Naive Bayes and Decision Trees for Function Tagging

Mihai Lintean and **Vasile Rus**
Department of Computer Science
The University of Memphis
Institute for Intelligent Systems
Memphis, TN 38152, USA
M.Lintean@memphis.edu
vrus@memphis.edu

Abstract

This paper describes the use of two machine learning techniques, naive Bayes and decision trees, to address the task of assigning function tags to nodes in a syntactic parse tree. Function tags are extra functional information, such as logical subject or predicate, that can be added to certain nodes in syntactic parse trees. We model the function tags assignment problem as a classification problem. Each function tag is regarded as a class and the task is to find what class/tag a given node in a parse tree belongs to from a set of predefined classes/tags. The paper offers the first systematic comparison of the two techniques, naive Bayes and decision trees, for the task of function tags assignment. The comparison is based on a standardized data set.

Introduction

Syntactic information is an important processing step to many language processing applications such as Anaphora Resolution, Machine Translation, and Question Answering. 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. For example, Treebank-style parsers (Bies *et al.* 1995) retrieve a hierarchical organization (tree) of smaller elements (called phrases, e.g. noun phrases - NP, verb phrases - VP, sentence - S), while Grammatical- Relations(GR)-style parsers explicitly output relations together with elements involved in the relation (e.g., subj(John,walk) explicitly indicates a subject relation between *John* and *walk*).

In this paper, we work in the realm of Treebank-style parsers. Given a sentence as input, these parsers output parse trees. Examples of such parse trees are shown in Figure 1 for the sentence *Mr. Hahn rose swiftly through the ranks.* The most successful Treebank-style parsers are based on statistical models. They use treebanks to derive the parameters of the models. A treebank is a collection of English sentences manually annotated with syntactic information by experts. State-of-the-art statistical parsers are trained on Penn Treebank (Bies *et al.* 1995) and focus on identifying the major phrases such as NP, VP, or S, although Penn Treebank contains annotations for other types of information such function tags (e.g., subject or predicate) and traces[1]. The parsers usually ignore the extra information in order to make the estimation of the parameters of their underlying statistical models more reliable. This paper presents a method to augment the output of Treebank-style syntactic parsers with functional information in the form of function tags.

The paper describes two techniques, one based on naive-Bayes and another based on decision trees, to assign function tags to nodes in parse trees. The function tags assignment problem is viewed as a classification problem, where the task is to find for a given node in a parse tree the correct tag/class from a list of candidate tags/classes. The paper presents a systematic comparison of the two techniques based on evaluations conducted on a standard data set. Further, the two techniques use the same underlying model and same set of data to derive the classifiers which allows for a fair comparison.

We chose naive-Bayes and decision trees for their simplicity and user-friendliness, respectively. Naive-Bayes classifiers make strong assumptions about how the data is generated, and use a probabilistic model that reflects these assumptions. They use a collection of labeled training examples to estimate the parameters of the generative model. Classification of new examples is performed with Bayes' rule by selecting the class that is most likely to have generated the example. The naive Bayes classifier assumes that all attributes of the examples are independent of each other given the context of the class. This is the so-called "naive Bayes assumption". This assumption is wrong in many real-world tasks, yet naive Bayes classifiers often perform very well. This paradox is explained by the fact that classification estimation is only a function of the sign (in binary cases) of the function estimation; the function approximation can still be poor while classification accuracy remains high (Friedman 1997). Because of the independence assumption, the parameters for each attribute can be learned separately, and this greatly simplifies learning, especially when the number of attributes is large (McCallum & Nigam 1998).

[1]Traces are remote dependencies between words that are far apart in a sentence such as the direct object relationship between *call* and *John* in the following sentence: John is the person I called yesterday

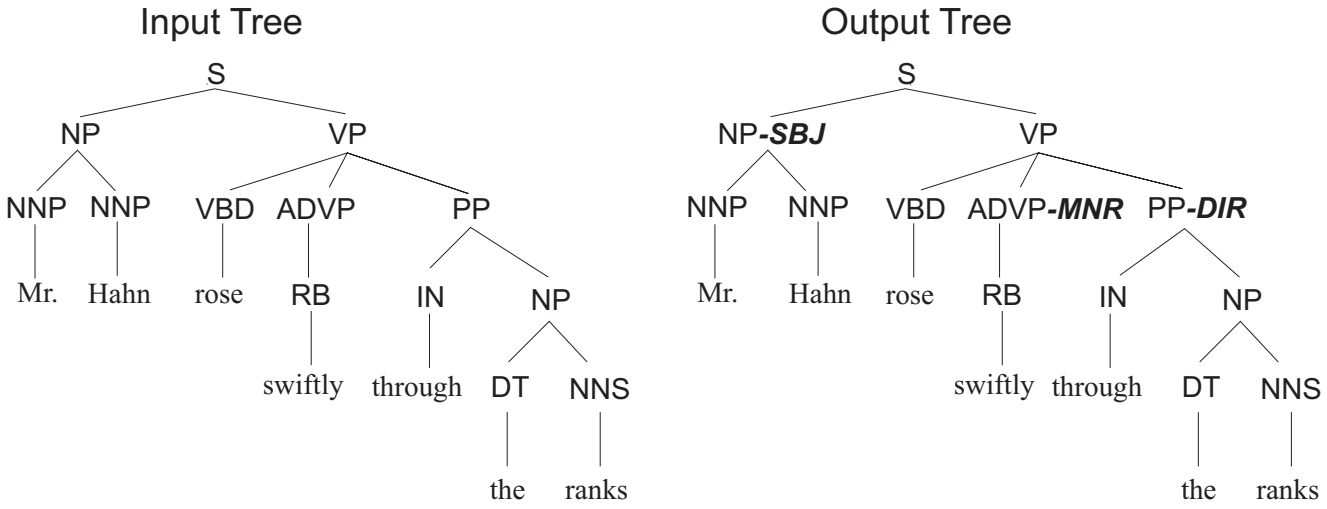

Figure 1: A Simple Syntactic Tree

Decision tree induction in machine learning were largely studied by Quinlan (Quinlan 1986). A computer program, called ID3, implements decision tree induction algorithms. ID3 has evolved into a new system, named C4.5 (Quinlan 1993). 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. One major advantage of decision trees is their generation of a tree model that can be easily interpreted by humans.

The rest of the paper is organized as in the followings. Next, in the *Related Work* section, we analyze previous efforts related to the task of function tags assignment. *The Problem* section clearly defines the problem we address. Details on how we approach the problem are presented in section *The Model*. The *Experiments and Results* section details the evaluation we conducted on a standardized data set. The *Conclusion* section ends the paper.

Related Work

Previous work to address the task of function tags assignment is presented in (Blaheta & Johnson 2000). They use a statistical algorithm based on a set of features grouped in *trees*, rather than *chains*. 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 (Collins 1997) used function tags to define certain constituents as complements. The technique was used to train an improved parser.

Also, there were previous attempts to enrich the output of syntactic parsers with additional information available in Penn Treebank such as dependency information (Johnson 2002; Jijkoun & De Rijke 2004).

There is no previous work, to our knowledge, that attempted to systematically compare naive Bayes or decision trees approaches to the task of function tagging. In (Rus & Desai 2005), preliminary work with a naive-Bayes approach is reported. They use different data preprocessing steps which makes it hard to directly compare with our work. Our preprocessing steps were so developed to reduce the number of values of certain features in our model. The reduction is necessary in order to generate decision trees that can fit in the computational resources available. Too many values for a feature would lead to numerous nodes corresponding to that value in the decision tree, and thus to a large decision tree.

We present in this paper a common framework to address the problem of function tagging and report how naive Bayes and decision trees perform within this framework. The framework is defined by a common underlying model and a common set of preprocessing steps. The model and the preprocessing steps are described later.

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.* [2] A state-of-the-art syntactic parser will generate the parse tree shown on the left hand side in Figure 1. Each word in the sentence has a corresponding leaf (terminal) node, representing 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.

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

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

It is not obvious from such syntactic parse trees which node plays 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.

Technically, the task of function tags assignment is to generate a parse tree that has function tags attached to certain nodes as shown on the right hand side in Figure 1, from an input parse tree that has no function tags such as the one on the left hand side in Figure 1.

The Model

We model 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 define a set of linguistically motivated features based on which we characterize the instances. We automatically generate instances from Penn Treebank and then use them to derive naive Bayes and decision trees classifiers as solutions to the function tags assignment problem.

Let us analyze the set of features and classes we used to build the classifiers.

We used a set of features inspired from (Blaheta & Johnson 2000) that includes the following: label, parent's label, right sibling label, left sibling label, parent's head pos, 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 in our model corresponds to the set of functional tags in Penn Treebank. In Section 2.2 of Bracketing Guidelines for Treebank II (Bies *et al.* 1995), 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, VP, S, etc.) of each node. For instance, a node can have a label such as NP-LGS-TPC to indicate a noun phrase (NP) that has attached to it two function tags, LGS (logical subject) and TPC (topicalisation). Those 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 rearrange the four categories into four new categories based on corpus evidence, in a way similar to (Blaheta & Johnson 2000). 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 node.

The above features and classes are used to derive both naive Bayes and decision trees classifiers. This common underlying model allows for a crisp comparison of the two techniques for the task of assigning function tags. The next section describes the experiments we conducted to derive and evaluate the classifiers.

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 generated classifiers. This split is standard in the syntactic parsing community (Charniak 2000). The evaluation follows a gold standard approach in which the output of classifiers is automatically compared with the correct values, also called gold data.

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* (Witten 2005) measures the agreement between predicted and actual/gold 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. More useful references about the k-statistic estimator can be found online on the Wolfram Mathworld site (http://mathworld.wolfram.com/k-Statistic.html).

Our evaluation is different from the one reported in (Blaheta & Johnson 2000) and thus a direct comparison is not possible. In (Blaheta & Johnson 2000), they considered both nodes with and without functional tags. Due to a large number of nodes with no function tags ($\sim$90%), we only considered nodes with tags to get a better idea of how good naive Bayes and decision trees are at detecting true function tags. In a quick intuitive analysis, the figures reported in (Blaheta & Johnson 2000) are close to our results for similar experiments. However, the reader must note that (Blaheta 2003) was not able to report full results on decision tree due to memory limitations. We do report here performance figures on decision trees. Further, we conducted two types of experiments as opposed to a single type. Our data instances for both training and testing were obtained from perfectly

Table 2: Number of distinct values for the head words

Type of transformation	# Distinct values	
	Head	Parent's Head
Untransformed	19730	14793
Letters only	18742	14380
Letter case insensitive	16675	13373
Stemmed words	12489	9004
Final-Transformation-Set	**11430**	**8402**

parsed trees in Treebank while (Blaheta & Johnson 2000) parse the test data using a state-of-the-art parser and considers only correct constituents in the output when reporting results of the functional tags assignment classifier.

To build the classifiers, we used the implementations of naive Bayes and decision trees in WEKA. WEKA (Witten 2005) is a comprehensive, open-source (GPL) machine learning and data mining toolkit written in Java. We have chosen Weka because it offers the same environment for naive Bayes and decision trees, and thus the comparison would be more reliable. 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 its size (see below details about preprocessing). We have used an AIX High Performance Computer (HPC) system with 64GB of RAM.

Data Collection

Before one can build naive Bayes or decision trees based classifiers, one needs to collect training data. The training 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. The values of the features and class are automatically extracted from Penn Treebank parse trees by simply traversing the trees and for each node with function tags extract values for the defined features.

To generate the training data, we have considered only nodes with functional tags, ignoring nodes unlabeled with such tags. This will allow for a more detailed comparison of the two approaches since nodes with no function tags are significantly outnumbering the ones with function tags. Including the no-function-tag nodes would have led to a less clear picture of how well the two approaches could detect true function tags. We plan to address this issue of detecting whether a node has functional tags or not in the future.

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. 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. Hence, there are two types of experiments we conducted: (1) each instance is assigned a composed tag from the set of all joint-tags categories (2) each instance is assigned a tag from each of four categories. While the first type is more convenient, the second is similar to what Treebank does, i.e. assigning tags from multiple categories.

Some simplifications were necessary to make the task feasible: 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**, have as values the words that represent the head word for a given node in the parsed tree. 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 (Magerman 1994). Due to lexical diversity, the two features have a very large set of different values, i.e. words. That would lead to a very large decision tree that cannot be handled by regular computers.

We've tried different approaches for reducing the number of possible values for head-words based on lexical and morphological transformations, e.g. stemming, grouping some of the words in lexical categories. The changes in number of different values for various transformations are shown in Table 2. The full set of applied transformations are given below.

1. Replace all the words that represent family, female or male names with labels indicating the following three generic classes: Family Name, Male Name and Female Name. We used three dictionaries to categorize proper nouns into one of the tree classes.
2. Replace words denoting numbers, identified by the part of speech label CD, with the following class labels: Number, Fraction (e.g. 3/4), Time (e.g. 10:30) and Date Period (e.g. 1987-19995).
3. Change everything to lower case. This is to prevent differences between words that are at the beginning of sentences and same words appearing somewhere in the middle of a sentence.
4. Replace numbers from complex words that are composed of a number and a word. For example words like

Table 3: Performance Measures on Naive Bayes and Decision Trees. Experiment 1.

Category	Training Data			Test Data		
	# Instances	Errors	Performance	# Instances	Errors	Performance
Naive Bayes	Kappa statistic = 0.8206			Kappa statistic = 0.8080		
All Categories	202311	26655	86.82%	11634	1644	85.87%
Grammatical	118483	3788	96.80%	6907	279	95.96%
Form/Function	66261	17940	72.92%	3902	1109	71.58%
Topicalisation	3715	131	96.47%	261	14	94.64%
Miscellaneous	16630	3966	76.15%	759	187	75.36%
Decision Trees	Kappa statistic = 0.8751			Kappa statistic = 0.8301		
All Categories	202311	18472	90.87%	11634	1444	87.59%
Grammatical	118483	2007	98.31%	6907	187	97.29%
Form/Function	66261	12689	80.85%	3902	1019	73.88%
Topicalisation	3715	180	95.15%	261	16	93.87%
Miscellaneous	16630	3911	76.48%	759	268	64.69%

10%-used, *80%-controlled* or *mid-1987*, were changed to *Percent-used*, *Percent-controlled* and *mid-YearPeriod*.

5. Eliminate special characters (dot or comma) from the words, because they are used as special characters by the classification program.
6. Stem all the words, i.e. reduce all morphological variations of a word to the base form of the word. For instance, both *go* and *went* would be mapped to *go*.

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 improves the performance of the classifiers.

Results

The results of our experiments are shown in Tables 3 and 4. Table 3 shows the results for the first type of experiments in which we used as classes composed tags. A single classifiers is generated from the training data. The top half of the table presents the results for naive Bayes. The bottom half shows the decision trees results. For each half, the row labeled *All Categories* gives the overall performance figures as reported by WEKA. The other rows, one for each function tags category, give figures for the performance of the classifiers for tags belonging to that category. For this first type of experiment we computed the per category figures by simply breaking the predicted composed tags into individual tags and check if there is a matching individual tag in the correct composed tag. For instance, if the classifiers predicted *LGS-DIR-TPC* and the correct composed tag is *LGS-TMP-DIR* then only the LGS and TPC individual tags that correspond to the Grammatical and Form/Function categories are hits. The Kappa statistic is reported only for the *All Categories* row since WEKA only reports it for the entire data set.

Table 4 shows the results for the second type of experiments. In these experiments, we divide the training and test data per category of function tags. Each instance has an associated class in the form of an individual function tag. 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 thus generate four different classifiers, one for each category.

From the tables we notice high values for Kappa which suggest that both naive Bayes and decision trees offer predictions that are in high agreement with the true, gold values. We can also see that the Form/Function category has considerably lower performance values than the other categories. This is mainly because there are more functional tags that can be predicted; and also because of the complex nature for some of these tags (e.g. TMP), which makes them harder using our chosen set of features.

While the results are close for both naive Bayes and decision trees, decision trees based results are better for most of the function tags categories and types of experiments. The reason why decision trees outperformed naive Bayes, which is somewhat expected, is that decision trees are more complex predictive models than naive Bayes. Their greater predicting power comes at the expense of more resources needed, mainly memory. This was the main issue that drove us to use the HPC system in our experiments. Blaheta (Blaheta 2003) attempted to use decision trees on regular machines but gave up due to memory limitations. Thus, our experiments reported in this paper are the first successful large scale experiments on functional tagging.

Conclusion

We presented in this paper a comparison of naive Bayes and decision trees techniques for the task of assigning function tags. Our experiments show that the two techniques are promising and offer high performance. The results reported are on perfectly parsed trees from the Penn Treebank corpus.

Table 4: Performance Measures on Naive Bayes and Decision Trees. Experiment 2.

Category	Training Data				Test Data			
	# Instances	Errors	Perf	Kappa	# Instances	Errors	Perf	Kappa
Naive Bayes								
Grammatical	118483	546	99.54%	0.9841	6907	69	99.00%	0.9680
Form/Function	66261	13083	80.25%	0.7516	3902	798	79.55%	0.7366
Topicalisation	3751	0	100.00%	1.0000	261	0	100.00%	1.0000
Miscellaneous	16630	132	99.21%	0.9326	759	3	99.60%	0.9856
Decision Trees								
Grammatical	118483	421	99.64%	0.9877	6907	21	99.70%	0.9901
Form/Function	66261	8929	86.52%	0.8271	3902	639	83.62%	0.7841
Topicalisation	3751	0	100.00%	1.0000	261	0	100.00%	1.0000
Miscellaneous	16630	95	99.43%	0.9494	755	4	99.47%	0.9807

References

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

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

Blaheta, D. 2003. *Function tagging*. Ph.D. Dissertation, Brown University. Advisor-Eugene Charniak.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of North American Chapter of Association for Computational Linguistics (NAACL-2000)*.

Collins, M. 1997. 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*.

Friedman, J. 1997. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* 1(1):55–77.

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

Johnson, M. 2002. 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*.

Magerman, D. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. Dissertation, Stanford University.

McCallum, A., and Nigam, K. 1998. A comparison of event models for naive bayes text classification. *Workshop on Learning for Text Categorization, AAAI*.

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

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

Rus, V., and Desai, K. 2005. Assigning function tags with a simple model. In *Proceedings of Conference on Intelligent Text Processing and Computational Linguistics (CICLing) 2005*.

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