KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 1–7

LARGE SCALE EXPERIMENTS WITH FUNCTION TAGGING

MIHAI LINTEAN [(1)] AND VASILE RUS[(2)]

Abstract. We present in this paper large scale experiments with two Decision Trees based approaches to the task of function tagging. The task of function tagging involves labeling certain nodes in an input parse tree with a set of functional marks such as logical subject, predicate, etc. In the first approach, we consider only nodes that are labeled with a functional tag. In the second approach, all nodes are considered whether they are labeled with function tags or not. The non-labeled nodes are simply considered being labeled with the generic tag NON-F. The results obtained on a standard data set are significantly outperforming baseline approaches when the most frequent tag is assigned.

1. Introduction

Syntactic information is an important processing step to many language processing applications such as Anaphora Resolution [11], Machine Translation [6], and Question Answering [16]. Syntactic parsing in its most general definition may be viewed as discovering the underlying syntactic structure of a sentence. The specificities include the types of elements and relations that are retrieved by the parsing process and the way in which they are represented. In this paper we focus on Treebank-style[1] syntactic parsers that retrieve phrases, e.g. NP - noun phrase, VP - verb phrase, S - sentence, and hierarchically organize them in parse trees.

Treebank-style state-of-the-art statistical parsers limit their output to basic structures such as NPs, VPs, PPs (Prepositional Phrases). They are not able to deliver richer syntactic information such as logical subject or predicate although Penn Treebank, the annotated corpus that state of the art parsers use for training, contains annotations for such type of information in the form of function tags and remote dependencies coded as traces. This paper presents experiments with Decision Trees based approaches to augment the output of Treebank-style syntactic parsers with functional information.

In Section 2.2 of Bracketing Guidelines for Treebank II [1], there are 20 function tags grouped in four categories: form/function discrepancies, grammatical role, adverbials, and miscellaneous. Up to 4 function tags can be added to the standard syntactic label (NP, ADVP - Adverbial Phrase, PP, etc.) of each bracket. Those

2000 *Mathematics Subject Classification.* code, code.
Key words and phrases. word, word.

TABLE 1. Categories of Function Tags

Category	Function Tags
Grammatical	DTV, LGS, PRD, PUT, SBJ, VOC
Form/Function	NOM, ADV, BNF, DIR, EXT, LOC, MNR, PRP, TMP
Topicalisation	TPC
Miscellaneous	CLR, CLF, HLN, TTL

tags were necessary to distinguish words or phrases that belong to one syntactic category and are used for some other function or when it plays a role that is not easily identified without special annotation. We rearranged the four categories into four new categories based on corpus evidence, in a way similar to [2]. The new four categories are given in Table 1 and were derived so that no two labels from same new category can be attached to the same bracket.

We present in this paper two approaches to automatically assign function tags to constituents in parse trees. The function tags assignment problem is viewed as a classification problem, where the task is to select the correct tag from a list of candidate tags. In the first approach, we only considered constituents from syntactic parse trees in Treebank that are labeled with functional tags. In the second approach, we considered both labeled and unlabeled constituents. The unlabeled constituents are simply considered being labeled with the generic label **NON-F** (non-functional tag).

This is the first large scale evaluation of Decision Trees based solutions to the task of functional tagging. We used the full data set that Penn Treebank makes available in order to train and test a Decision Trees based functional tagger. In [3], the Decision Trees approach was abandoned (see section *5.2 Why we abandoned decision trees*) due to memory limitations. We addressed the memory issue by using a set of smart preprocessing steps applied to training and test data and by using a High-Performance Computer (IBM AIX System with 64GB of RAM). The preprocessing was necessary in order to reduce the number of distinct values some features have, e.g. lexical based features such as head word. Too many distinct values for these features led to very large Decision Trees that would not fit even in the memory of a High-Performance Computer similar to the one that we used.

The rest of the paper is organized as follows. The next section presents related work in the area of functional tagging and Decision Trees. Section 3 describes the problem we study in this paper while Section 4 presents the generic model we use to solve the problem. Next, the *Experimental Setup and Results* section provides details about the conducted experiments and results. *Conclusions* end the paper.

2. Related Work

Blaheta and Johnson [2] addressed the task of function tagging. They use a statistical algorithm based on a set of features grouped in *trees*, rather than *chains*.

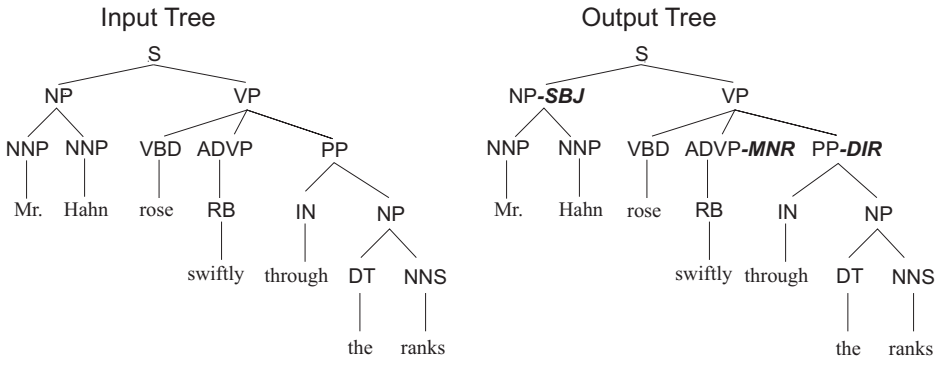

FIGURE 1. A Simple Syntactic Tree

The advantage is that features can better contribute to overall performance for cases when several features are sparse. When such features are conditioned in a chain model the sparseness of a feature can have a dilution effect of a ulterior (conditioned) one.

Previous to that, Michael Collins [7] only used function tags to define certain constituents as complements. The technique was used to train an improved parser.

Related work on enriching the output of statistical parsers, with remote dependency information, were exposed in [10] and [9].

Decision Trees induction has been studied in detail both in the area of pattern recognition and in the area of machine learning [8]. In the vast literature concerning Decision Trees, also known as classification trees or hierarchical classifiers, at least two seminal works must be mentioned, those by Breiman et al. [4] and Quinlan [14]. The former originated in the field of statistical pattern recognition and describes a system, named CART (Classification And Regression Trees), which has mainly been applied to medical diagnosis and mass spectra classification. The latter synthesizes the experience gained by people working in the area of machine learning and describes a computer program, called ID3, which has evolved into a new system, named C4.5 [15].

Decision Trees are mainly used for classification purposes, but they are also helpful in uncovering features of data that were previously unrecognizable to the eye. Thus, they can be very useful in data mining activities as well as data classification. They work well in many different areas, from typical business scenarios to airplane autopilots and medical diagnoses.

3. The Problem

The task of function tagging is to add extra labels, called function tags, to certain constituents in a parse tree. Let us pick as an illustrative example the sentence *Mr. Hahn rose swiftly through the ranks*[1]. A state-of-the-art syntactic parser will generate

[1]This sentence is from Wall Street Journal portion of Penn Treebank.

the parse tree shown on the left hand side in Figure 1. Each word in the sentence has a corresponding leaf (terminal) node, denoting that word's part of speech. For instance, the word *ranks* has NNS as its part of speech (NNS indicates a common noun in plural form). All the other nodes, called internal or non-terminal nodes, will be labeled with a syntactic tag that marks the grammatical phrase corresponding to the node, such as NP, VP, or S.

It is not obvious from such syntactic parse trees which nodes are playing the role/function of logical subject for instance. An user of these parse trees needs to develop extra procedures to detect the roles played by various words or phrases. The task of function tagging, the problem addressed in this paper, is to add function tags to nodes in a parse tree.

4. The Model

We modeled the problem of assigning function tags as a classification problem. Classifiers are programs that assign a class from a predefined set of classes to an instance based on the values of attributes used to describe the instance. We defined a set of linguistically motivated attributes/features based on which we characterized the instances.

Let us analyze the set of features and classes we used to build the classifiers. We used a set of features inspired from [2] that includes the following: label, parent's label, right sibling label, left sibling label, parent's head pos (part-of-speech), head's pos, grandparent's head's pos, parent's head, head. We did not use the alternative head's pos and alternative head (for prepositional phrases that would be the head of the prepositional object) as explicit features but rather modified the phrase head rules so that the same effect is captured in pos and head features, respectively.

The set of classes we used corresponds to the set of functional tags in Penn Treebank. The functional tags are grouped in four categories given in Table 1. The four categories were derived so that no two labels from same new category can be attached to the same bracket.

The above features and classes are used to derive Decision Trees classifiers. The next section describes the experiments we conducted to derive and evaluate the classifiers.

5. Experimental Setup And Results

We trained the classifiers on sections 1-21 from Wall Street Journal (WSJ) part of Penn Treebank and used section 23 to evaluate the classifiers. This split is standard in the syntactic parsing community [5]. The evaluation follows a gold standard approach in which the classifiers' output is automatically compared with the correct values, also called gold values.

The performance measures reported are accuracy and kappa statistic. The *accuracy* is defined as the number of correctly tagged instances divided by the number of attempted instances. *Kappa statistic* [17] measures the agreement between predicted and observed classes, while correcting for agreement that occurs by chance. Kappa

can have values between -1 and 1 with values greater than 0.60 indicating substantial agreement and values greater than 0.80 showing almost perfect agreement. We also report precision, recall, and F-measure for the second experiment when we also consider the non labeled nodes (viewed as negative instances). Such measures are reported on positive instances (true positives are considered correct). *Precision* is the number of correct guesses of tags from one category over the total number of guesses (correct of incorrect) for tags from that category. *Recall* is the number of correct guesses of tags in some category over the actual number of instances that should have a tag from that category. The *F-measure* value is calculated based on Precision and Recall from the following formula:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

To build the classifiers, we used the implementation of Decision Trees in WEKA. WEKA [17] is a comprehensive, open-source (GPL) machine learning and data mining toolkit written in Java. WEKA requires a lot of memory to build the models from large training sets, especially for Decision Trees. A regular machine with 2GB of memory is not sufficient even after the preprocessing steps aimed at reducing the size of data (see below details about preprocessing). We used an IBM AIX High Performance Computer (HPC) system with 64GB of RAM.

5.1. **Data Collection.** To build Decision Trees based classifiers, one must collect training data. The data is a set of problem instances. Each instance consists of values for each of the defined features of the underlying model and the corresponding class, i.e. function tag in our case. Instances are automatically created from Penn Treebank parse trees by simply traversing those trees and for each node extracting the values for each feature and for the class attribute.

Since a node can have several tags there are two possible setups for our classification task. We can define a class as a composed tag of all possible combinations of function tags that can be assigned to a node. A single classifier is generated in this case that would assign one composed tag to a node, i.e. one or more individual functional tags at once. We do not use this setup in this work. It was previously studied by Lintean and Rus [12]. Alternatively, we can try to build four separate classifiers, one for each of the four functional categories described earlier in the paper. Knowing that a node cannot have more that one tag from a given category, each classifier will be used to predict the functional tag from the corresponding category. We focus on this latter setup in this paper.

Some simplifications are necessary to make the task feasible. In those experiments punctuation was mapped to an unique tag PUNCT and traces were left unresolved and replaced with TRACE. Furthermore, two features, **parent's head** and **head**, require special attention. They have as values the words that represent the head

word[2] for a given node in the parse tree. Due to lexical diversity, the two features have a very large set of different values, i.e. words. This leads to very large Decision Trees that cannot be handled by regular computers. We applied a set of transformations aimed at reducing the number of possible values for head-words. Due to space constraints we briefly list some of the techniques: replace all person names with a generic label PERSON_NAME, replace words denoting numbers with a generic word NUMBER, everything is changed to lower cases, eliminate special characters (dot or comma) from the words, stem all the words, i.e. reduce all the morphological related forms to the base form of the word. These transformations reduce the number of distinct values by almost a half for the head and parent's head features from 19,730 to 11,430 and from 14,973 to 8,402, respectively. Because by applying this set of transformations, a property regarding the similarity between the **parent's head** and **head** features might be lost, we added a new feature to the set of features presented at the beginning of this section. The new feature is a Boolean value that shows if the parent's head is the same with the current node's head. In our experiments, we noticed that this new feature slightly improved the performance of the classifier models.

5.2. **Results.** We conducted two types of experiments. In both experiments, the training and test data was divided according to the function tag category (Grammatical, Form/Function, Topicalisation, Miscellaneous). An instance from Treebank that has a composed tag such as *LGS-TMP* would lead to one instance for the Grammatical and Function/Form categories each. We generate four different classifiers, one for each category.

In a first experiment, we considered constituents with functional tags. From each parse tree in Treebank only nodes with functional tags were used to generate training and testing instances. Number of instances for the test data set are given in the second column of Table 2. We obtained 118,483 training instances for Grammatical, 66,261 for Form/Function, 3,751 for Topicalization and 16,630 for Miscellaneous. In the second experiment, we considered all internal nodes in parse trees. We did not generate instances for leaf nodes corresponding to part-of-speech tags. Nodes without a functional tag were assigned the new NON-F value indicating no functional tag. A total of 827,193 training instances and 47,333 test instances were generated.

Table 2 presents the results for the first experiment while Table 3 shows the results for the second experiment. The figures represent results on the test data, i.e. section 23 from Treebank. Each table also includes results for a baseline approach. The baseline approach always assigns the most frequent tag from a given category. For instance, for the Grammatical category the SBJ tag is the most frequent and thus the baseline approach always assigns this tag. For the first experiment, the baseline performance is shown as the second value in the fourth column *Acc./Baseline Acc.*

[2]The head of a node in a syntactic parse tree is the word that gives most of the meaning of the phrase represented by that node. There is a set of deterministic rules to detect the head word of syntactic phrases [13].

TABLE 2. Performance Measures on Decision Trees. Experiment 1.

Category	# Instances	Errors	Acc./Baseline Acc.	Kappa
Grammatical	6907	21	99.70/81.79%	0.9901
Form/Function	3902	639	83.62/35.93%	0.7841
Topicalisation	261	0	100.00/100.00%	1.0000
Miscellaneous	755	4	99.47/84.45%	0.9807

TABLE 3. Compared results. Baseline vs. Decision Trees

Category	Accuracy	Tag	Precision	Recall	F-Measure	Kappa
Baseline	never tag	always choose most likely tag				
Grammatical	86.93%	SBJ	10.53%	80.63%	18.63%	0
Form/Function	91.79%	TMP	3.10%	37.79%	5.74%	0
Topicalisation	99.41%	TPC	0.59%	100.00%	1.18%	0
Miscellaneous	98.44%	CLR	1.31%	84.21%	2.59%	0
Decision Tree results						
Grammatical	98.45%	-	99.19%	90.14%	94.45%	0.9370
Form/Function	95.15%	-	74.19%	52.54%	61.52%	0.6405
Topicalisation	99.87%	-	86.80%	90.40%	88.56%	0.8849
Miscellaneous	98.54%	-	61.75%	23.19%	33.72%	0.3318

For the second experiment, since accuracy is computed on both positive and negative instances the most frequent tag is the newly introduced NON-F label that indicates no function tag.

From the tables we noticed high values for Kappa which suggest that Decision Trees offer predictions that are in high agreement with the true, gold values.

6. CONCLUSIONS

We presented in this paper successful experiments with building Decision Trees from large data sets. The paper shows how good Decision Trees are at predicting function tags when trained on the whole Treebank data set. The proposed methods significantly outperform a baseline approach. We plan to expand our research to explore the feasibility of building one single Decision Tree that would assign all function tags at once. A set of preprocessing step and a re-engineering of the feature set may be necessary for that.

References

[1] M; Katz K Bies, A; Ferguson and R MacIntyre. Bracketing guidelines for treebank ii style. Penn Treebank Project, 1995.

[2] D Blaheta and M Johnson. Assigning function tags to parsed text. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 234–240, Seattle, May 2000.

[3] Don Blaheta. *Function tagging*. PhD thesis, Brown University, August 2003. Advisor-Eugene Charniak.

[4] J; Olshen R Breiman, L; Friedman and C Stone. A comparative analysis of methods for prunning decision trees. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 19(5):476–491, 1997.

[5] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of North American Chapter of Association for Computational Linguistics (NAACL-2000)*, Seattle, WA, April 29 - May 3 2000.

[6] Knight K. Charniak, E. and Yamada K. Syntax-based language models for machine translation. In *Proceedings of Machine Translation Summit IX*, New Orleans, 2003.

[7] M Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, 1997.

[8] D Esposito, F; Malerba and G Semeraro. A comparative analysis of methods for prunning decision trees. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 19(5):476–491, 1997.

[9] V Jijkoun and M De Rijke. Enriching the output of a parser using memory-based learning. In *Proceedings of the ACL 2004*, 2004.

[10] M Johnson. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[11] S. Lappin and H. J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535561, 1994.

[12] M. Lintean and V. Rus. Naive bayes and decision trees for function tagging. In *Proceedings of the International Conference of the Florida Artificial Intelligence Research Society (FLAIRS) 2007*, Key West, FL, May 2007 (in press).

[13] D.M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, February 1994.

[14] J R Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[15] J R Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[16] E. Voorhees. Overview of the trec 2002 question answering track. In *Proceedings of the Eleventh Text Retrieval Conference*, 2002.

[17] E Witten, I ; Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann Publishers, 2005.

[1] Department of Computer Science, Institute for Intelligent Systems, FedEx Institute of Technology, The University of Memphis, Memphis, TN 38152, USA

E-mail address: M.Lintean@memphis.edu

[2] Department of Computer Science, Institute for Intelligent Systems, FedEx Institute of Technology, The University of Memphis, Memphis, TN 38152, USA

E-mail address: vrus@memphis.edu