794
Haghighi *et al* (2005)	Maximum Entropy	No	0.795	0.774	0.785
Marquez *et al* (2005)	AdaBoost Algorithm	No	0.795	0.765	0.780
Pradhan *et al* (2005)	Support Vector Machines	No	0.819	0.733	0.774
Surdeanu & Turmo (2005)	AdaBoost Algorithm	Yes	0.803	0.730	0.765

5 Conclusion

We have illustrated a shallow technique in which semantic labels are extracted in forms of chunks of phrases or words using a two-phase feature-enhanced string matching algorithm. This shallow technique is inspired by the research in the area of bio-molecular sequences analysis which advocates 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, 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 alternate, but plausible, way to handle the real corpus.

Acknowledgments. The work described in this paper was partially supported by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK4438/04H and CUHK4706/05H).

References

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

Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. *Proceedings of the 9th Conference on Computational Natural Language Learning* (*CoNLL*), 152-164.

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.

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.

CKIP (2004). *Sinica Chinese Treebank: An Introduction of Design Methodology*. Academic Sinica.

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

Haghighi, A., Toutanova, K., & Manning, C. (2005). A joint model for semantic role labeling. *Proceedings of the 9th Conference on Computational Natural Language Learning* (*CoNLL*).

Manning, C.D., & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Màrquez, L., Comas, P. R., Giménez, J., & Català, N. (2005). Semantic role labeling as sequential tagging. *Proceedings of the 9th Conference on Computational Natural Language Learning* (*CoNLL*).

Pradhan, S., Hacioglu, K., Ward, W., Martin, J.H., & Jurafsky, D. (2005). Semantic role chunking combining complementary syntactic views. *Proceedings of the 9th Conference on Computational Natural Language Learning* (*CoNLL*).

Punyakanok, V., Koomen, P., Roth, D., & Yih, W. (2005). Generalized inference with multiple semantic role labeling systems. *Proceedings of the 9th Conference on Computational Natural Language Learning* (*CoNLL*).

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

Surdeanu, M., & Turmo, J. (2005). Semantic role labeling using complete syntactic analysis. *Proceedings of the 9th Conference on Computational Natural Language Learning* (*CoNLL*).

Wagner, R.A., & Fischer, M.J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21, 1, 168-173.