Dependency Forest for Sentiment Analysis

Zhaopeng Tu*, Wenbin Jiang, Qun Liu, and Shouxun Lin

Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, CAS, Beijing, China
{tuzhaopeng,jiangwenbin,liuqun,sxlin}@ict.ac.cn

Abstract. Dependency Grammars prove to be effective in improving sentiment analysis, because they can directly capture syntactic relations between words. However, most dependency-based systems suffer from a major drawback: they only use 1-best dependency trees for feature extraction, which adversely affects the performance due to parsing errors. Therefore, we propose an approach that applies dependency forest to sentiment analysis. A dependency forest compactly represents multiple dependency trees. We develop new algorithms for extracting features from dependency forest. Experiments show that our forest-based system obtains 5.4 point absolute improvement in accuracy over a bag-of-words system, and 1.3 point improvement over a tree-based system on a widely used sentiment dataset. Our forest-based system also achieves state-of-the-art performance on the sentiment dataset.

Keywords: dependency forest, sentiment analysis.

1 Introduction

Dependency grammars have received a lot of attention in sentiment analysis (SA). One important advantage of dependency grammars is that they can directly capture syntactic relations between words, which are key to resolving most parsing ambiguities. As a result, employing dependency trees produces substantial improvements in sentiment analysis [12,6,10].

However, most dependency-based systems suffer from a major drawback: they only use 1-best dependency trees for feature extraction, which adversely affects the performance due to parsing errors(93% [8] and 88% [3] accuracies for English and Chinese on standard corpora respectively). To make things worse, sentiment corpora usually consist of noisy texts from web, which will lead to a much lower parsing quality. As we will show, the tree-based systems still commits to using features extracted from noisy 1-best trees. Due to parsing error propagation, many useful features are left out of the feature set.

To alleviate this problem, an obvious solution is to offer more alternatives. Recent studies have shown that many tasks in natural language processing can benefit from widening the annotation pipeline: using packed forests [13] or dependency forests [20] instead of 1-best trees for statistical machine translation,

* Corresponding author.

M. Zhou et al. (Eds.): NLPCC 2012, CCIS 333, pp. 69–77, 2012.

packed forests for semantic role labeling [22], forest reranking for parsing [2], and word lattices reranking for Chinese word segmentation [4].

Along the same direction, we propose an approach that applies dependency forest, which encodes exponentially many dependency trees compactly, for tree-based sentiment classification systems. In this paper, we develop a new algorithm for extracting features from dependency forest. Experiments show that our forest-based system obtains 5.4 point absolute improvement in accuracy over a bag-of-words system, and 1.3 point improvement over a tree-based system on a widely used sentiment dataset [18].

2 Related Work

Our research builds on previous work in the field of sentiment classification and forest-based algorithms. For sentiment classification, the design of lexical and syntactic features is a fundamental step. There has been an increasing amount of work on feature-based algorithms for this problem. Pang and Lee [18] and Dave et al. [9] represent a document as a bag-of-words; Matsumoto et al. [12] extract frequently occurring connected subtrees from dependency parsing; Joshi and Penstein-Ros'e [6] use a transformation of dependency relation triples; Liu and Seneff [10] extract adverb-adjective-noun relations from dependency parser output while Wu et al. [21] extract features from phrase dependency parser.

Previous research has convincingly demonstrated forests ability to offer more alternatives, which is useful to solving parsing error propagation problem, and has led to improvements in various NLP tasks, including statistical machine translation [13,20], semantic role labeling [22], and parsing [2].

3 Background

In this section, we present the baseline system that extracts features from 1-best dependency trees.

Figure 1(a) shows a dependency tree of the English sentence *the film is sick, slick fun*. The dependency tree expresses relation between words by head-dependents relationships of nodes. The arrow points from the dependent to its head. For example, in Figure 1(a), *fun* is the head of *slick*.

Inspired by the previous work [12], we extract connected subtrees from dependency trees. A connected subtree is a more general form obtained by removing zero or more nodes from the original dependency tree. Figure 1(b) shows some examples. The top left subtree is obtained by removing the node *slick*, and the bottom left subtree is obtained by further removing the node *the*.

Recent studies show that this kind of partial subtrees could capture the syntactic information between words quite well [12,14,5]. For example, in Figure 1(b), to express the relation between the words *film* and *fun*, a subtree t does not only show the co-occurrence of *film* and *fun*, but also make sure that they are syntactically connected by the word *is*.

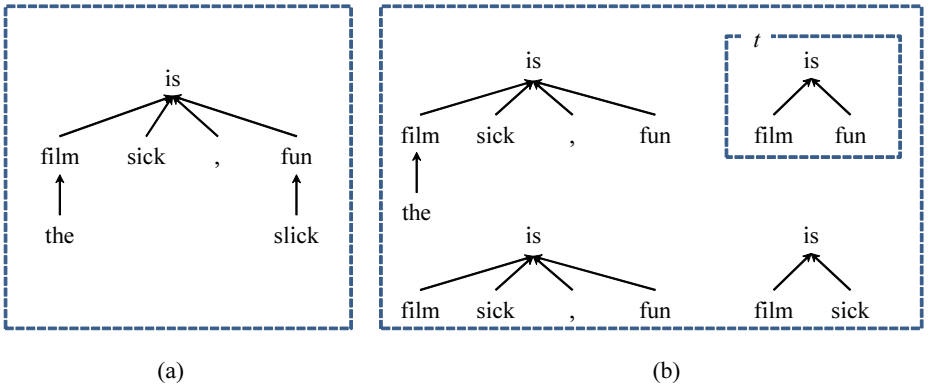

Fig. 1. A dependency tree of the sentence the film is sick, slick fun and examples of corresponding dependency substructures

Since the number of dependency features tends to be very large, we only remain the features that occur frequently in the corpus.

4 Dependency Forest

As the tree-based system relies on 1-best trees, the quality of features might be affected by parsing errors and therefore ultimately results in classification mistakes. We propose to encode multiple dependency trees in a compact representation called dependency forest, which provides an elegant solution to the problem of parsing error propagation.

Figure 2(a) and 2(b) show two dependency trees for the example English sentence in Figure 1. The word sick can either be an adjective as an attribute of the film, or be a modificatory word like slick for the word fun. The two dependency trees can be represented as a single dependency forest by sharing common nodes and edges, as shown in Figure 2(c).

Each **node** in a dependency forest is a word. We assign a span to each node to distinguish among nodes. For example, the span of the word *sick* is (3,4) because it is the fourth word in the sentence. Since the seventh word *fun* dominates the word *slick*, the span is (5,7). Note that the position of *fun* itself is taken into consideration.

The nodes in a dependency forest are connected by **hyperedges**. While a edge in a dependency tree points from a dependent to its head, a hyperedge groups all dependents of their common head. For example, the hyperedge e_2:

$$e_2\colon \langle is_{0,7}, (film_{0,2}, fun_{3,7})\rangle$$

denotes that both $film_{0,2}$ and $fun_{3,7}$ are dependents of the head $is_{0,7}$.

Formally, a *dependency forest* is a pair $\langle V, E\rangle$, where V is a set of nodes, and E is a set of hyperedges. For a given sentence $w_{1:n} = w_1 \ldots w_n$, each node

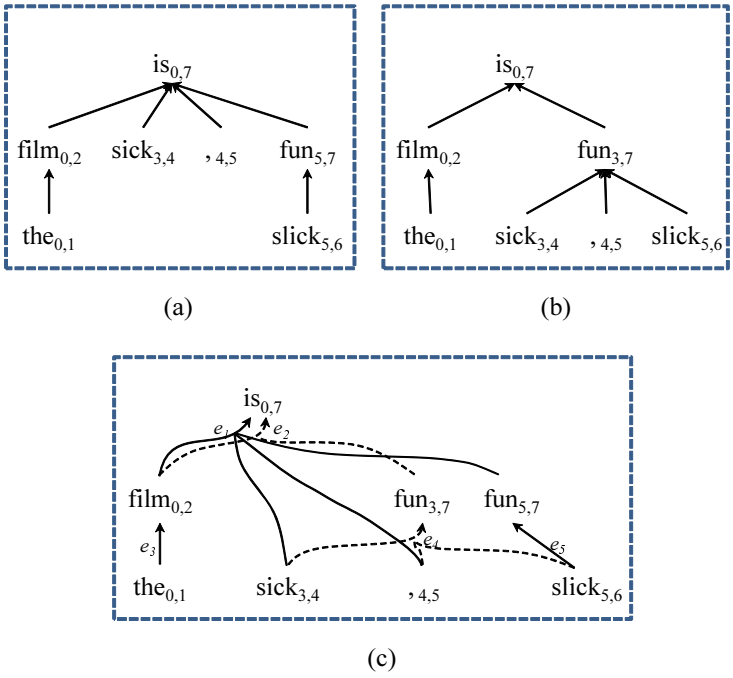

Fig. 2. (a) the dependency tree in Figure 1, (b) another dependency tree for the same sentence, and (c) a dependency forest compactly represents the two trees.

$v \in V$ is represented as $w_{i,j}$, denoting that the word w dominates the substring $w_{i+1} \dots w_j$. Each hyperedge $e \in E$ is a pair $\langle head(e), tails(e) \rangle$, where $head(e) \in V$ is the head of the hyperedge and $tails(e) \in V$ are its dependents.

We followed the work [20] to construct dependency forest from k-best parsing results, and transform a dependency forest into a hypergraph.

5 Forest-Based Subtree Extraction

In tree-based systems, we can extract dependency subtrees by simply enumerating all nodes in the tree and combining the subtrees of their dependents with the heads. However, this algorithm fails to work in the forest scenario because there are usually exponentially many subtrees of a node.

To solve this problem, we develop a new bottom-up algorithm to extract dependency subtrees. Our approach often extracts a large amount of dependency subtrees as each node has many hyperedges. To maintain a reasonable feature set size, we discard any subtrees that dont satisfy two constraints:

1. appear in at least f distinct sentences in the dataset [12];
2. the *fractional count* should not be lower than a pruning threshold p;

Here the fractional count of a subtree is calculated like in the previous work [13,20]. Given a tree fragment t, we use the inside-outside algorithm to compute its posterior probability:

$$\alpha\beta(t) = \alpha(root(t)) \times \prod_{e \in t} p(e) \times \prod_{v \in leaves(t)} \beta(v) \quad (1)$$

where $root(t)$ is the root of the subtree, e is an edge, $leaves(t)$ is a set of leaves of the subtree, $\alpha(\cdot)$ and $\beta(\cdot)$ are outside and inside probabilities, respectively. For example, the subtree rooted at $fun_{3,7}$ in Figure 2(c) has the following posterior probability:

$$\alpha(fun_{3,7}) \times p(e_4) \times \beta(sick_{3,4}) \times \beta(,_{4,5}) \times \beta(slick_{5,6})$$

Now the fractional count of the subtree t is

$$c(t) = \frac{\alpha\beta(t)}{\alpha\beta(TOP)} \quad (2)$$

where TOP is the root node of the forest. As a partial connected subtree might be non-constituent, we approximate the fractional count by taking that of the minimal constituent tree fragment that contains the connected subtree.

We observed that if a subtree appears in at least f distinct sentences, then each edge in the subtree should also occurs at least f times. According to this observation, we can first enumerate all edges that appear in at least f distinct sentences, then we can check whether the edge in this set when dealing with a head and a dependent. Take the edge $\langle sick, fun \rangle$ in Figure 2(c) for example, if the occurrence of the edge is lower than f, then any subtree that contain this edge could not be possible appear in at least f times.

Algorithm 1 shows the bottom-up algorithm for extracting subtrees from a dependency forest. This algorithm maintains all available subtrees for each node (line 12). The subtrees of a head can be constructed from those of its dependents. For instance, in Figure 2(c), as the subtree rooted at $fun_{3,7}$ is

$$(slick)\ fun$$

we can obtain another subtree for the node $is_{0,7}$ by attaching the subtree of its dependent to the node (*EnumerateSubtrees* in line 8)

$$is\ ((slick)\ fun)$$

Note that we only keep the subtrees that appear in at least f distinct sentences (line 6), and have a fractional count not lower than p (line 10).

6 Experiments

We carried out experiments on the movie review dataset [18], which consists of 1000 positive reviews and 1000 negative reviews. To obtain dependency trees,

Algorithm 1. Algorithm of extracting subtrees from a dependency forest. All subtrees should appear in at least f distinct sentences and have a fractional count not lower than p.

Input: a forest F, and f and p
Output: minimal subtree set $\mathcal{T}$

avail_edges ← [edges that appear in at least f distinct sentences]
for each node $v \in V$ in a bottom-up order **do**
  for each hyperedge $e \in E$ and $head(e) = v$ **do**
    avail_nodes ← $\emptyset$
    for each node $w \in tails(e)$ **do**
      if edge $\langle v, w \rangle \in$ *avail_edges* **then**
        avail_nodes.append(w)
    $S \leftarrow$ *EnumerateSubtrees* (v, *avail_nodes*)
    for each subtree $t \in S$ **do**
      if $c(t) \geqslant p$ **then**
        $\mathcal{T}$.append(t)
  keep subtrees for v

we parsed the document using the Stanford Parser [7] to output constituency trees and then passed the Stanford constituency trees through the Stanford constituency-to-dependency converter [11].

We consider two baselines:

1 **Unigrams**: using unigrams that occur at least 4 times [18]
2 **Unigrams+Subtree$_{1-best}$**: Unigrams features plus partial connected subtrees extracted from 1-best dependency trees

We construct dependency forest from 100-best parsing list.[1] We set the parameter $f = 10$ for both dependency trees and forests, and pruning threshold $p = 0.01$ for dependency forests.[2] All experiments are carried out using the MaxEnt toolkit[3] with default parameter settings. All results reported are based on 10-fold cross validation.

Table 1 shows the result on the movie review dataset. The first column Features indicates where the features are extracted from: 1-best dependency trees, 100-best list or dependency forests. The second column denotes the averaged number of extracted features, while the last column is the averaged time of subtree extraction. We compare our method to previously published results on the same dataset (rows 9-11), showing that our approach is very competitive. We find that using subtrees from 100-best list or forests achieve significant improve-

[1] The speed of construction is approximately dozens of milliseconds per sentence. Most of the time cost is attributed to the calculation of inside and outside probabilities.

[2] We only use fractional count pruning for dependency forest, because the inside-outside algorithm for computing fractional count is only available for hypergraphs. As we extract features from the trees in k-best list individually, we cannot use it for k-best list scenario.

[3] `http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html`

Table 1. Result on the movie review dataset. Here "Number" (column 2) indicates the averaged number of features used and "Time" (column 4) denotes the averaged subtree extraction time (second/document). "Subtree$_{structure}$" denotes that the subtrees are extracted from 1-best tree, 100-best list or forest. The baseline system (row 2) used the unigrams that occur at least 4 times, and another baseline system (row 4) furthermore incorporates dependency subtrees extracted from 1-best trees. We use "†" to denote a result is better than baseline "Unigram" significantly, and "‡" denote better than both "Unigram" and "Unigram + Subtree$_{1-best}$" significantly, at $p < 0.01$ (sign test).

Features	Number	Accuracy	Time
Unigram	17,704	86.2	–
Subtree$_{1-best}$	12,282	74.2	0.33
Unigram + Subtree$_{1-best}$	29,986	90.3†	–
Subtree$_{100-best}$	24,006	81.9	35.47
Unigram + Subtree$_{100-best}$	41,710	90.2†	–
Subtree$_{forest}$	18,968	81.2	6.93
Unigram + Subtree$_{forest}$	36,674	**91.6**‡	–
Pang et al. [18]	–	87.1	–
Ng et al. [17]	–	90.5	–
Yessenalina et al. [23]	–	91.8	–

ment over 1-best trees, validating our belief that offering more alternatives could produce substantial improvements. Using 100-best list produce only double subtrees in over 100 times longer than using 1-best trees, indicating that a k-best list has too few variations and too many redundancies [2]. When incorporating unigrams features, forest-based system obtains significant improvement of 5.4 point in accuracy over the bag-of-words system, and 1.3 point improvement over the tree-based system. An interesting finding is that combining subtrees from 100-best list and unigrams features doesnt achieve any improvement over 1-best tree. We conjecture that: (1) as most syntactic information is already captured by 1-best trees, using 100-best list can introduce little new information, (2) more noisy information would be introduced when extracting features from 100-best list, because there would be some low-quality parsing trees in the 100-best list (e.g. the trees at the foot of 100-best list). In contrast, we can extract new subtrees from dependency forests, which could not be extracted from any single tree in 100-best list (e.g. a subtree that consists of two parts from two different dependency trees). On the other hand, with the help of fractional count pruning, we would discard most low-quality subtrees.

7 Conclusion and Future Work

In this paper, we have proposed to extract features represented as partial connected subtrees from dependency forests, and reduced the complexity of the extraction algorithm by discard subtrees that have low fractional count and occurrence. We show that using dependency forest leads to significant improvements over that of using 1-best trees on a widely used movie review corpus.

In this work, we still select features manually. As convolution kernels could exploit a huge amount of features without an explicit feature representation [14,5,1,16,15,19], we will combine dependency forest and convolution kernels in the future.

Acknowledgments. The authors were supported by 863 State Key Project No. 2011AA01A207. We thank the anonymous reviewers for their insightful comments.

References

1. Bunescu, R., Mooney, R.: A Shortest Path Dependency Kernel for Relation Extraction. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 724–731. Association for Computational Linguistics, Vancouver (2005)
2. Huang, L.: Forest reranking: discriminative parsing with non-local features. In: Proceedings of ACL 2008: HLT, Columbus, Ohio, pp. 586–594 (May 2008)
3. Jiang, W., Liu, Q.: Dependency parsing and projection based on word-pair classification. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 12–20. Association for Computational Linguistics, Uppsala (2010)
4. Jiang, W., Mi, H., Liu, Q.: Word lattice reranking for chinese word segmentation and part-of-speech tagging. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pp. 385–392. Coling 2008 Organizing Committee, Manchester (2008)
5. Johansson, R., Moschitti, A.: Syntactic and semantic structure for opinion expression detection. In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, Uppsala, Sweden, pp. 67–76 (July 2010)
6. Joshi, M., Penstein-Rosé, C.: Generalizing dependency features for opinion mining. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pp. 313–316. Association for Computational Linguistics, Suntec (2009)
7. Klein, D., Manning, C.D.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pp. 423–430. Association for Computational Linguistics, Sapporo (2003)
8. Koo, T., Collins, M.: Efficient third-order dependency parsers. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 1–11. Association for Computational Linguistics, Uppsala (2010)
9. Kushal Dave, S.L., Pennock, D.: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In: Proceedings of the 12th International Conference on World Wide Web, pp. 519–528. ACM (2003)
10. Liu, J., Seneff, S.: Review Sentiment Scoring via a Parse-and-Paraphrase Paradigm. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, pp. 161–169 (August 2009)
11. de Marneffe, M.C., Manning, C.D.: The stanford typed dependencies representation. In: Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation, Manchester (August 2008)
12. Matsumoto, S., Takamura, H., Okumura, M.: Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 301–311. Springer, Heidelberg (2005)

13. Mi, H., Huang, L.: Forest-based translation rule extraction. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, Honolulu, Hawaii, pp. 206–214 (September 2008)
14. Moschitti, A.: Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 318–329. Springer, Heidelberg (2006)
15. Moschitti, A., Pighin, D., Basili, R.: Tree kernels for semantic role labeling. Computational Linguistics 34(2), 193–224 (2008)
16. Moschitti, A., Quarteroni, S.: Kernels on Linguistic Structures for Answer Extraction. In: Proceedings of ACL 2008: HLT, Short Papers, pp. 113–116. Association for Computational Linguistics, Columbus (2008)
17. Ng, V., Dasgupta, S., Arifin, S.M.N.: Examining the Role of Linguistic Knowledge Sources in the Automatic Identification and Classification of Reviews. In: Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, Sydney, Australia, pp. 611–618 (July 2006)
18. Pang, B., Lee, L.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, pp. 271–278 (June 2004)
19. Tu, Z., He, Y., Foster, J., van Genabith, J., Liu, Q., Lin, S.: Identifying high-impact sub-structures for convolution kernels in document-level sentiment classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 338–343. Association for Computational Linguistics, Jeju Island (2012)
20. Tu, Z., Liu, Y., Hwang, Y.S., Liu, Q., Lin, S.: Dependency Forest for Statistical Machine Translation. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), Beijing, China, pp. 1092–1100 (July 2010)
21. Wu, Y., Zhang, Q., Huang, X., Wu, L.: Phrase Dependency Parsing for Opinion Mining. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, pp. 1533–1541 (August 2009)
22. Xiong, H., Mi, H., Liu, Y., Liu, Q.: Forest-based semantic role labeling. In: Twenty-Fourth AAAI Conference on Artificial Intelligence, pp. 1039–1044 (2010)
23. Yessenalina, A., Choi, Y., Cardie, C.: Automatically generating annotator rationales to improve sentiment classification. In: Proceedings of the ACL 2010 Conference Short Papers, pp. 336–341. Association for Computational Linguistics, Uppsala (2010)