Learning Syntactic Patterns Using Boosting and Other Classifier Combination Schemas

András Hócza, László Felföldi, and András Kocsor

University of Szeged Department of Informatics
6720 Szeged, Árpád tér 2.
{hocza, lfelfold, kocsor}@inf.u-szeged.hu
http://www.inf.u-szeged.hu

Abstract. This paper presents a method for the syntactic parsing of Hungarian natural language texts using a machine learning approach. This method learns tree patterns with various phrase types described by regular expressions from an annotated corpus. The PGS algorithm, an improved version of the RGLearn method, is developed and applied as a classifier in classifier combination schemas. Experiments show that classifier combinations, especially the Boosting algorithm, can effectively improve the recognition accuracy of the syntactic parser.

1 Introduction

Syntactic parsing is the process of determining whether sequences of words can be grouped together. Syntactic parsing is an important part of the field of natural language processing and it is useful for supporting a number of large-scale applications including information extraction, information retrieval, named entity identification, and a variety of text mining applications.

Hungarian language is customarily defined as an agglutinative, free word order language with a rich morphology. These properties make its full analysis difficult compared to Indo-European languages. Unambiguous marks for the automatic recognition of phrase boundaries do not exist. For example, the right bound of noun phrases could be the nouns as a head, but there is a possibility of replacing noun phrase heads with its modifiers. Determining the left bound of noun phrases is harder than the head, as it could be a determinant element. However, due to the possibility of a recursive insertion, it is so not easy to decide which determinant and head belong together.

This paper introduces the PGS (Pattern Generalization and Specialization) algorithm, an improved version of the RGLearn algorithm [4], for learning syntactic tree patterns, and it describes how one can improve the performance of this learner by applying classifier combination schemas. Classifier combinations aggregate the results of many classifiers, overcoming the possible local weakness of the individual inducers, producing a more robust recognition. After comparing them to related works the results look fairly promising.

This paper is organized as follows. Section 2 summarizes the related works on the topic of syntactic parsing. Section 3 presents the method used for learning

grammar from an annotated corpus. Section 4 then gives a short introduction to classifier combination techniques. The proposed methods are tested in Section 5. After, conclusions and suggestions for future study are given in the last section.

2 Related works

Several authors published results of syntactic parsing especially made for English. Generally the performance is measured with three scores: precision, recall and an $F_{\beta=1}$ rate which is equal to 2*precision*recall/(precision*recall). The latter rate has been used as the target for optimization. Ramshaw and Marcus [7], for instance, built a chunker by applying transformation-based learning ($F_{\beta=1}$=92.0). They applied their method to two segments of the Penn Treebank [6] and these are still being used as benchmark data sets. Tjong Kim Sang and Veenstra [8] introduced cascaded chunking ($F_{\beta=1}$=92.37). The novel approaches attain good accuracies using a system combination. Tjong Kim Sang [9] utilized a combination of five classifiers for syntactic parsing ($F_{\beta=1}$=93.26).

Up till now there is no good-quality syntactic parser available for the Hungarian language. Benchmark data sets for correctly comparing results on Hungarian do not exist yet either. The HuMorESK syntactic parser [5] developed by MorphoLogic Ltd uses attribute grammar, assigning feature structures to symbols. The grammar part employed in the parser was made by linguistic experts. Another report on the ongoing work of a Hungarian noun phrase recognition parser [10] is based on an idea of Abney's [1] using a cascaded regular grammar and it has been tested on a short text of annotated sentences ($F_{\beta=1}$=58.78). The idea of using cascaded grammars seems beneficial, this technique being used in all Hungarian parser developments. A noun phrase recognition parser [4] is applied machine learning methods to produce grammar of noun phrase tree patterns from annotated corpus ($F_{\beta=1}$=83.11).

3 Learning tree patterns

In this section the learning task of syntactic tree patterns will be described which contains the preprocessing of training data, generalization and specialization of tree patterns. An improved version of RGLearn [4] named PGS (Patten Generalization and Specialization) was used as a tree pattern learner. The novelty of PGS is the use of λ parameters which have an influence on the quality of learned tree patterns. The pattern unification method and the search method for best patterns have also been modified.

3.1 Preprocessing of training data

The initial step for generating training data is to collect syntactic tree patterns from an annotated training corpus. The complete syntax tree of sentence must be divided into separate trees and a cascade tree building rules to prepare the

parser to reconstruct it. In parsing, using of context free grammar has a lot of advantages, but the conditions of pattern usage may completely disappear. Some structural information can be salvaged if tree patterns are used. To generate cascaded grammar, linguistic experts have defined the following processing levels for the Hungarian language:

- Short tree patterns of noun, adjective, adverb and pronoun phrases.
- Recursive complex patterns of noun, adjective, adverb and pronoun phrases.
- Recursive patterns of verb phrases.
- Recursive patterns of sub-sentences.

3.2 Generalization of tree patterns

Using the collected tree patterns the syntactic parser is able to reconstruct the tree structure of training sentences. But, in order to perform the syntactic parsing of an unknown text to a fair accuracy, the collected tree patterns must be generalized. Generalization means that the lexical attributes of each tag are neglected except for the POS codes. In this phase the learning problem is transformed into a classification problem. Namely, which set of lexical attributes would supply the best result for the decision problem of tree pattern matching, i.e a given tree structure covers a given example or not. To support the learner, positive and negative examples are collected from a training set for each tree type. The example in Figure 1 shows the complete tree pattern learning process.

3.3 Specialization of tree patterns

Negative examples are the bad classifications of generalized tree pattern and they must be eliminated. Therefore specialization selects each possible lexical attribute from positive examples making new tree patterns and tries to find the best tree patterns with unification.

The initial step of specialization generates all possible new tree patterns extending generalized tree patterns with exactly one attribute from the covered positive examples. The next steps of specialization extends the set of tree patterns with all possible new tree patterns by a combination of each pair of tree patterns. The combination of two tree patterns means the union of their lexical attributes. To avoid the exponential growth of a tree pattern set weak tree patterns are excluded by applying error statistics on positive and negative examples. Here the following score of a given tree pattern is used as the target for maximization:

$$\text{score} = \lambda_1 * (pos\text{-}neg)/pos + \lambda_2 * (pos\text{-}neg)/(pos+neg)$$

where *pos* is the number of covered positive examples, *neg* is the number of covered negative examples and $\lambda_1 + \lambda_2 = 1$.

Fruitful unifications dramatically decrease the negative coverage, resulting positive coverage almost in the same time. The score maximization runs parallel on every positive example. A new tree pattern is stored only if a covered positive

Sentence parts (examples):
1: . . . $({}_{NP}Tf({}_{ADJP}Afp-sn)_{ADJP}Np-sn)_{NP}$ . . .
2: . . . $({}_{NP}Tf({}_{NP}Afp-pn)_{NP}Nc-pa---s3)_{NP}$ . . .
3: . . . $({}_{NP}({}_{NP}Ti({}_{NP}Afs-sn)_{NP})_{NP}({}_{NP}Nc-s2)_{NP}$ . . .
4: . . . $({}_{NP}Tf({}_{ADJP}Afp-sn)_{ADJP}Nc-sn)_{NP}$ . . .
5: . . . $({}_{NP}Tf({}_{ADJP}Afp-sn)_{ADJP}({}_{ADJP}Afp-sn)_{ADJP})_{NP}$ . . .

Generalized pattern (one of four possible): $({}_{NP}T*({}_{ADJP}A*)_{ADJP}N*)_{NP}$
Coverage: positive {1,4}, negative {2,3}, uncovered {5}
Specialized pattern: $({}_{NP}T*({}_{ADJP}A*)_{ADJP}N???n)_{NP}$
Coverage: positive {1,4}, negative {}, uncovered {2,3,5}

In the lexical codes each letter is a lexical attribute, the first one being the part of speech. Notations:
$T*$: determiner, $A*$: adjective, $N*$: noun, $N???n$: noun with a lexical attribute,
?: a letter of any kind, $*$: one or more letters of any kind,
$({}_X,)_X$: beginning and ending of phrase X, NP: noun phrase, $ADJP$: adjective phrase.

Fig. 1. A tree pattern learning example.

example exists where the score of new tree pattern is greater than the previous maximum value. Specialization stops when the current step did not improve any maximum value.

Appropriate setting of λ factors in linear combination can provide the optimal tree pattern set. A greater λ_1 may result in tree patterns with high coverage, while a greater λ_2 may result high accuracy but there is a possibility of low tree patterns appearing with a low coverage.

4 Classifier Combinations

Classifier Combinations are effective tools for machine learning and can improve the classification performance of standalone learners. A combination aggregates the results of many classifiers, overcoming the possible local weakness of the individual inducers, producing a more robust recognition. A fair number of combination schemas have been proposed in the literature [12], these schemas differing from each other in their architecture, the characteristics of the combiner, and the selection of the individual classifiers. From a combination viewpoint, classifiers can be categorized into the following types:

- abstract: the classifier yields only the most probable class label
- ranking: it generates a list of class labels in order of their probability
- confidence: the scores for each class are available

In the following we will assume that the classifiers are capable of generating information of the confidence type.

4.1 Combination schemas

Let x denote a pattern, and $(\omega_1, \ldots, \omega_n)$ the set of possible class labels. The parameters p_i^j will represent the output of the i-th classifier for the j-th class. Furthermore, let $\mathcal{L}(x)$ denote the correct class labelling for each training sample $x \in S$, and $\mathcal{C}_i$ refer to a function that maps the pattern x to the class label assigned by the i-th classifier:

$$\mathcal{C}_i(x) = \omega_k, \quad k = \underset{j}{\operatorname{argmax}}\, p_i^j(x). \tag{1}$$

The combined class probabilities $\hat{p}^j$ are calculated from the corresponding values of classifiers p_i^j according to combination rules described later. The class label $\hat{\mathcal{C}}(x)$ selected by the combiner is the one with the largest probability:

$$\hat{\mathcal{C}}(x) = \omega_k, \quad k = \underset{j}{\operatorname{argmax}}\, \hat{p}^j(x). \tag{2}$$

There are numerous combination rules mentioned in the literature. The traditional combination methods are listed here:
Sum Rule:

$$\hat{p}^j(x) = \sum_{i=1}^{N} p_i^j(x) \tag{3}$$

Product Rule:

$$\hat{p}^j(x) = \prod_{i=1}^{N} p_i^j(x) \tag{4}$$

Max Rule:

$$\hat{p}^j(x) = \max_{i=1}^{N} p_i^j(x) \tag{5}$$

Min Rule:

$$\hat{p}^j(x) = \min_{i=1}^{N} p_i^j(x) \tag{6}$$

Borda Count:

$$\hat{p}^j(x) = \sum_{i=1}^{N} \sum_{\substack{k=1 \\ p_i^k(x) \le p_i^j(x)}}^{n} 1 \tag{7}$$

4.2 Boosting

Boosting[13] was introduced by Shapire as a method for improving the performance of a weak learning algorithm. The algorithm generates a set of classifiers by applying bootstrapping on the original training data set and it makes a decision based on their votes. AdaBoost changes the weights of the training instances provided as input for each inducer based on classifiers that were previously built. The final decision is made using a weighted voting schema for each classifier, whose weights depend on the performance of the training set used to

Adaboost.M1 algorithm

Require: Training Set S of size m, Inducer $\mathcal{I}$
Ensure: Combined classifier C^*
  $S' = S$ with weights assigned to be $1/m$
  for $i = 1 \dots T$ **do**
    $S' =$ bootstrap sample from S
    $C_i = \mathcal{I}(\mathcal{S}')$
    $\epsilon_i = \sum_{\substack{\mathbf{x_j} \in S' \\ C_i(\mathbf{x_j}) \neq \omega_j}}$ weight of $\mathbf{x_j}$
    if $\epsilon_i > 1/2$ **then** reinitialize sample weights
    $\beta_i = \frac{\epsilon_i}{(1-\epsilon_i)}$
    for all $\mathbf{x_j} \in S'$ such $C_i(\mathbf{x_j}) = \omega_j$ **do**
      weight of $\mathbf{x_j}$ = weight of $\mathbf{x_j} \cdot \beta_i$
    end for
    normalize weights of instances to sum 1
  end for
  $C^*(\mathbf{x}) = \underset{j}{\operatorname{argmax}} \sum_{\substack{i \\ C_i(\mathbf{x}) = \omega_j}} \log \frac{1}{\beta_i}$

build it. The Adaboost algorithm requires a weak learning algorithm whose error is bounded by a constant strictly less than 1/2. In the case of multi-class classifications this condition might be difficult to guarantee, and various techniques should be applied to overcome this restriction.

5 Experiments

5.1 Evaluation domain

In order to perform well and learn from the various *Natural Language Processing* tasks and achieve a sufficient standard of *Information Extraction*, an adequately large corpus had to be collected which serves as the training database. A relatively large corpus of texts of various types was collected, called the Szeged Corpus [2]. It has six topic areas of roughly 200 thousand words each, meaning a text database of some 1.2 million words. One of the domain is short business news items issued by the Hungarian News Agency[1].

Initially, corpus words were morpho-syntactically analysed and then manually POS tagged by linguistic experts. The Hungarian version of the internationally acknowledged MSD (Morpho-Syntactic Description) schema [3] was used for the encoding of the words. The MSD encoding schema can store morphological information about part-of-speech determined attributes on up to 17 positions. About 1800 different MSD labels are employed in the annotated corpus. The texts of the Szeged Corpus have been parsed, where annotators marked various type of phrase structures.

[1] MTI, Magyar Távirati Iroda (http://www.mti.hu), "Eco" service.

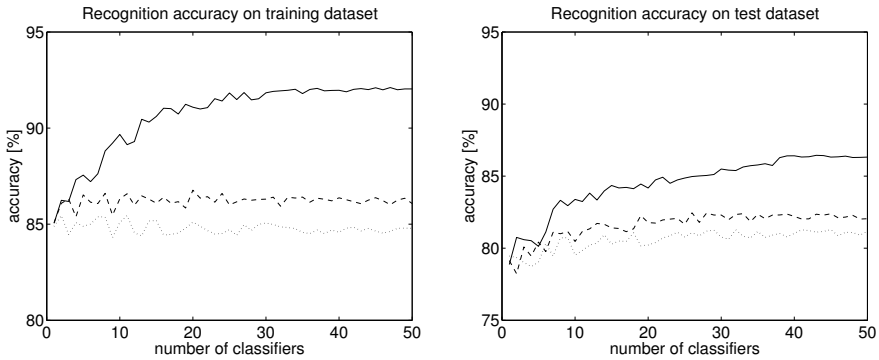

Fig. 2. Classification accuracy of the combination schemas on the training and testing datatset (solid: Boosting, dashed: Sum Rule, dotted: Max Rule).

5.2 Evaluation methods

The training and test datasets were converted from a subset of the business news domain of the Szeged Corpus. During the experiments we generated 50 learners by training the PGS algorithm on different training sets, these sets beeing created by randomly drawing 4000 instances with replacement from the original training set. The λ_1 parameter of the PGS algorithm was selected for optimal performance on the original train dataset. According to the preliminary investigations the PGS algorithm attains its maximal recognition accuracy when λ_1 is set to 0.7, hence this setting was used during the combination experiments.

5.3 Results

The syntactic tree pattern recognition accuracy of the standalone classifier was 78.5 % on the business-news domain using 10-fold cross-validation. Based on their performance the combination schemas can be divided into 3 groups. The schemas Max, Min, and Prod have roughly the same performance: they cannot significantly improve the classification accuracy of the PGS learner. Borda Count and Sum rule can take advantage of combinations, and get an 82 % score on the test data-set. The best classification was obtained by using the Boosting algorithm, achieving an an accuracy of 86 %. Note that the Adaboost.M1 algorithm cannot reduce the training error rate to zero owing to the fact that the Boosting algorithm requires that the weighted error should be below 50%, and this condition is not always fulfilled.

Fig. 2 shows the classification performance for schemas Max, Sum, and Boosting. The graphs show the measured accuracy on the training and test datasets as a function of the number of the applied classifiers in the combination schema. Comparing these results with the performance of the standalone learner, we see that combinations can improve the classification accuracy by some 10%.

6 Summary and future work

In this paper, the authors presented a general machine learning method for syntactic parsing. A new learning method the PGS was introduced as an improved version of the RGLearn algorithm. The accuracy of tree pattern recognition was effectively improved using classifier combination schemas, the best performance being achieved by the Boosting algorithm.

In the future we plan to use the ontological information that can be the extension of a morpho-syntactic description. This system has been primarily applied to the business news domain so far, but we would like to process other Hungarian domains of the Szeged Corpus and to adapt the methods to parsing English texts as well.

References

1. Abney S. (1996) Partial Parsing via Finite-State Cascades, in Proceedings of ESSLLI'96 Robust Parsing Workshop, pp. 1-8.
2. Alexin, Z., Csirik, J., Gyimóthy, T., Bibok, K., Hatvani, Cs., Prószéky, G., Tihanyi, L. (2003) Manually Annotated Hungarian Corpus, in Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguis-tics EACL03, Budapest, Hungary, pp. 53-56.
3. Erjavec, T. and Monachini, M., ed. (1997) Specification and Notation for Lexicon Encoding, Copernicus project 106 "MULTEXT-EAST", Work Package WP1 - Task 1.1 Deliverable D1.1F.
4. Hócza (2004) Noun Phrase Recognition with Tree Patterns, in Proceedings of the Acta Cybernetica, Szeged, Hungary
5. Kis, B., Naszódy, M., Prószéki, G. (2003) Complex Hungarian syntactic parser system in Proceedings of the MSZNY 2003, Szeged, Hungary, pp. 145-151
6. Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993) Building a large annotated corpus of English: the Penn Treebank, Association for Computational Linguistics.
7. Ramshaw, L. A., and Marcus, M. P. (1995) Text Chunking Using Transformational-Based Learning, in Proceedings of the Third ACL Workshop on Very Large Corpora, Association for Computational Linguistics.
8. Tjong Kim Sang, E. F., and Veenstra, J. (1999) Representing text chunks, in Proceedings of EACL '99, Association for Computational Linguistics.
9. Tjong Kim Sang, E. F. (2000) Noun Phrase Recognition by System Combination, in Proceedings of the first conference on North American chapter of the Association for Computational Linguistics, Seattle, pp. 50-55.
10. Váradi T. (2003) Shallow Parsing of Hungarian Business News, in Proceedings of the Corpus Linguistics 2003 Conference, Lancaster, pp. 845-851.
11. R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification,* John Wiley & Sons Inc., 2001.
12. A. K. Jain, *Statistical Pattern Recognition: A Review,* IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 22. No. 1, January 2000.
13. R.E. Shapire, *The Strength of Weak Learnability,* Machine Learnings, Vol. 5, pp. 197-227, 1990.
14. V. N. Vapnik, *Statistical Learning Theory,* John Wiley & Sons Inc., 1998.