ГРАФОВЫЙ ПОДХОД В ЗАДАЧЕ ПОСТРОЕНИЯ СИНТАКСИЧЕСКИХ ДЕРЕВЬЕВ ДЛЯ РУССКОГО ЯЗЫКА

Музычка С. (s.muzychka@samsung.com),
Пионтковская И. (p.irina@samsung.com)

Исследовательский Центр Самсунг, Москва, Россия

Синтаксический анализ является одним из ключевых компонентов в большом количестве задач автоматической обработки текстов на естественном языке. Решение задачи построения деревьев зависимостей необходимо достаточно широкому кругу систем машинного перевода, автоматического синтеза и распознавания речи и пр. В статье изложен метод автоматического построения синтаксических деревьев на основе графового (graph-based) подхода. Мы предлагаем достаточно простую для реализации вероятностную модель, являющуюся модификацией работы [3]. Наш подход состоит в замене статистической суммы (partition function) некоторым приближением, не ухудшающим качество алгоритма и существенно снижающим сложность вычислений. Продемонстрировано, что указанный метод может быть успешно применен к синтаксическому анализу национального корпуса русского языка SynTagRus. Полученная точность алгоритма (по UAS) превосходит существующие аналоги.

Ключевые слова: синтаксический анализ, графовый подход, SynTagRus, графические модели

GRAPH-BASED APPROACH IN THE DEPENDENCY PARSING TASK FOR RUSSIAN LANGUAGE

Muzychka S. (s.muzychka@samsung.com),
Piontkovskaya I. (p.irina@samsung.com)

Samsung R&D Institute, Moscow, Russia

Dependency parsing is one of the key components in a large number of tasks of automatic processing of natural language texts. Effective dependency tree construction can be applied to a wide variety of machine translation systems, automatic speech synthesis and recognition, and so forth. Graph-based approach in dependency parsing proved to be efficient for morphologically rich languages due to its possibility to deal with non-projective dependency trees and flexible word order. Usually graph-based methods enable to perform probabilistic analysis over distribution on the set of syntax trees. In some NLP tasks it is not required to present a full syntactic parsing (in particular, to set labels on the edges of the tree). It is enough to find a parent for a given token. In this case, the graph-based approach is more appropriate because the likelihood that a token is an ancestor of the other, can be calculated by the explicit formula.

We consider a task of automatic syntax tree construction with application to Russian language corpus SynTagRus. We propose a novel technique which enables to reduce time costs for training and doesn't affect resulting accuracy. Experiments show that our algorithm outperforms existing analogues on SynTagRus in UAS (unlabeled attachment score) measure (percentage of correctly identified unmarked dependencies).

Key words: syntax parsing, graph-based approach, SynTagRus, dependency parsing

1. Introduction

Dependency parsing is one of the key components in a large number of tasks of automatic processing of natural language texts. An automatic construction of syntactic trees can be applied to a wide variety systems of machine translation, automatic speech synthesis and recognition.

Existing methods of dependency parsing can be divided into two fairly large groups: transition-based and graph-based approaches [4]. The first is based on the construction of a finite state automaton. Tokens of the sentence are sequentially fed to the algorithm after which the automaton transfers to a new state and corrects syntax tree. One popular example of this approach is MaltParser, which has been successfully applied to the analysis of many languages with complex morphology, in particular for the Russian language [7, 8].

Graph-based approach is based on a discriminative probabilistic model of dependencies between tokens of the sentence. Its popular implementation is MSTParser [5]. The advantage of graph-based approach towards a transition-based parsing is the capability of non-projective trees construction without any processing.

Usually graph-based methods enable to perform probabilistic analysis over distribution on the set of syntax trees. In some NLP tasks it is not required to present a full syntactic parsing (in particular, to set labels on the edges of the tree). It is enough to find a parent for a given token. In this case, the graph-based approach is more appropriate because the likelihood that a token is an ancestor of the other, can be calculated by the explicit formula [3].

We propose a simple for implementation probability model on the basis of graph-based approach which is a modification of [3]. The approach is based on the replacement of the partition function with its approximation which doesn't affect the performance of the algorithm and reduces computation costs. Out algorithm can be successfully applied to Russian language corpus SynTagRus. The accuracy of the resulting classifier outperforms existing analogues in UAS (unlabeled attachment score) measure [7, 8].

2. Model Description

Dependency tree *T* of the sentence $x = \{x_i\}_{i=1}^{N}$ consisting of *N* tokens x_i is an oriented graph-tree which nodes correspond to *x* and one outstanding node without incoming edges is called root *r*(*T*). Denote the set of all dependency trees of sentence *x* by $\mathcal{T}(x)$. The problem is to construct the most appropriate syntax tree $T \in \mathcal{T}(x)$ for a given sentence *x*. Everywhere below for simplicity we identify each token with its index in the sentence *i*.

The following probability model was suggested by [3]. For each pair of tokens (*h*, *m*), *h* ≠ *m* (*h*—head, *m*—modifier) we construct a set of features $\mathbf{f}_{hm}$ (for example, $\mathbf{f}_{hm}$ can contain parts of speech of *h* and *m*, distance between them or lemmas of their neighbours) and calculate its pairwise potential

$$w_{hm} = exp\left(\sum_{f \in \mathbf{f}_{hm}} \theta_f\right) \ (\theta_f \in \mathbb{R} \text{ are parameters of the model})$$

measuring the rate of dependence between *h* and *m*. Also for each token *m* we construct a set of features $\mathbf{f}_m$ and calculate single potential

$$w_{0m} = exp\left(\sum_{f \in \mathbf{f}_m} \theta_f\right)$$

(index 0 expresses fictitious node-root). The weight *w*(*t*) of dependency parsing tree *T* is defined by

$$w(T) = w_{0r(T)} \prod_{(h,m) \in E(T)} w_{hm},$$

where *r*(*T*) is the tree root, and *E*(*T*) is the set of all edges of *T*. The probability of the tree P(*T*|*x*) is a normalized weight

$$P(T|x) = \frac{w(T)}{Z_0}, \text{where } Z_0 = \sum_{T \in \mathcal{T}(x)} w(T) \quad \text{is its partition function} \tag{1}$$

Having trained model the most appropriate dependecy tree can be found using Eisner [2] or Chu-Liu-Edmonds algorithms [1].

The parameters of the model $\{\theta_f\}$ are fitted on the training corpus using maximum likelihood method. Applying matrix tree theorem [9] we get that Z_0 is equal to the determinant of the matrix [3]

$$\begin{pmatrix} w_{01} & w_{02} & w_{03} & & w_{0N} \\ -w_{21} & \sum_i w_{i2} & -w_{23} & & -w_{2N} \\ -w_{31} & -w_{32} & \sum_i w_{i3} & & -w_{3N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -w_{N1} & -w_{N2} & -w_{N3} & & \sum_i w_{iN} \end{pmatrix}$$

Therefore the most computationally difficult problem is to calculate logarithm of partition function (1) and its gradient. To deal with this obstacle we use 2 technical tricks (see appendix for proof)

1. Replace Z_0 with the sum over all oriented graphs

$$Z_1 = \prod_{\substack{(h.m):h\neq m \\ 0\leq h\leq N \\ 0<m\leq N}} (1 + w_{hm}) \tag{2}$$

2. Replace Z_0 with the sum over all functional graphs (oriented graph is functional if number of input edges equals 1 for each node in the graph

$$Z_2 = \prod_{m=1}^{N} \sum_{\substack{(h.m):h\neq m \\ 0\leq h\leq N}} w_{hm} \tag{3}$$

Application of (2) and (3) sufficiently reduces computational complexity during training and doesn't decrease accuracy of the resulting algorithm as experiments show.

3. Results

The experiments were carried out on the SynTagRus corpus [7]. We used different combinations of parts of speech, words and lemmas as features. Let *pos*(*h*) be a part of speech of token h; also denoted by *morph*(*h*) its full morphological label; *lemma*(*h*)—its lemma; *token*(*h*)—its token and *dist*(*h,m*)—distance between tokens *h* and *m*. In the following table we present the list of the most significant features used in our model

Features for w hm	Features for w hm
pos(*h*) + *pos*(*m*)	*pos*(*h*)
morph(*h*) + *morph*(*m*)	*token*(*h*)
token(*h*) + *pos*(*m*)	*lemma*(*h*)
pos(*h*) + *token*(*m*)	*morph*(*h*)
lemma(*h*) + *pos*(*m*)	
pos(*h*) + *lemma*(*m*)	
"head=" + *pos*(*h*)	
"modifier=" + *pos*(*m*)	
dist(*h, m*)+*pos*(*h*)+*pos*(*m*)	
dist(*h,m*)	

Also we used features taking into account punctuation, frequent tokens and neighbours of h and m.

For training and testing the corpus was separated into 2 parts by random split: 90%—for training and 10%—for testing. We consider 3 variants of training corresponding to the choice of partition function and 2 variants of inference: Eisner and Chu-Liu-Edmonds algorithms. The results of the testing (by UAS measure) are presented in the following table.

	Z_0	Z_1	Z_2
Eisner (perfect morphology)	90.50%	90.29%	**90.56%**
Chu-Liu-Edmonds(perfect morphology)	90.09%	90.03%	90.31%
Eisner (predicted morphology)	86.76%	86.54%	86.97%
Chu-Liu-Edmonds (predicted morphology)	86.05%	85.92%	86.46%

Morphological tagging was performed with morphological parser [6]. The following table shows the results of error statistics depending on part of speech of dependent word for the best experiment (perfect morphology + Eisner + Z_2)

Part of speech	Accuracy on POS	Common mistake
S	93.80%	2.490%
PR	82.57%	2.020%
V	88.90%	1.600%
CONJ	80.67%	1.240%
ADV	85.87%	0.850%
A	95.83%	0.600%
PART	90.30%	0.400%
NID	87.39%	0.050%
NUM	94.48%	0.110%
UNKNOWN	76.41%	0.050%
COM	85.71%	0.002%

In order to compare our accuracy with the current state-of-the-art we present the following table based on the papers [7, 8].

Nivre	Sharoff
89.00%	89.40%

4. Discussion

The accuracy of resulting classifier outperforms existing analogues [7, 8] by UAS-measure. The choice of partition function doesn't affect an accuracy of the algorithm. Moreover usage of Z_1 and Z_2 sufficiently increase efficiency of the algorithm.

References

[1] *J. Edmonds,* (1967), Optimum branching, Journal of Research of the National Bureau of Standards, 71B, pp. 233–240.

[2] *J. Eisner,* (1996), Three new probabilistic models for dependency parsing: An exploration, Proceedings of the 16th conference on Computational linguistics, Copengagen, pp. 340–345.

[3] *T. Koo, A. Globerson, X. Careras, M. Collins,* (2007), Structured Prediction Models via the Matrix Tree Theorem, Proceedings of EMNLP, Prague, pp. 141–150.

[4] *S. Kubler, R. McDonald, J. Nivre,* (2009), Dependency parsing, Morgan & Claypool publishers.

[5] *R. McDonald, F. Pereira, K. Ribarov, J. Hajic,* (2005), Non-projective Dependency Parsing using Spanning Tree Algorithms, Proceedings of HLT/EMNLP, Vancouver, pp. 523–530.

[6] *S. Muzychka, A. Romanenko, I. Piontkovaskaya,* (2014), Conditional Random Field for morphological disambiguation in Russian, Proceedings of the International conference "Dialogue 2014" Bekasovo, 2014, pp. 456–465.

[7] *J. Nivre, I. Boguslavsky, L. Iomdin,* (2008), Parsing the SynTagRus Treebank of Russian, Proceedings of the International conference "Dialogue 2008", Bekasovo, pp. 641–648.

[8] *S. Sharoff,* (2011), The proper place of men and machines in language technology: Processing Russian without any linguistic knowledge, Proceedings of the International conference "Dialogue 2011", Bekasovo.

[9] *W. Tutte,* (1984) Graph Theory, Addison-Wesley, Menlo Park, 1984.

Appendix

Proof of (2) Denote by the set of all pairs $(i, j), i \neq j, 0 < i \leq N, 0 < j \leq N$. Opening the brackets we get

$$Z_1 = \prod_{\substack{(h.m):h \neq m \\ 0 \leq h \leq N \\ 0 < m \leq N}} (1 + w_{hm}) = \sum_{V \subset U} \prod_{(h,m) \in V} w_{hm}$$

Each term in the right part corresponds to the weight of the oriented graph with edges $(h, m) \in V$. Since each corresponding graph is present in the sum only once we get the result.

Proof of (3) Opening the brackets we get

$$Z_2 = \prod_{m=1}^{N} \sum_{\substack{(h.m):h \neq m \\ 0 \leq h \leq N}} w_{hm} = \sum_{h_1 \neq 1} \sum_{h_2 \neq 2} \dots \sum_{h_N \neq N} (w_{h_1 1} w_{h_2 2} \dots w_{h_N N})$$

Each term in the right part corresponds to the weight of the corresponding functional graph. Since each corresponding graph is present in the sum only once we get the result.