An Algorithm of Identifying Semantic Arguments of a Verb from Structured Data

Minhua Huang Department of Computer Science Graduate School and University Center City University of New York New York, U.S.A mhuang@gc.cuny.edu

Abstract

We discuss a method for identifying semantic arguments of a verb from a sentence. It differs from existing methods by an unique feature that represents all semantic arguments of a verb in a syntactic parse tree. The feature is a path in which at least one of the children of a node is a root of a subtree that associates with a semantic argument. Experiments on WSJ data from Penn TreeBank and PropBank show that our method achieves an average of precision 92.3% and an average of recall 94.2% on identifying semantic arguments of over six hundred verbs.

1 Introduction

Semantic argument identification is one of the subtasks of semantic role labeling (Gildea and Jurafsky, 2002) (Chen and Rambow, 2002) (Hacioglu, 2004b) which classifies a sequence of words associated with a semantic argument of a verb but does not assign its role. It is the most difficult task in semantic role labeling. Moreover, it is one of the core techniques for a machine to understand the semantics of a sentence. For instance, in the sentences Lisa cut the ribbon with a pair of scissors. and The ribbon was cut by Lisa with a pair of scissors, cut is the verb. Semantic arguments of cut will be Lisa, the ribbon, and a pair of scissors, where Lisa is the one who performs the action of cutting, the ribbon is the material to be cut by Lisa, and a pair of scissors is the tool used for cutting. For semantic role labeling, arguments of cut need to be determined. Then, each argument will be assigned to a label, such as *agent*, *theme*, and *instrument* in the example. In this report, we presents an algorithm for finding a semantic argument of a verb which is the first task required for assigning a role for the verb.

Robert M. Haralick Department of Computer Science Graduate School and University Center City University of New York New York, U.S.A haralick@aim.com

Over the years, two approaches have been discussed by researchers, such as methods developed based on hierarchical trees (Gildea and Jurafsky, 2002) (Hacioglu, 2004b) (Hacioglu, 2004c) and methods developed based on flat chunks (Hacioglu and Ward, 2003) (Hacioglu, 2004a). In almost all the methods of the first approach, a syntactic tree is transformed into a sequence of constituents. Each semantic argument of a verb is represented by a set of constituents. Each constituent is represented by a set of features. These features are extracted based on linguistic knowledge and local knowledge of the tree structure. Finally, sophisticated classifiers such as support vector machines or maximum entropy modeling classifiers are employed to identify semantic arguments of each verb. In contrast to these methods, our method is based on the idea that if a sentence has a correspondent labeled rooted tree (parser tree), a semantic argument of a verb in the sentence will be associated with a labeled rooted subtree. Hence, all semantic arguments of a verb in the sentence will be represented by a set of labeled rooted subtrees. For each verb node v, there exists a path from node a to node b, from which, all roots of the subtrees will be extracted. Obviously, all semantic arguments of a verb are represented by an unique feature - a path.

We find the path for a verb in a labeled rooted tree associated with a sentence by the probabilistic graphical model discussed in the paper (Huang and Haralick, 2009). This model is fast, uses less memory, and is very effective on text data. We construct the path by starting from a verb node and determining the next node by selecting the node that has the largest probability value among the adjacent nodes which have not been encountered yet. Then, a sibling or a child of a node in the path is identified as a root of a subtree associating with a semantic argument of the verb.

We have tested our method on the WSJ data

the 00 section from Penn TreeBak and PropBank (Weischedel et al., 2007). There are a total of 233 trees associating with about 600 verbs and 2000 semantic arguments. The evaluation metrics we have used are *precision*, *recall*, and *f-measure*. By applying 10-folder cross validation technique, we have obtained an average of precision 92.64%, an average of recall 94.94%, and an average of f-measure 93.81%. Our experiments show that our method is particularly effective for identifying such semantic arguments, which them are associated with a sequence of consecutive words. Our method is less effective for semantic arguments, which they are associated with two or more sequences of consecutive words (separated by other phrases). Details are shown in Section 4. We are doing more experiments on CoNLL-2005 shared task data set to further verify our method.

The paper is organized into six sections. Section two defines a labeled rooted tree and forest; section three discusses the algorithm; section four demonstrates empirical results; section five shows related research and comparisons; and section six gives a conclusion.

2 A Labeled Rooted Tree and a Labeled Rooted Forest

A rooted tree T is a 3-tuple (V, E, r), where V is a finite set of vertices, $E \subseteq V \times V$ is a finite set of edges, and $r \in V$ is the root that all edges of T are directed away from it. The tree-order is the partial ordering on V for any $v, u \in V, u \leq v$ if and only if the unique path from the root r to v passes through u.

In T, the root r is a unique minimal vertex and has level 0. An edge (x < y) in E is an ordered pair $(x, y) \in (V \times V)$ s.t. x < y and there exists no $z \in V$ with x < z < y. In this case, x is a parent of y and y is a child of x. If two nodes¹ x, y have the same parent z, x and y are called siblings. Any node y is on the unique path from r to x is called an ancestor of x. In this case, x is a descendant of y. The sub-tree rooted at node x is the tree induced by descendants of x. A node with no children is an external node or a leaf. A node that is not a leaf node is an internal node. The largest depth of a node in T is the *height* of T.

2.0.1 Definition of a Labeled Rooted Tree

A labeled rooted tree is a 5-tuple (V, E, r, A, L). It is a rooted tree with additional two elements: a labeling alphabet A and a labeling function L: $V \rightarrow A$ that assigns labels to vertices.

2.0.2 Definition of a Labeled Rooted Forest

A labeled rooted forest is a set of labeled rooted trees, s.t. $F = \{T_i | i = 1...N\}$ where T_i is a labeled rooted tree.

3 The Method

3.1 Defining the Task

Let T = (V, E, r, A, L) be a labeled rooted tree associated with a sentence, where A is defined by (Weischedel et al., 2007). Let π be a set of labels associated with verbs, s.t. $\pi \subseteq A$. Let $C = \{C_1, C_2\}$ be a set of class categories, where C_1 represents that a path will be extended from the current node to an adjacent node; C_2 represents that a path will not be extended from the current node to an adjacent node.

The task can be stated as follows:

- Form a path P(x) = τ₁, → ..., → τ_K, where x ∈ V, L(x) ∈ π, and x is not a node in P'(y), P'(y) is a path that has been already formed previously. Each τ_k ∈ V, k = 1,..., K
 - find a sequence nodes $< \tau_1, \ldots, \tau_K >$, s.t.

$$<\tau_1,\ldots,\tau_K>$$

= $\underset{b_1,\ldots,b_K}{argmax} p(c_1,\ldots,c_K,b_1,\ldots,b_K)$

- where $c_k \in C$, b_k is one of adjacent nodes of b_{k-1} , b_{k-1} , $b_k \in V$, $b_{k-1}b_k \in E$.
- Form a set of roots $R(x) = \{r_i | i = 1 \dots M\}$, where $r_i \le \tau_k$, $L(r_i) \notin \pi$, and $1 \le k \le K$.
- Form a labeled rooted forest $F(x) = \{T_1, \ldots, T_M\}$, where each T_i is a labeled rooted tree, rooted as r_i , and induced by the descendants of r_i .
- T_i associates with a semantic argument of x.

Figure 1 illustrates the labeled rooted tree for the sentence Mrs. Hills said that the U.S. is still concerned about "disturbing developments in

¹In a rooted tree, a vertex can be also called a node.



Figure 1: Mrs. Hills said that the U.S. is still concerned about "disturbing developments in Turkey and continuing slow process in Malaysia".

Turkey and continuing slow process in Malaysia". $\pi = \{VB, VBN, VBG, VBZ, VBP, VBD\} \subseteq A.$

3.2 The Algorithm

3.2.1 Obtaining $< \tau_1, ..., \tau_K >$

We use equation (1) proposed by (Huang and Haralick, 2010b) (Huang and Haralick, 2009) to obtain the probability value of $p(c_1, \ldots, c_K, b_1, \ldots, b_K)$.

$$p(c_{1}, \dots, c_{K}, b_{1}, \dots, b_{K})$$

$$= \prod_{i=1}^{K} p(b_{i-1}|b_{i}c_{i})p(b_{i+1}|b_{i}, c_{i})p(b_{i}|c_{i})p(c_{i})$$

$$= \prod_{i=1}^{K} P(b_{i-1}, b_{i}, b_{i+1}, c_{i})$$
(1)

We use the equation (2) to find a sequence of optimal nodes $\langle \tau_1, ..., \tau_K \rangle$ in *T*, where $\tau_i \neq \tau_j$, $i, j = 1, ..., K, \tau_{i-1}\tau_i \in E$ and $\tau_i \tau_{i+1} \in E$ but $\tau_{i-1}\tau_{i+1} \notin E$

$$< \tau_1, ..., \tau_K >= \underset{c_1 \in C, b_1, b' \in E}{argmax} \{ p(b_2|b_1, c_1)p(b_1|c_1)p(c_1) \} \underset{c_2 \in C, b_2, b' \in E}{argmax} \{ p(b_1|s_2, c_2)p(b_3|b_2, c_2)p(b_2|c_2)p(c_2) \}$$

$$\underset{c_{K}\in C, b_{K}, b'\in E}{\operatorname{argmax}} \{ p(b_{K-1}|b_{K}, c_{K}) p(b_{K}|c_{K}) p(c_{K}) \}$$

$$(2)$$

Note: b' is a node in a path, $b'b_k \in E$, k = 1...K.

3.2.2 Time Complexity

For each node b_k , we need to assign a c_k , s.t.

$$\mathcal{P}_k = max\{P(b_{k-1}, b_k, b_{k+1}, c_k) \mid c_k \in C\}$$

To compute a $P(b_{k-1}, b_k, b_{k+1}, c_k)$, we need to have four multiplications. To obtain the maximum probability value \mathcal{P}_k , we need to have M-1 comparisons. In the case of a path of N nodes, we have

$$T_c = 4 * N * (M - 1) * (L - 1) = O(N * M * L)$$

Note: M is the cardinality of C, L is the maximum degree of a node in the tree, and N is the length of the path.

3.2.3 Memory Complexity

Because the global maximum probability is determined by each local maximal probability, for a path of N symbols, we only need to store the information of the current node. That is, we need only store M probability values in order to find the maximal probability value. Therefore,

$$M_c = M = O(M)$$

3.2.4 An Example of a Path

The path P(x), where L(x) = VBN (associating with the verb *concern*) in Figure 1 is $VBZ \rightarrow VP \rightarrow ADJP - PRD \rightarrow VBN$ in Figure 2.



Figure 2: All the semantic arguments of the verb concern can be extracted from this path.

3.2.5 Finding a set of roots $r_i \in R(x)$

Let Q(x) denote a set of nodes in path $\mathcal{P}(x)$ and let R(x) denote a set of roots we want to find.

- $R(x) = \phi, Q(x) = \{\tau_i | 1 \le i \le K\}$
- If Q(x) ≠ φ continue the following procedure:
 - 1. For each $\tau_i \in Q(x)$

R(x).

- 2. For all siblings of τ_i , find z, s.t. $L(z) \notin \pi$ and $z \notin \{\tau_i | i = 1, ..., K\}, R(x) \leftarrow R(x) \cup \{z\}$
- For all children of τ_i, if none of children z, L(Z) ∈ π, GOTO 4. Otherwise, find y, s.t. L(y) ∉ π and y ∉ {τ_i|i = 1,...,K}, R(x) ← R(x) ∪ {y}
 Q(x) ← Q(x) {τ_i}
- Otherwise, stop the procedure and return

3.2.6 Building a Labeled Rooted Forest F(x)Let T be a original labeled rooted tree, R(x) be a set of roots of subtrees that we want to build, and $F(x) = \phi$.

- For each r_i ∈ R(x), we assign r_i to the variable α. We initialize T_i with only a vertex r_i. We visit α.
- For {α, β} ∈ E, and β has not been visited, we attach {α, β} to T_i.

- 3. Assign β to α and visit α . Go to 2.
 - If α = r_i, then the labeled rooted tree T_i has been built. F(x) ← F(x) ∪ T_i
 - If α ≠ r_i, backtrack from α to its parent β in T. Then assign β to α and go to 2

Figure 3 illustrates a labeled rooted forest for verb *concern* for the labeled rooted tree corresponds to the sentence *Mrs. Hills said that the U.S. is still concerned about "disturbing developments in Turkey and continuing slow process in Malaysia".*

4 **Experiments**

We have tested our method on data set developed by (Weischedel et al., 2007), specifically, the WSJ section 00 from Penn Treebank and PropBank. A total of 233 trees associates with 233 sentences and 621 verbs, each verb has an average of three semantic arguments, hence about 2000 semantic arguments are in total. The evaluation metrics we have used are *precision*, *recall, and f-measure* (F_1) . Moreover, we have used 10-fold cross validation technique to obtain the average result.

For each sentence, Treebank provides a corresponding parse tree while PropBank provides corresponding semantic arguments of predicates in the sentence. These trees were generated by a statistic parser from corresponding sentences with an average accuracy 95%. These semantic arguments of predicates in PropBank were generated manually.

From the experiment, among 621 verbs, we found 621 paths in total. By excluding 30 types of paths of which occurs less than 2 times, six types of paths are remained. Among these remaining patterns, 86% paths fall in the first three patterns. Table 1 shows these patterns.

Moreover, a set of labeled rooted subtrees managed by labeled rooted forests are obtained based on the procedure described in Section 3.2.5. The test results are shown in Table 2. Note, the precision (recall or f-measure) is obtained by applying 10-fold cross validation. On the average, each time, among the $\frac{1}{10}$ semantic arguments that have been classified, about 93% semantic arguments are correctly identified and 7% semantic arguments are classified wrong. By checking these classified instances, we found that our method is very effective in the case of a semantic argument being a sequence of consecutive words. However, if a



Figure 3: A labeled rooted forest $F = \{T_1, T_2, T_3\}$ for verb *concern* for the sentence *Mrs. Hills said* that the U.S. is still concerned about "disturbing developments in Turkey and continuing slow process in Malaysia".

| NO | % | Path |
|----|------|---|
| 1 | 62.1 | $VBZ(VBD, VBG, VBP, VBN, VB) \rightarrow VP$ |
| 2 | 14.2 | $MD(TO) \rightarrow VP \rightarrow VP \rightarrow VB$ |
| 3 | 10.1 | $VBP(VBZ, VBD) \rightarrow VP$ |
| | | $\rightarrow VP \rightarrow VBN$ |
| 4 | 4.2 | $VBD(VBZ, VBN) \rightarrow VP$ |
| | | $\rightarrow RB \rightarrow VP \rightarrow VB$ |
| 5 | 2.4 | $TO \to VP \to VP \to VB \to VP \to VBN$ |
| 6 | 2.2 | $MD \to VP \to RB \to VP$ |
| | | $\rightarrow VBP(VB) \rightarrow VP \rightarrow VBN$ |

| | Table 2: | testing | result | on | W | SJ | data |
|--|----------|---------|--------|----|---|----|------|
|--|----------|---------|--------|----|---|----|------|

| Files | Precision | Recall | F-Measure |
|-------------|-----------|---------|-----------|
| | % | % | % |
| WSJ | | | |
| 20,37,49,89 | | | |
| Average | 92.335 | 94.1675 | 93.2512 |
| Standard- | | | |
| Deviation | 0.6195 | 0.5174 | 0.4605 |

semantic argument consists of two or more word fragments, separated by some phrases, our algorithm is less effective. For example, the sentence: *He wants to see for instance the movie Superman*. Our methods can not distinguish the semantic argument of *want* from the phrase *for instance*. The reason is that this phrase is the part of leaves of the tree induced from one of the roots determined by our algorithm. This suggests us that in order to exclude phrases from a semantic argument, we need to develop a method so that a set of subroots can be found. Each of them corresponds to a fragment of a semantic argument. Then, these fragments must be combined together to obtain the semantic argument. Moreover, other misclassified instances are generated by errors carried in original syntactic trees.

5 Related Researches and Comparisons

Methods for identifying semantic arguments of predicates in a sentence can be divided into two categories with respect to the representation of the sentence, namely tree-related (Gildea and Jurafsky, 2002) (Hacioglu, 2004b) (Hacioglu, 2004c) and chunk-related (Hacioglu and Ward, 2003) (Hacioglu, 2004a) semantic argument identifiers. While systems are built use the first approach are more accurate, systems are build use the second approach are very efficient and robust.

In the first approach, a sentence is represented by a syntactic tree (Gildea and Jurafsky, 2002) or some variants, such as a dependence tree (Hacioglu, 2004c) obtained from a syntactic tree. For each predicate in a tree, a set of syntactic constituents (non-terminals) is extracted. Each constituent is determined by a set of features derived from sentence structure or a linguistic context defined for the constituent. These features may be predicate lemma, path from constituents to the predicate, phrase type, dependency relations between predicates and constituents, position of constituent with respect to it predicate, voice, head word stem, sub-categorization. Classifiers such as support vector machines and maximum entropy models have been employed to identify constituents into one of semantic arguments of predicates.

In the second approach, semantic argument identification is formulated as a chunking task (Hacioglu, 2004a). For each predicate in a sentence, each word in the sentence is classified into three categories which are inside a semantic argument, outside a semantic argument, or begin a new semantic argument by using a set of features defined for the word. These features may be the lexicon of the word, the POS of the word, and the syntactical phrase chunks. Then, a bank of SVM classifiers, a one-versus-all classifier, can be used for each class.

Our method is based on syntactic trees. However, our method differs from others in several ways. Instead of linearly transforming a syntactic tree into a sequence of syntactic constituents, we directly traverse the tree from top to bottom and left to right to find a set roots, each of them corresponds a semantic argument of a verb. Moreover, instead of finding a set of features for each semantic arguments of a verb based on the linguistic knowledge or syntactic structure, we find our feature, a path, by the method proposed by (Huang and Haralick, 2010a). This method is simple, fast, and uses less memory. In contrast with other methods, our feature represents not one semantic argument but all semantic arguments of a verb. Furthermore, instead of finding semantic arguments of a verb by using complex classifiers such as support vector machine or maximum entropy models, we determine the semantic arguments of a verb only by setting simple rules of looking up relatives of each node in our path. We argue that our feature is the most effective, efficient, and simplest feature compared with the existing methods.

6 Conclusion

An algorithm for identifying semantic arguments of a verb in a sentence has been discussed throughout this paper. The method is developed based on the argument that a link must exist from a verb to its all semantic arguments if a sentence is structured syntactically with the root vertex been labeled with S and the leaf vertices been labeled with lexicon of words in the sentence. A semantic argument of a verb in the sentence can be represented as a labeled rooted subtree rooted at an internal node and induced by its all descendants. Therefore, to find semantic arguments of a verb is to find a set of such subtrees, more precisely, a set of roots. In our method, we apply a probabilistic graphical model to extract such a link – a path. Then we determine these roots from the path by a set of predefined rules. Experiments are conducted on WSJ data set from Penn Treebank and PropBank. Results demonstrate that our method is effective.

References

- John Chen and Owen Rambow. 2002. Use of deep linguistic features for the recognition and labeling of semantic argument. In *Proceedings of EMNLP-2003*.
- Daniel Gildea and Daniel Jurafsky. 2002. Automatic labelling of semantic roles. *Computational Linguistics*, pages 245–288.
- Kadri Hacioglu and Wayne Ward. 2003. Target word detection and semantic role chunking using support vector machines. In *Proceedings of HLT/NAACL-03*.
- Kadri Hacioglu. 2004a. A lightweight semantic chunking model based on tagging. In *Proceedings* of *HLT/NAACL-04*.
- Kadri Hacioglu. 2004b. A semantic chunking model based on tagging. In *Proceedings of HLT/NACCL-2004*.
- Kadri Hacioglu. 2004c. Semantic role labeling using dependency trees. In *Proceedings of Coling 2004*, pages 1273–1276, Geneva, Switzerland, Aug 23– Aug 27. COLING.
- Minhua Huang and Robert M. Haralick. 2009. Discovering patterns in texts. In 2009 IEEE International Conference on Semantic Computing, pages 59–64.
- Minhua Huang and Robert M. Haralick. 2010a. Discovering semantics of a word from a sentence. In 2010 International Conference on Artificial Intelligence and Pattern Recognition, pages 51–57.
- Minhua Huang and Robert M. Haralick. 2010b. Recognizing Patterns in Texts. River.
- Ralph Weischedel, Martha Palmer, Mitchell Marcus, and Eduard Hovy. 2007. Ontonotes release 2.0 with ontonotes db tool v. 0.92 beta and ontoviewer v.0.9 beta. In *http://www.bbn.com/NLP/OntoNotes*.