Automated Learning Applied to Functional Argument Identification

Vasile Rus

Department of Computer Science, The University of Memphis, USA
Institute for Intelligent Systems, Fedex Institute of Technology, USA

This paper reports experiments on applying machine learning for identifying functional arguments of verbs such as logical subjects. In particular, it is shown that using decision trees for functional arguments identification is beneficial. The paper also argues that linguistically-motivated features gathered from a large corpus can capture functional information.

Keywords: logic form, machine learning, decision trees, knowledge representation.

1. Introduction

Syntactic functional information is vital for advanced text understanding technologies such as information extraction, machine translation, question answering and others.

Treebank II [18] includes functional tags as part of its annotation tags. In spite of this, modern parsing technologies generated from Treebank only offer surface syntactic information in the form of a bracketed representation in which main constituents and major structural phrases of sentences are identified. Null elements and markers to co-indexed elements are not handled at all, thereby leading to major gaps in the relational information available to possible users. Moreover, from the bracketed representation produced by parsers one can easily identify the surface syntactic subject. Nevertheless, the logical subject requires further processing, as illustrated in Table 1: the bracketed form for the two examples is the same (it was generated with a state-of-the-art statistical parser) and thus a surface level pattern that identifies as subject the first NP in a (S (NP VP))[1] phrasal structure would wrongly label *something* as the subject of *tell*[2].

Moreover, for verbs in coordinations the parsers are not able to correctly distinguish among which arguments are shared and which are not. This latter issue is illustrated by arguments *ball* and *newspaper* in Table 2: *ball* is shared by *throw* and *catch* as opposed to what the bracketed form encodes.

Sentence	Bracketed Form	Grammatical Function

Something told John.	(S (NP (NN something)) (VP (VBD told)) (NP (NNP John)))	l-sbj=John
		d-obj=something

John told something	(S (NP (NNP John)) (VP (VBD told)) (NP (NN something)))	l-sbj=John
		d-obj=something

Table 1. Examples of similarly bracketed sentences by state of the art parsers but with different underlying logical structure.

[1] NP stands for noun phrase, VP for verb phrase and S for sentence.
[2] *Something told John* is the inverted form of *John told something* a frequent sentential form in Treebank.

Sentence	Bracketed Form	Shared Grammatical Function ?
John throws and catches the ball .	(S (NP (NNP John)) (VP (VP (VBZ throws)) (CC and))) (VP (VBZ catches) (NP (DT the) (NN ball)) (. .)))	d-obj=ball shared=yes
John eats and reads the newspaper .	(S (NP (NNP John)) (VP (VP (VBZ eats)) (CC and))) (VP (VBZ reads) (NP (DT the) (NN newspaper) (. .)))	d-obj=newspaper shared=no

Table 2. Examples of shared and non-shared direct objects for verb in coordination for which the sharing problem cannot be solved only looking at the output of state-of-the-art parsers.

To overcome those drawbacks of modern parsing technology, novel methods are necessary in order to offer accurate, robust and scalable solutions to the problem of finding syntactic functional information.

In this work a model is introduced and further used to induce automated tools able to detect functional information (logical) in English sentences. The tools are obtained using the C4.5 package for decision tree induction.

In addition, this paper argues that linguistically motivated features gathered from a large corpus can capture functional information.

The paper is organized as follows. The next section presents related work. In Section 3 we describe our approach and Section 4 details experiments and results. The following section compares our approach with similar approaches and Section 6 describes the impact of our solution on the task of logic form identification. We then conclude the paper.

2. Related Work

Usually, when syntactic information is used to study a certain linguistic problem, people either use the bracketed form and are happy with surface level syntactic information or have their own pattern-based methods, which lack generality and scalability: in [25] a general method for word sense disambiguation is presented which yields high performance by taking advantage of full sentential context including raw surface syntactic information provided by a parser. Lapata in [15] proposed a technique which acquires alternating verbs from large balanced corpora by using partial-parsing methods. As part of the process, syntactic patterns to guess, i.e. heuristics, the double object frame, are applied on top of a parser's output: if the syntactic pattern contains at least two proper names adjacent to each other (e.g. *killed John Kennedy*), then reject. Similarly, in [26], a set of heuristics are used to find counts of transitive frames for verbs: a number, a pronoun, a determiner, an adjective, or a noun were considered to be indication of a potential object of the verb.

We already mentioned that Treebank II includes functional tags in its annotation guidelines (about 20 of them that can be appended to constituent labels, e.g. NP-LGS to denote a noun phrase, which is also a logical subject). Those tags were manually inserted in Treebank.

Attempts to enrich the output of parsers with information available in treebanks were described by Johnson [12] and Collins [6]. They develop methods for recovering non-local dependencies in phrase trees. Jijkoun [11] uses pattern-matching methods to solve non-local dependencies in parse trees.

An attempt to address the issue of assigning functional tags was made in [2]. They use a statistical algorithm for assigning the 20 function tags in Treebank (classified in four categories). The reported precision over all four categories is 87.17% on text already parsed. In our work, we limit ourselves to assigning only grammatical labels (one of the four categories in [2]).

The present work is similar to the approaches to the problem of shallow semantic parsing presented in [21] [27] [9] [4]. Shallow semantic parsing is the process of annotating texts with semantic roles specified, either using predicate specific labels [3] or predicate independent labels [13]. They address the problem of shallow semantic parsing as a classification problem and use different machine learning methods to induce a classifier (Support Vector Machines, Decision Trees). Our work is similar to those approaches in several ways: (1) we address the task of detecting logic roles (as opposed to semantic roles) as a classification problem (2) we use a set of features similar, to some extent, to those used by the mentioned studies and (3) the induced classifier plays an important role in a natural language-based knowledge representation: the Logic Form [20] and Propbank [13] respectively.

The PropBank project adds a layer of semantic annotation to the Penn English TreeBank. It provides a consistent argument labeling of predicates, particularly verbs, participial modifiers and nominalizations. In the sentence *John broke the window*, the *breaking* event is described by the following argument structure **break(John, window)**. In PropBank, the arguments of the verb are labeled sequentially from ARG0 to ARG5, where ARG0 is usually the subject of a transitive verb; ARG1, its direct object, etc. Adjunct arguments are also marked for temporals and locatives.

In [20] a natural language-based knowledge representation is presented, namely Logic Form, which requires logic functional arguments for its predicates. In [22] the logic functional arguments were detected using a set of structural patterns that were well suited for small English sentences (definitions of concepts in electronic dictionaries), but for open text that approach lacks scalability.

LF is first order, syntactically simple, logic and was used in many language processing applications. Davidson [8] proposed the predicate treatment of verbs and then Hobbs [10] applied this concept to automated text processing, particularly interpretation of texts. Rus [22] employed LF to Question Answering, to search answers that were not explicitly stated in supporting documents.

In LF, a predicate is generated for every noun, verb, adjective or adverb. The name of the predicate is a concatenation of the word's base form and its part-of-speech.

An example on how this is done is illustrated for the following sentence:

The Earth provides the food we eat every day.

Its corresponding logic form (LF) is:

Earth:n_(x1) & provide:v_(e1, x1, x2) & food:n_(x2) & we:n_(x3) & eat:v_(e2, x3, x2; x4) & every:a_(x4) & day:n_(x4)

In the example given above, the verb *provides* is mapped onto the predicate *provide:v*, where *provide* is the base form for *provides* and *v* stands for verb (the part of speech of *provide*). Arguments for predicates are of two types: *e* - for events specified by verbs, *x* - for entities. The LF of the entire sentence is the conjunction of individual predicates.

The argument's position is also important as it encodes syntactic information: the second argument for a verb is syntactic subject, the third is direct object and the fourth is indirect object. For instance, the second argument of the predicate *provide:v_(e1, x1, x2)* is *x1* (Earth), the subject of the *providing* event. Arguments after ; (semicolon) are adjuncts (arguments which are not mandatory to convey the message of the sentence — such as time or place) and were introduced in [23].

The Role Assignment (RA) Problem

We look at a few examples that will show the complexity of the task of identifying functional arguments of verbs, also called the role assignment problem. Role assignment is an important step for Logic Form Identification (LFI), the task of mapping English sentences onto LF.

In the following sentence:

The judges hear or try a court case.

there are two verbs *hear* and *try* in a coordination (indicated by preposition *or*) and one direct object *a court case*. The issue is whether the direct object is shared by the two verbs or not. In other words, out of two possible interpretations, which one is correct:

I1: *The judges (hear or try) a court case.*
LF1: judge:n_ (x1) & hear:v_ (e1, x1, x2) & or (e3, e1, e2) & try:v_ (e2, x1, x2) & nn (x2, x3, x4) & court:n_ (x3) & case:n_ (x4)
I2: *The judges hear or (try a court case).*
LF2: judge:n_ (x1) & hear:v_ (e1, x1) & or (e3, e1, e2) & try:v_ (e2, x1, x2) & nn (x2, x3, x4) & court:n_ (x3) & case:n_ (x4)

In I1, the compound noun *court case* is shared since *hear* and *try* are bracketed together. In I2, the direct object is *bracketed* with the second verb (the verb *hear* being in an intransitive subcategorization frame). For this particular instance the correct interpretation is I1.

An example in which the direct object should not be shared is:

They steal or commit a violent act.

It is very unlikely to state *steal a violent act* and thus, the correct interpretation is:

I: *They steal or (commit a violent act).*
LF: they (x1) & steal:v_ (e1, x1, x2) & or(e2, e1, e3) & commit:v_ (e3, x1, x2) & violent:a_ (x2)

Here is an example of two verbs that do not share the subject, a situation which might be interpreted otherwise at first sight.

An aircraft that has a fixed wing and is powered by propellers or jets.

The reason why the noun *aircraft* does not play the role of subject for both verbs *has* and *power* is the changing of voice in coordination. The second verb has its subject introduced by preposition *by*.

This paper addresses the RA problem and demonstrates how a machine learning method can offer a robust solution.

3. Approach

Our approach is to address the RA problem as a classification task: given a verb in a sentence and a candidate phrasal head, find the most appropriate syntactic role that the head plays. The set of possible roles contains: subject, direct object, indirect object, prepositional object or norole (a value which indicates that the candidate head does not play any role for the given verb). To preview our results, we demonstrate that combining a set of indicators automatically extracted from large text corpora provides good performance.

The key to any automatic classification task is to determine a set of useful features for discriminating the items to be classified. The features of our model are presented below. Each feature is accompanied by a small description explaining the rationale of picking it.

Head The candidate head could indicate whether it is an appropriate filler for a specific syntactic role of a verb. From the previous examples, we know that *act* cannot stand as an object for *steal*. Similarly, the verb *write* takes as subject only a person, or a referent to a person, such as a personal pronoun (persons are able to write).

Lexical Category of Head A noun is most likely to be direct object of, say, the verb *give*, and a pronoun the subject or indirect object of the verb *write*.

Voice As we saw in the last example of the previous section, the voice of a verb can play an important role in deciding what head word is the subject. In case the verb is in passive voice, the subject is most likely the prepositional object of the following *by*, if any such preposition is present in the sentence. We have a dozen patterns to detect the voice of the verb.

Type of Clause We refer here to the type of sentence (S, SINV) that includes the target verb. An example is the main clause in *I do not believe it, said Peter DaPuzzo, head of retail equity trading stock prices during program-dominated trading.* where *said John* is an inverted sentence in which the subject follows the verb, as opposed to the most common case *John said.* The type is read from the parse tree by following links from verb to parent up the parse tree until a S (S, SINV, SBAR) node is found.

Position of Head If the head is after or before the verb and at what distance, in terms of words (punctuation is ignored when the position is determined).

Those features could be automatically extracted from a large corpus, either manually annotated, or automatically generated.

The basic steps of the feature extraction method are outlined below:

```
procedure ExtractFeatures(Sentence)
        – Generate a full syntactic parse tree for the Sentence
        – Stem the words in the sentence
    for verb in Sentence do
        – Extract the set of features for each node in the tree relative to the verb
        – Classify each node as norole or as logical subject
    end for
```

4. Experimental Setup and Results

There are three major issues that we need to address before doing any kind of experiments: what verbs to focus on, where should we gather training data from, and what machine learning algorithm to use. In the next few paragraphs we provide our answers for each of those issues.

Previous work on verb meaning research, such as [14] and [28], reported experiments on a set of 14 target verbs that exhibit multiple argument patterns: *ask, begin, believe cause, expect, find, give, help, like, move, produce, provide, seem, swing*. We adopted those 14 verbs since we believed it would be a good starting point to have a small set of verbs with many argument ambiguities, thus balancing manageability of the experiments against a set of challenging verbs.

Next, we looked for a corpus as a source of training data. Treebank [18] is a good candidate, because it contains role annotations. We started by developing patterns for tgrep, a tree retrieval pattern-based tool, to identify sentences containing target verbs from Wall Street Journal (WSJ) corpus and used the online form to retrieve the data (`http://www.ldc.upenn.edu/ldc/online/treebank/`). The set of sentences previously obtained is further processed: a stemmer is applied to obtain the stem of individual words[3] and then the target verb is identified and features extracted. One or more training examples (positive and negative) are generated from a sentence (see the next section).

As learning paradigm, we opted for decision trees. Decision Trees are a popular method to approach classification problems. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent rules. Rules can readily be expressed so that humans can understand them.

The C4.5, an algorithm for decision tree generation and an extension of ID3, is used to generate a classifier for the RA problem. The algorithm ID3 uses top-down induction of decision trees.

Given a set of classified examples, a decision tree is induced, biased by the information gain measure, which heuristically leads to small trees. The examples are given in attribute-value representation. The set of possible classes is finite.

Only tests that split the set of instances of the underlying example languages, depending on the value of a single attribute, are supported. Depending on whether the attributes are nominal or numerical, the tests either have a successor for each possible attribute value, or split according to a comparison of an attribute value to a constant, or depending on if an attribute value belongs to a certain interval or not.

The algorithm starts with the complete set of examples, a set of possible tests and the root node as the actual node. As long as the examples propagated to a node do not all belong to the same class and there are tests left, a test with highest information gain is chosen, the corresponding set of successors is created for the actual node and each example is propagated to the successor given by the chosen test. ID3 is called recursively for all successors.

Some of the additional features of C4.5 over ID3 are: incorporation of numerical (continuous) attributes, nominal (discrete) values of a single attribute may be grouped together to support more complex tests, post-pruning after induction of trees, e.g. based on test sets. In order to increase accuracy, C4.5 can deal with incomplete information (missing attribute values).

[3] The stem of a word is its base form, for instance the stem of *children* is child

One problem with decision trees is their tendency to overfit the training data, i.e. if tested on training cases from which it was constructed, it may give a poor estimate of its accuracy on new cases. The true predictive accuracy of the classifier can be estimated by sampling, or by using a separate test file. Either way, the classifier is evaluated on cases different from the ones used to build it. However, this estimate can be unreliable, unless the numbers of cases used to build and evaluate the classifier are both large.

One way to get more reliable estimate of predictive accuracy is by *k-fold cross validation*. The cases are divided into *k* blocks of roughly the same size and class distribution. For each block, a classifier is constructed from the cases in the remaining blocks and tested on the cases in the hold-out block. The error rate of a classifier produced from all of the cases is estimated as the ratio of the total number of errors on the hold-out cases and the total number of cases. Here, we predict the accuracy of our induced classifiers using *k=10* or 10-fold cross validation.

In the following sections, we present two major experiments: (1) using the set of features presented before and (2) adding the verb as an extra feature. Each experiment has two subexperiments: (i) traces in parse trees from Treebank II are not solved and thus the training and test data sets do not include examples for those traces and (ii) traces are solved and the corresponding examples are added to the training and test data sets.[4]

Experiment 1

In this experiment we focus on identifying a specific syntactic role for a specific verb: the role values can be either **subject** or **none**.

In Figure 1 a parse tree is provided for the sentence: *Chris knew yesterday that Terry would catch the ball.* The numbers below individual words is the index of that word in the sentence, starting with 0. We will use this example to illustrate our solution.

We pick a verb, say *know*, and then from annotated corpora, training data is generated. For each word in the sentence a positive example is obtained if the word is the subject of the verb. Examples from different verbs are not mixed. Table 3 contains results for the first trial in which traces are not resolved (examples with traces are eliminated from training data). We pay special attention to punctuation (time stamps such

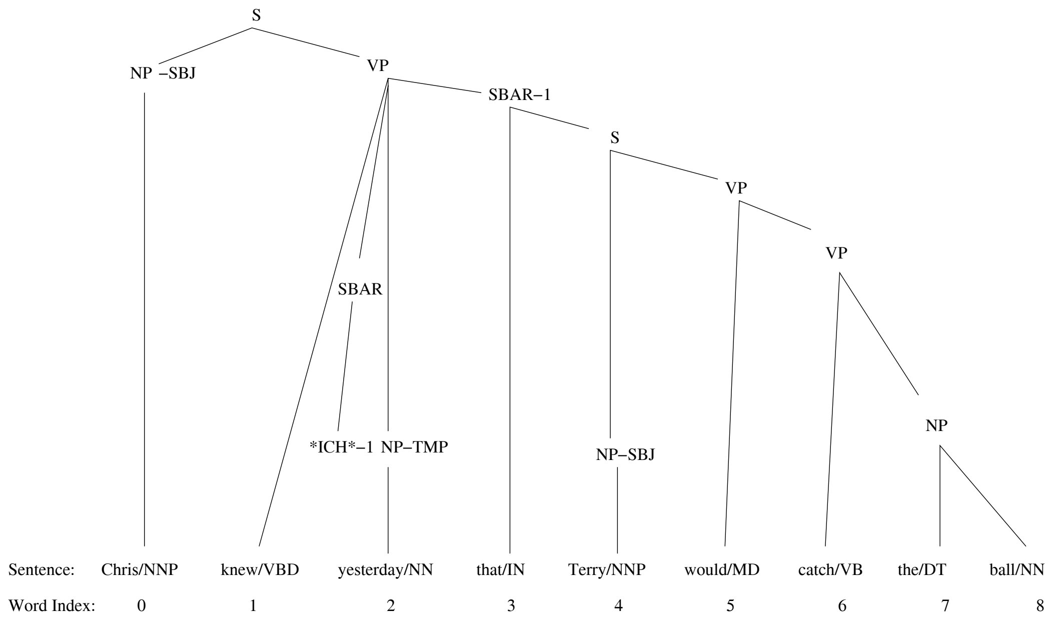
S
NP –SBJ
VP
SBAR–1
S
VP
VP
SBAR
ICH–1 NP–TMP
NP–SBJ
NP
Sentence: Chris/NNP knew/VBD yesterday/NN that/IN Terry/NNP would/MD catch/VB the/DT ball/NN
Word Index: 0 1 2 3 4 5 6 7 8

Fig. 1. An example of parsed sentence in Treebank II.

[4] Traces are artificial links introduced in Treebank to accomodate the bracketed representation to remote dependencies. An example is the remote relation between the verb *know* and its direct object, introduced later in the sentence, as a relative clause *SBAR-1* (Figure 1). To indicate the relationship, a new tag – *ICH-1* is introduced immediately after the verb which is co-indexed with the relative clause.

Verb	Training Size	Errors(%)	Estimate Errors(%)	Errors before Pruning
ask	6968	6.8	8.1	7.7
begin	6691	7.7	8.7	8.1
believe	8766	9.0	9.9	9.8
cause	5990	8.8	10.0	9.7
expect	27792	7.8	8.2	7.9
find	6286	7.2	8.5	7.9
give	11764	6.9	7.7	8.1
help	13359	7.7	8.4	9.1
like	18295	7.4	8.0	8.6
move	10885	7.5	8.2	8.3
produce	9579	7.4	8.2	8.3
provide	10000	7.0	7.7	8.0
seem	7536	7.3	8.4	7.8
swing	1248	9.1	9.8	7.1
all	145159	7.5	7.7	7.8
all+no-head	145159	8.5	8.8	8.9

Table 3. Errors reported by the induced decision trees with 10-fold cross validation.

as 10:40 are changed to 1040) and numbers (350,000 is changed to 350000) to comply with the input requirements for the C4.5. learner. In [19], commas, dots and columns are assigned a special notation (CO and DO respectively) but that was necessary because commas, dots and columns are features in their learning model (they used Timbl [7] which accepts C4.5-like input). For some sentences there is no logical subject identified, because the logical subject is unspecified.

Examples of training data for the sentence in Figure 1 are given below as one training instance per line:

know, Chris, NN, active, S, -1, subject
know, that, SBAR, active, S, 1, none
know, yesterday, NN, active, S, 2, none
know, that, IN, active, S, 3, none
know, Terry, NN, active, S, 4, none
know, catch, VB, active, S, 6, none
know, ball, NN, active, S, 8, none
catch, Chris, NN, active, S, -7, none
catch, know, VB, active, S, -6, none
catch, that, IN, active, S, -5, none
catch, yesterday, NN, active, S, -4, none
catch, that, IN, active, S, -3, none
catch, Terry, NN, active, S, -2, subject
catch, ball, NN, active, S, 2, none

Entries for traces, when considered, have the corresponding lexical head and lexical category features borrowed from the constituent where a trace is resolved. In our example, the training entry for ICH has *that* as its lexical entry and *SBAR* as its category.

In Table 3 the line having *all* in the verb column contains results when training examples of all target verbs were considered together in a single experiment, say *all-verb*. The training time becomes larger, with larger training sets, while the error and estimated error are similar (7.5 – 7.7%). The *all-verb* case illustrates a more general approach in which for all target verbs we generate a single decision tree, as opposed to having a single decision tree for each verb. The last line in the table shows results when the head feature is ignored. There is a small increase in the error rate (1%), but a simpler, less-lexicalized model is obtained.

Verb	Training Size	Training Size Increment	Errors(%)	Estimate Errors(%)	Errors before Pruning
ask	7886	13%	6.9	8.2	7.6
begin	7184	7%	7.7	8.7	8.1
believe	9624	10%	8.5	9.6	9.3
cause	6563	10%	9.2	9.5	9.7
expect	29613	7%	7.8	8.3	8.2
find	6902	10%	7.0	8.4	7.8
give	12766	9%	6.6	7.4	7.8
help	14531	8.7%	7.4	8.3	9.3
like	19811	8.2%	7.4	8.1	8.0
move	11629	7%	7.1	8.0	8.6
produce	10121	6%	6.9	7.8	8.3
provide	10754	8%	6.8	7.6	7.9
seem	8071	7%	7.2	8.4	7.7
swing	1298	4%	8.6	9.2	7.4
all	160056	10%	7.3	7.5	7.6
all+no-head	160056	10%	8.0	8.3	8.4

Table 4. Errors when traces are solved.

Table 4 contains error rates for a trial in which traces in the corpus are resolved. The feature extraction method includes the new step of *Solve traces*:

```
procedure ExtractFeatures(Sentence)
        - Generate a full syntactic parse tree for the Sentence
        - Stem the words in the sentence
        - Solve traces
        - Identify the target verb or all verbs
    for verb in Sentence do
        - Extract the set of features for each node in the tree relative to the verb
        - Classify each node as norole or as logical subject
    end for
```

Double link traces, such as the links between indexes 3-2-1 in the following example, are ignored: (SBARQ (WHNP-1 Who) (SQ was (NP-SBJ-2 *T*-1) (VP believed (S (NP-SBJ-3 *-2) (VP to (VP have (VP been (VP shot (NP*-3))))))))). The number of training examples generated for each verb is larger (on average there are 10% more examples) and resulting error rates are mixed: it drops for *believe*, it increases for *ask*, and it remains the same for *begin*. Overall, there is a *0.1%* decrement in the error rate (see line *all*). The last line in the table shows results when the lexical information about the word that plays the role is ignored. As we notice, there is only a marginal increase in the error rate. It seems that in the absence of deeper semantic information about the word itself (for example the semantic class for objects used to specify selectional restrictions) there is not much impact on the original model by the lexical feature.

Experiment 2

In this second experiment, training examples from all chosen verbs are put together and each example has the verb as a feature. We would like to explore the impact on performance of the verb itself. As shown in Table 5, there is no change in the error rates when the verb is considered for the all-verb experiment. This, somehow surprising result, may be explained with the small set of verb we used, only 14, for our experiments. There is a small increase in the

Verb	Training Size	Errors(%)	Estimate Errors(%)	Errors before Pruning
all+verb	145159	7.5	7.7	7.8
all+verb+no-head	145159	8.3	8.6	8.7

Table 5. Error rates with the verb as a feature and no traces.

Verb	Training Size	Training Size Increase	Errors(%)	Estimate Errors(%)	Errors before Pruning
all+verb	160056	10%	7.3	7.5	7.7
all+verb+no-head	160056	10%	7.8	8.2	8.5

Table 6. Error rates with the verb as a feature and no traces.

error rate when the head of the candidate word is eliminated (see line *all+verb+no-head*). The same trends can be seen when the training set is expanded with examples containing solved traces, as illustrated in Table 6.

Class-based Generalization

An improvement in performance was obtained by generalizing examples with named entities as head feature. For instance, the *IMA* and the *Department of Trade and Industry* were changed to group, *Angelo* and *Keneth Roman* to person and *Arizona* to location. A NE component was used to recognize person names (PER), organizations/groups (ORG) and places/locations (LOC). We applied this technique on training data for the verb *ask* and obtained an improvement in performance of about 2.19%. As an alternative, the person category could be mapped into a personal pronoun. This would help especially when new names (unseen data) are encountered. For common nouns, a related solution could be implemented using a general English taxonomy, such as WordNet, similar to what Li and Abe [16] did for generalizing case frames. Each common noun would be replaced with more general concepts, carefully chosen from the hierarchy. An important issue for such an approach is the set of classes/general concepts to be used.

The class-based generalization method leads to a smaller training set and, consequently, to a smaller training time and a smaller decision tree.

5. Comparing Performance with Other Systems

Although we are not aware of any systems that address the task of logic roles identification, we have mentioned systems that address the related task of shallow semantic parsing. Table 7 offers a comparative view at the precision of our model (the first model with traces resolved for all-verb experiment) and the ones in [9](G&P in the table) and [21](SVM in the table), for which we report the best overall precision on arguments 0–5 for the *gold* experiments. The gold experiments use gold standard parses.

System	Precision
Logic	92.5
SVM	89
G&P	85

Table 7. Comparing logic role identification with shallow semantic parsing.

The results reported here are considerably better, probably due to the relatively simpler task of identifying logical roles, as compared to the shallow semantic parsing problem (which includes two subtasks: argument classification and argument identification). The performance of the best mentioned systems in each individual subtask is in the upper 80s or lower 90s, thus comparable to the performance on the functional arguments identification task.

6. Impact on the Logic Form Task

In this section, we study the impact our machine learning solution on the bigger task of mapping English sentences onto logic form (LF).

Let us start by providing several metrics to quantify the quality of LF produced by some automated system. There are two metrics we are going to use: precision and recall.

Argument Level

At argument level we define Precision as the number of correctly identified arguments divided by the number of all identified arguments. Recall at argument level is the number of correctly identified arguments divided by the number of arguments that were supposed to be identified.

Predicate Level

Precision at predicate level is the number of correctly and fully identified predicates (with ALL arguments correctly identified) divided by the number of all attempted predicates. Recall at predicate level is the number of correctly and fully identified predicates (with ALL arguments correctly identified) divided by the number of all predicates that were supposed to be identified.

Let us suppose that a system generates the following logic form for the above example:

Sample Output: Earth:n_(x1) provide:v_(e1, x1, x2) food:n_(x2) we(x3) eat:v_(e2, x3, x4) every:a_(x4) day:n_(x4)
Correct Output: Earth:n_(x1) provide:v_(e1, x1, x2) food:n_(x2) we(x3) eat:v_(e2, x3, x2, x4) every:a_(x4) day:n_(x4)

where *x4* is incorrectly identified as the direct object of the *eating* event. In the correct output there are 12 slots to be filled and predicate *eat* should have 4 arguments. The previously defined measures for the sample output are given in Table 8.

Metric – Level	Argument Level	Predicate Level
Precision	10/11	6/7
Recall	10/12	6/7

Table 8.

In addition to precision and recall, a more global measure is reported, namely exact sentence, which is defined as the number of sentences whose logic form was fully identified (all predicates and arguments correctly found) divided by the number of sentences attempted.

In order to see the impact of our model on the performance of a LFI task, we collected from the web 200 sentences containing our target verbs in a coordination. Further, we have manually annotated the logical subject for them, thus obtaining a gold standard. At the same time, we generated logic forms using the approach presented in [24]. From the generated logic forms the subject information is retrieved and compared with the gold standard. We repeated this process by applying our best model to the 200 sentences and compared the results with the gold standard. An increase in precision of about 13% was observed at sentence level, 1.5% at argument level and 3.2% at predicate level. The big impact at sentence level can be explained by the fact that the method in [24] generates many sentences which have only few arguments wrongly assigned and, thus, correcting them will make the whole sentence a correct case when computing the precision. The recall for the learning method presented here is 100% due to the generality of the model, thus not suffering from the coverage problem (cases not handled by the system) in [24]. The 200 sentences were less than 40 words (to ease the task of syntactic parsing) and exhibited a diverse population in terms of subcategorization frames with regard to the target verbs in our test set.

7. Conclusions

In this paper we presented a method on how to apply decision trees to induce classifiers for the problem of functional roles detection. The method relied on several models which were built using a set of linguistic features gathered from large corpora.

The reported results for the induced decision trees are an upper bound of the performance of the models on the given set of verbs, since the tagging and parsing of sentences from which we derived the training data are accurate (tag or parse errors (noise) are present, though at a very low rate, in Treebank).

We plan to introduce tagging and parsing errors in the training data to see how the models behave in the presence of noise. Using noisy state-of-the-art parsers for that purpose will not help, as they do not provide information regarding syntactic functional arguments at logic level.

The relatively small number of verbs in the test set can constitute a drawback of our experiments. Nevertheless, the reader should keep in mind that those verbs are highly ambiguous. We plan to extend our experiments to a set of verbs that better resembles the natural distribution of argument ambiguity of English verbs. Such a distribution will have an average ambiguity level lower than that of the set of verbs considered so far and thus, we expect our results to improve.

The classifiers were induced from data extracted from a large corpus, namely Treebank, and the good performance obtained proves that gathering linguistically-motivated features from such a corpus can capture functional argument information.

References

[1] Marcus, M. and Santorini, B. and Marcinkiewicz, Building a Large Annotated Corpus of English: the Penn Treebank, *Computational Linguistic*, volume 19, number 2, pp. 313–330.

[2] Blaheta, Don and Charniak, Eugene, Assigning function tags to parsed text, *Proceedings of the 1st Meeting of NAACL*, (2000), pp. 234–240.

[3] Collin F. Baker, Charles J. Fillmore and John B. Lowe, The Berkeley FrameNet Project, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, (1998), Morgan Kaufmann Publishers, San Francisco, California, editors Christian Boitet and Pete Whitelock, pp. 86–90.

[4] Chen, J. and Rambow, O., Use of Deep Linguistic Features for the Recognition and Labeling of Semantic Arguments, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (2003), Sapporo, Japan.

[5] Michael John Collins, A New Statistical Parser Based on Bigram Lexical Dependencies, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, (1996), Morgan Kaufmann Publishers, San Francisco, editors Arivind Joshi and Martha Palmer, pp. 184–191.

[6] Michael Collins, Three generative, lexicalised models for statistical parsing, *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, (1997), San Francisco, Morgan Kaufmann.

[7] Walter Daelemans, Jakub Zavrel, Ko van der Sloot and Antal van den Bosch, *TiMBL: Tilburg Memory-Based Learner – version 5.0 Reference Guide*, ILK Technical Report 03–10.

[8] David Davidson, The logical form of action sentences, *The Logic of Decision and Action*, pp. 81–95, University of Pittsburgh Press, editor Rescher, N.

[9] Gildea, D. and Hockenmaier, J., Identifying Semantic Roles Using Combinatory Categorial Grammar, *Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing* (EMNLP), Sapporo, Japan.

[10] Hobbs, J. R., Ontological Promiscuity, *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, (1983), pp. 57–63, Cambridge, MA.

[11] Jijkoun, Valentin, Finding non-local dependencies: Beyond pattern matching, *Proceedings of the ACL-2003 Student Research Workshop*, (2003), pp. 37–43.

[12] Johnson, Mark, A simple pattern-matching algorithm for recovering empty nodes and their antecedents, *Proceedings of the 40th Meeting of ACL*, (2002), pp. 136–143.

[13] Kingsbury, P. and Palmer, M. and Marcus, M., Adding Semantic Annotation to the Penn TreeBank, *Proceedings of the Human Language Technology Conference*, San Diego, California.

[14] A. Korhonen and G. Gorrell and D. McCarthy, Statistical filtering and subcategorization frame acquisition, *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong.

[15] MARIA LAPATA, Acquiring Lexical Generalizations from Corpora: A Case Study for Diathesis Alternations, *Proceedings of the 37th Meeting of the North American Chapter of the Association*, (1999), College Park, MD, pp. 397–404.

[16] HANG LI AND NAOKI ABE, Generalizing Case Frames Using a Thesaurus and the MDL Principle, *Computational Linguistics*, volume 24, number 2, pp. 217–244.

[17] MAGERMAN, D., Statistical Decision-Tree Models for Parsing, *Proceedings of the Association of Computational Linguistics Conference*, June (1995).

[18] MARCUS, M., SANTORINI, B. AND MARCINKIEWICZ, Building a Large Annotated Coprus of English: the Penn Treebank, *Computational Linguistic*, volume 19, number 2, pp. 313–330.

[19] RADA MIHALCEA AND VIVIANA NASTASE, Diacritics Restoration: Learning from Letters versus Learning from Words, *CICLing*, pp. 339–348.

[20] MOLDOVAN, DAN I. AND RUS, VASILE, Logic Form Transformation of WordNet and its Applicability to Question Answering, *Proceedings of ACL 2001*, Toulouse, France, 6–11 July, Association for Computational Linguistics.

[21] PRADHAN, S., HACIOGLU, K., WARD, W., MARTIN, J. AND JURAFSKY, D., Semantic Role Parsing: Adding Semantic Structure to Unstructured Text, *Proceedings of the International Conference on Data Mining* (ICDM-2003), November, Melbourne, FL.

[22] RUS, VASILE, *Logic Form for WordNet Glosses and Applications*, Southern Methodist University, PhD Thesis, May (2002).

[23] RUS, VASILE, *A Semantically Enhanced Logic Form: Introducing Adjuncts*, Southern Methodist University, Unpublished Manuscript.

[24] RUS, VASILE, High Precision Logic Form Transformation, *Proceedings of the International Conference with Tools in Artificial Intellingence*, (2001), Dallas, TX, November, IEEE Computer Society, IEEE Press.

[25] JIRI STETINA, SADAO KUROHASHI AND MAKOTO NAGAO, General Word Sense Disambiguation Method Based on A Full Sentential Context, in *Use of WordNet in Natural Language Processing Systems: Proceedings of the Workshop, Association for Computational Linguistics, Somerset*, (1998), New Jersey, pp. 1–8.

[26] S. STEVENSON AND P. MERLO, Automatic verb classification using distributions of grammatical features, *Proceedings of the 9th Conference of the European Chapter of the ACL*, (1999), Bergen, Norway, pp. 45–52.

[27] SURDEANU, M., HARABAGIU, S., AARSETH, P. AND WILLIAMS, J., Using Predicate-Argument Structures for Information Extraction, *Proceedings of the Association of Computational Linguistic Meeting*, Sapporo, Japan, July (2003), pp. 7–12.

[28] BRISCOE TED AND J. CARROLL, Automatic Extraction of Subcategorization from Corpora, Automatic Extraction of Subcategorization from Corpora, *Proceedings of the 5th Conference on Applied Natural Language Processing* (ANLP-97), Washington, DC, USA.

Received: March, 2004
Revised: July, 2004
Accepted: July, 2004

Contact address:
Vasile Rus
Department of Computer Science
The University of Memphis
373 Dunn Hall
Memphis, TN 38152
USA
e-mail: `vrus@memphis.edu`

VASILE RUS is an Assistant Professor of Computer Science at The University of Memphis. He also holds an appointment in the Institute for Intelligent Systems of Fedex Institute of Technology. Dr. Rus received his Masters and PhD degrees from Southern Methodist University in Dallas, TX and was an Assistant Professor at Indiana University before moving to Memphis. His research interests are in the areas of intelligent systems, autotutoring systems, artificial intelligence, knowledge-based systems, natural language based knowledge representation. His professional interests are mainly in software systems, software configuration management and distributed systems. He has published his work in many international peer-reviewed conferences and journals and served as member or chair in as many program commitees. Currently, Dr. Rus is organizing a Special Track on "Natural Language based Knowledge Representations: New Perspectives" at FLAIRS 2005, an AAAI affiliated conference.