A Survey on Semantic Role Labeling and Dependency Parsing

by

Avishek Dan
(Roll No. 113050011)
and
Janardhan Singh
(Roll No. 10305067)

Under the guidance of
Prof. Pushpak Bhattacharyya

Abstract

Semantics is a field of Natural Language Processing concerned with extracting meaning from a sentence. *Semantic Role Labeling* takes the initial steps in extracting meaning from text by giving generic *labels* or *roles* to the words of the text. The meaning of this small set of labels can be assumed to be understood by the machine. To help semantic extraction the relationship between the words in a text needs to be understood at the syntactic level. Dependency Grammar and Parsing give binary relationships between the words of a text giving clues to their semantic relations. This document attempts to give a brief survey on these two important fields concerned with Semantic extraction from text. Semantic Role Labeling Task was surveyed till the year 2010 while concepts of Dependency Parsing were covered upto 2008.

Table of Contents

Chapter 1

Semantic Role Labeling

Semantic Role Labeling is the task of assigning semantic roles to the constituents of the sentence.

In recent years, we have seen successful deployment of domain specific semantic extraction systems. The challenge is to move from domain specific systems to domain independent and robust systems. This is now possible because of the development of large semantic databases (corpora) and progress in machine learning algorithms. Most of these databases are hand-tagged corpora developed by large groups of people. Although, many rule-based techniques like Link Parser, MiniPar and Lexical Functional Grammar have been traditionally used for this task, success has also been achieved by statistical techniques. One of the foundational works on this ground was done by Jurafsky and Gildea(2002) [DD02]. This work uses lexical resources like *WordNet* and *Framenet*. Statistical techniques are applied on these semantic databases for semantic extraction. The idea is to train supervised classifiers on these corpora that can be used to automatically tag vast amount of unseen text with shallow semantic information.

Semantic role labelers are commonly developed using a supervised learning paradigm where a classifier learns to predict role labels based on features extracted from annotated training data. The creation of resources that document the realization of semantic roles in example sentences such as FrameNet [BFL98b] and PropBank [KP02] has greatly facilitated the development of learning algorithms capable of automatically analyzing the role semantic structure of input sentences.

The section starts by describing semantic roles. It then discusses about the different lexical resources that can be used for Semantic Role Labeling. It then describes the different *Semantic Role Labeling* techniques.

1.1 Semantic Roles

In linguistic theory, *semantic roles* are one of the oldest classes of constructs. The Paninian *karaka theory* is probably one of the oldest works in this field[DD02]. A lot of variety in semantic roles exist today. The semantic roles could be domain-specific or generic. Fillmore[BFL98a] gives a hierarchical classification of semantic roles. The Framenet project was based on these Framenet roles given by Fillmore. We will look at Framenet later in the chapter. Let's look at some examples of semantic roles.

Consider an example from the *Cognition* domain in figure 3.1

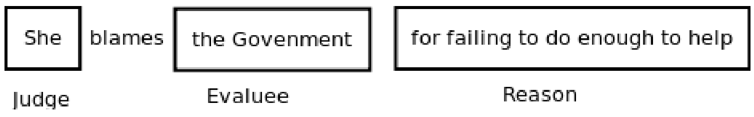


Figure 1.1: Semantic Role Labeling example

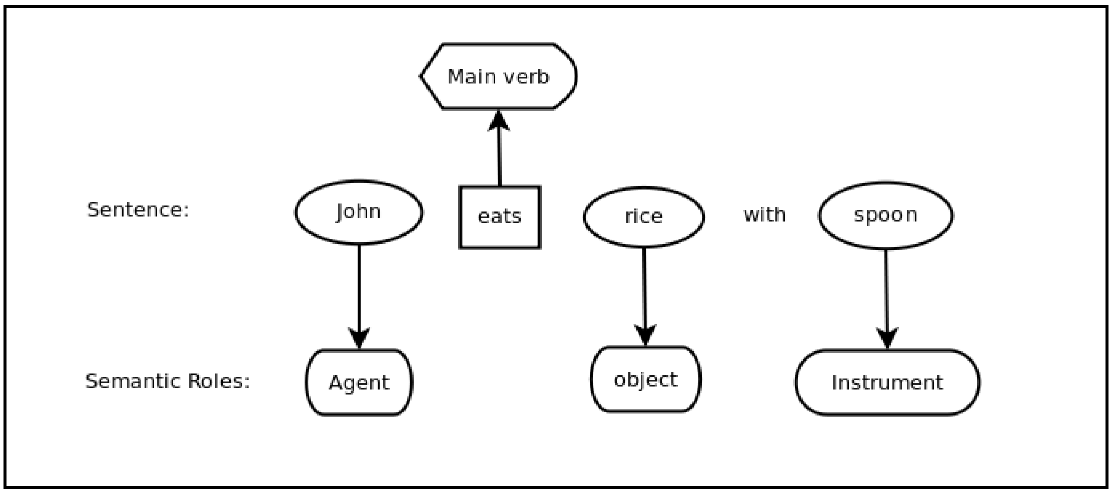


Figure 1.2: Semantic Role Labeling example

Here, the semantic roles *Judge*, *Evaluee* and *Reason* are specific to the cognition domain when a judgment is made. In the Framenet hierarchy, these roles fall in the *Judgment* frame.

For an example with more generic roles, consider the sentence in figure 1.2.

1.2 Lexical Resources

This section looks at some of the useful lexical resources used for Semantic Role Labeling.

1.2.1 Framenet

A frame is a structure used to define the semantic meaning of a word. It is a generalizable concept with recurring frame elements that are recognized intuitively. Frame elements are the elements which make up a frame. FrameNet currently has about 170,000 manually annotated sentences providing a unique training dataset for semantic role labeling [BFL98b]. In FrameNet dataset, the sentences are arranged in a hierarchical order with each frame referring to a concept. Frames at the higher level refer to a more generic concept while frames at the lower level refer to more specific concepts. Figure 1.3 [DD02] gives the structure of frames in the FrameNet.

Figure 1.3 that every frame has invoking *predicates* attached to it. These are the verbs and some nouns of English that invoke the concept, referred to, by the frame they are attached to. Sentences that have these *predicates* would have constructs that play the role given by the *frame elements* of the invoked*frame*. For example, in figure 3.1 the predicate *blame* invokes the *Judgment frame* and other constructs in the sentence play the invoked semantic roles. In that example:

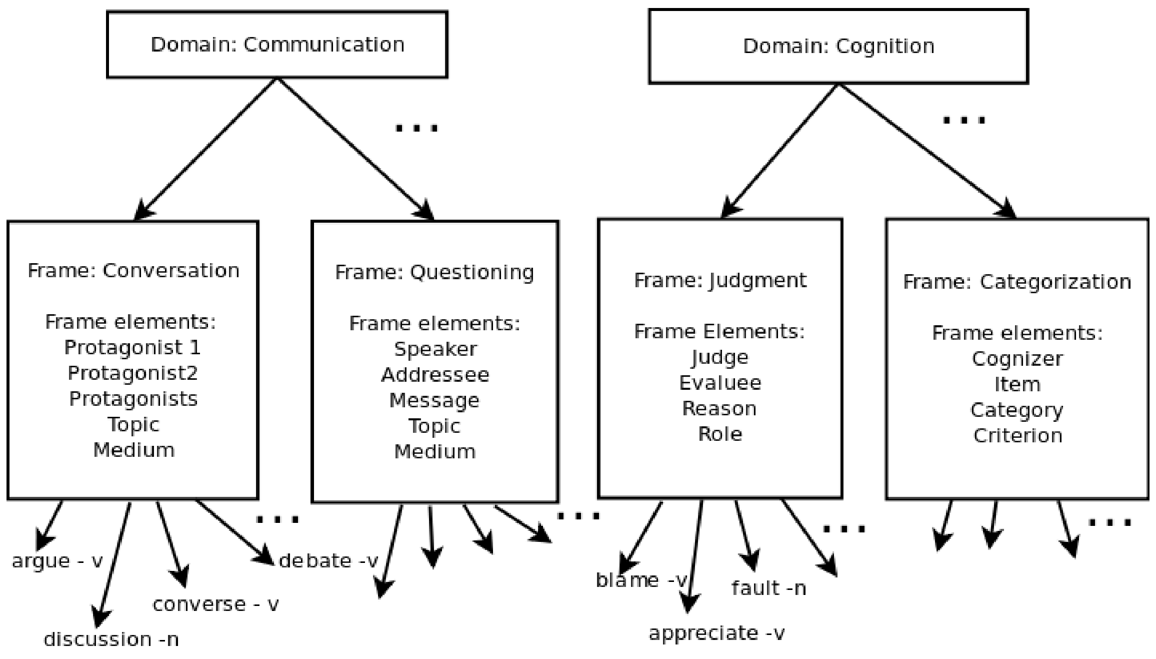


Figure 1.3: Sample domains and Frames in FrameNet

(She) plays the role *(Judge)*, *(the Government)* plays the role *(Evaluee)*, *(for failing to do enough to help)* plays the role *(Reason)*,

Figure 1.4 shows how the *Transportation* frame is inherited by the *Driving* frame which in turn is inherited by the *Riding* frame. It can be seen that the frame elements are inherited and they become more and more specific down the order. The scenes on the other hand, do not have such a relationship.

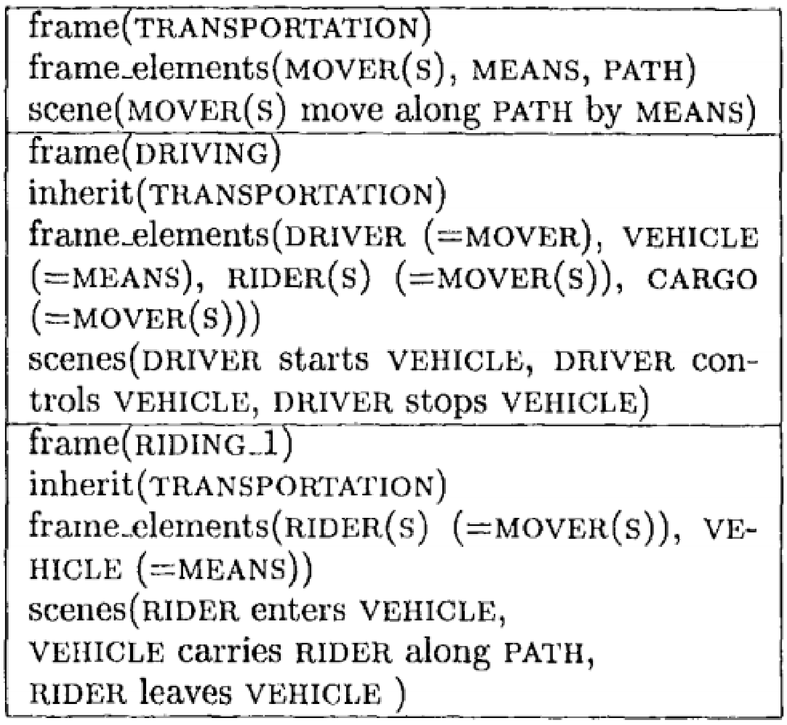
1.2.2 Propbank

Propbank[KP03] is another important lexical resource for Semantic Role Labeling. Propbank is a *proposition bank* in which sentences are annotated with *verbal propositions* and their *arguments*. It was proposed by Martha Palmer et. al. It is similar to Framenet but differs in two major ways[Dut11]:

1. All the verbs in the corpus are annotated.
2. All arguments to a verb must be syntactic constituents. A standard set of argument labels have been defined for this purpose.

Verbs are annotated with coarse grained senses and with inflectional information. Inflection describes how does the form of the verb modify with change in tense, person, aspect, mood, number and other grammatical categories [inf11]. The arguments to a verb include:

1. Core argument labels from **ARG0** to **ARG5**. Each of them have a specific meaning like that of the *karakas* in Paninian karaka theory.
2. All arguments to a verb must be syntactic constituents. A standard set of argument labels have been defined for this purpose. eg **ARGM-ADV**: General purpose modifier label.


frame(TRANSPORTATION)
frame_elements(MOVER(S), MEANS, PATH)
scene(MOVER(S) move along PATH by MEANS)

frame(DRIVING)
inherit(TRANSPORTATION)
frame_elements(DRIVER (=MOVER), VEHICLE (=MEANS), RIDER(S) (=MOVER(S)), CARGO (=MOVER(S)))
scenes(DRIVER starts VEHICLE, DRIVER controls VEHICLE, DRIVER stops VEHICLE)

frame(RIDING_1)
inherit(TRANSPORTATION)
frame_elements(RIDER(S) (=MOVER(S)), VEHICLE (=MEANS))
scenes(RIDER enters VEHICLE,
VEHICLE carries RIDER along PATH,
RIDER leaves VEHICLE)

Figure 1.4: Frame Inheritance

1.2.3 Verbnet

VerbNet[Sch05] is a hierarchical lexical resource that organizes English verbs into different classes based on the verbal classification of Levin (1993). Each verbal class takes different thematic roles and certain syntactic constraints that describe their superficial behavior. VerbNets hierarchical verb classes establish a set of possible thematic roles and a set of possible syntactic realizations. VerbNet contains mappings to other resources such as FrameNet and WordNet. Figure 1.5 shows a VerbNet class with it thematic roles and various frames.

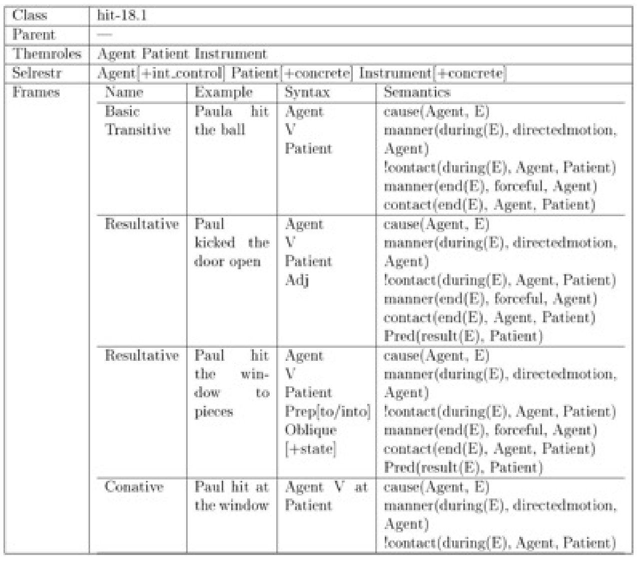
VerbNet can be highly effective in choosing the correct number of roles for a verb.

1.2.4 Wordnet

WordNet[Fel98] is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The result is a network of meaningfully related words and concepts.

1.3 Link Parser based on Link Grammar

Link parser uses Link Grammar[ST95] to give semantic roles to words in the form of relation between pair of words. The pair forms a grammatical relation. This grammatical relation defines the role of one word with respect to the other. The word whose role is defined modifies



Class	hit-18.1			
Parent	—			
Themroles	Agent Patient Instrument			
Selrestr	Agent[+int_control] Patient[+concrete] Instrument[+concrete]			
Frames	Name	Example	Syntax	Semantics
	Basic Transitive	Paula hit the ball	Agent V Patient	cause(Agent, E) manner(during(E), directedmotion, Agent) !contact(during(E), Agent, Patient) manner(end(E), forceful, Agent) contact(end(E), Agent, Patient)
	Resultative	Paul kicked the door open	Agent V Patient Adj	cause(Agent, E) manner(during(E), directedmotion, Agent) !contact(during(E), Agent, Patient) manner(end(E), forceful, Agent) contact(end(E), Agent, Patient) Pred(result(E), Patient)
	Resultative	Paul hit the window to pieces	Agent V Patient Prep[to/into] Oblique [+state]	cause(Agent, E) manner(during(E), directedmotion, Agent) !contact(during(E), Agent, Patient) manner(end(E), forceful, Agent) contact(end(E), Agent, Patient) Pred(result(E), Patient)
	Conative	Paul hit at the window	Agent V at Patient	cause(Agent, E) manner(during(E), directedmotion, Agent) !contact(during(E), Agent, Patient)

Figure 1.5: VerbNet hit class

the other word, the governing word.

1.4 Link Grammar

Link grammar contains set of words which act as its terminal symbols, a set of relations which define the links between a pair of words and a set of linking requirement which are the properties of the words. The linking requirements of words are stored in a dictionary.

1.4.1 Links and Linking Requirements

Link is the connection or relation between two words. Linking requirements of a word define the roles the word can play. It also defines about the word it can be linked with. Example: *The word ”cat”*

- can be a Subject (S)
- can be an Object (O)
- will have a Determiner (D)

1.4.2 Connectors and Formulae

The symbols S, O and D represent connectors of the words. The connectors define what the word can be and what the word needs. Connectors end with ”+” or ”-” which indicates the direction of its connecting word. ”+” means the word it will be connected to is in the right of it and ”-” means the word is to the left of it. The connectors at the opposite ends must match (+ -). A *linking requirements* is represented by a *formula* of connectors combined by

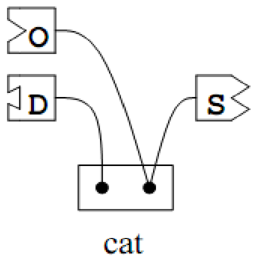


Figure 1.6: The roles ”cat” can play

binary operators ”&” and ”or”. Let C and D be two connectors of a word. Then C & D would mean the linking requirement of the word is that both connectors C and D must be connected to their complement connectors. C or D means only one of C and D needs to get connected to a corresponding complementary connector.

1.4.3 Disjunctive Form

When formula of each word is represented in disjunctive form it has a set of disjuncts associated with it. A disjunct has two ordered lists of connectors -

1. Left list - Connectors with ”-” mark with it.
2. Right list - Connectors with ”+” mark with it.

A disjunct looks like ((L1,L2,L3,L4,,Ln)(R1,R2,R3,R4,,Rm)). To convert a formula into its disjunctive form we need to find all the valid permutations of connectors. By ”valid” it means that connectors separated by “&“ must be present in the list and among connectors separated by “or“ only one should be present in the list. Some examples are shown in table1.1

1.4.4 Grammar Rules

Link Grammar contains rules which put constraints on the formation of relations among the words of the given sentence. These rules are:

1. **Planarity** - Links between the words, when all of the links are drawn above the words, will not cross.
2. **Connectivity** - All the words should have a connection.
3. **Satisfaction** - The linking requirement of each word is satisfied by the provided links.
   (a) **Ordering**: When the left connectors of a formula are traversed from left to right, the words to which they connect proceed from near to far. When the right connectors of a formula are traversed from left to right, the words to which they connect proceed from far to near.
   (b) **Exclusion**: No two links may connect the same pair of words.

Word	Formulae	Disjunctive Form
Sit	S- & (O+ or B+)	((S)(O)), ((S)(B))
A	D+	(()(D))
Dog	{@A-} & (D-) & {B+} & (S+ or O-)	((D) (S)), ((D,O) ()), ((D) (S,B)), ((D,O) (B)), ((A,D) (S)), ((A,D,O) ()), ((A,D) (S,B)), ((A,D,O) (B))
Black	A+	(()(A))

Table 1.1: Disjunctive Form

1.5 MiniPar

Minipar is a principle-based English parser. It represents its grammar as a network where the nodes represent grammatical categories and the links represent types of syntactic (dependency) relationships. Minipar uses grammatical principle instead of grammatical rules which act as constraints associated with the nodes and links.[Roy11]

1.5.1 Principle-based Parser

Principle-based grammars[Lin94] use principles like government-binding (GB) theory. Let us consider a sentence in passive voice: The ice-cream was eaten. If a rule-based parser handles this passive voice while finding the object of the action ”eat” it would use an IF-THEN rule:

```
If  (Subject be-Verb + ed No Object)
The     make the Subject the Object
```

This is a shallow approach. The basic idea of principle-based parser is to replace this shallow approach with a much deeper and explanatory set of principles. This set of principles is classified into Generators and Filters.

1.5.2 Generating Principles

Generating principles produce possible structures of a given sentence. This class includes:

- **X-bar Theory**: This theory describes the basic shapes of tree allowed in the language. In natural language there are roughly two forms of tree: function-argument form, like a verb begins in a verb phrase and argument-function form where X ends in a XP.
- **Movement Principle**: Movement principle says that any phrase can be moved anywhere. This would create new possibilities, though this might violate other principles. Example: *John likes ice-cream.* This can changed after moving to: Ice-cream, John likes.
- **Binding theory**: Binding theory specifies how pronouns can be bound to their antecedents. Multiple mappings of one pronoun to different antecedents would create new possible structures.
  Example: *John thinks he likes ice-cream.*
  ”he” may refer to John or some other person.(two possibilities).
  He thinks John likes ice-cream.
  ”He” refers surely some other person except John.(one possibility)

1.5.3 Filtering Principles

Filters remove possible structures which fail some given constraints.

- **Case Filter**: Case theory specifies that every noun phrase should be assigned a case. The subject is given a nominative case and direct objects are given accusative case. If a phrase tree fails to satisfy that the tree is an invalid one.
  Example: *It is likely that John will win.*
  John: nominative case. This is a valid sentence structure.
  It is likely John to win.
  No case for John. This is an invalid sentence structure.

- **Theta Criterion**: Theta theory determines the number of arguments a verb needs and assigns thematic roles to its arguments. This brings in the concepts of transitive and intransitive verbs. Transitive verbs should specify the agent and patient of the action. If any verb requires two objects it must specify the thematic roles of both. One such example is the verb give which demands what to give and whom to give.
  Example: *John put the book on the shelf.*
  ”put” has two objects book, thing to put, and shelf, place to put. This is valid.
  John put the book.
  ”put” has only objects book, thing to put ,not the place to put. This is invalid.

- **Empty Category Principle and Locality Theory**:Empty category principle says that all traces must be properly governed. The trace should not be too far away from its antecedent.
  Example: *Who do you think likes Mary?*
  Who_i *do you think* $[t_i likesMary]$*?*
  Here the trace t_i is governed by its antecedent who_i. This is valid.
  Who_i *do you think that* $[t_i likesMary]$*?*
  Here the trace t_i far away from its antecedent who_i. This is invalid.

Given the generator-filter model, the simplest way build a parser is to cascade the principles in a sequence.

1.6 Automatic Semantic Role Labeling

[DD02] This is a statistical technique of semantic role labeling, in which, a statistical classifier is trained over a corpora of sentences for the Semantic Role Labeling(SRL) task. Two sets of experiments were described by Jurafsky and Gildea, which were conducted by them on the Framenet corpus. In the first set, inputs to the system were a sentence, a target word, a frame and the frame element boundaries labeled by hand. The outputs were the frame element labels. In the second set of experiments, inputs to the system included just a sentence, a target word and a frame. The system now performed the dual task of *frame element boundary identification* and *frame element labeling*. Again the outputs were the frame element labels.

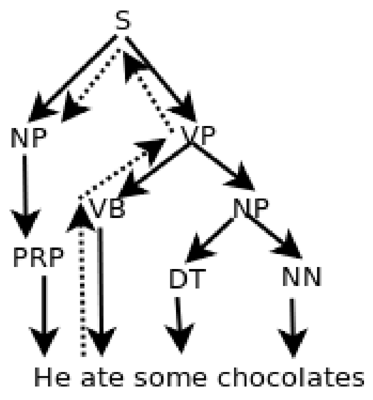


Figure 1.7: Parse tree path example.

1.6.1 Features for frame element labeling

The sentence is first parsed by a constituent parser to obtain a parse tree. Then following features are derived:

Phrase type For every constituent of the sentence its phrase type can be determined by the constituent parse.

Governing Category A Noun Phrase(NP) can be directly a constituent of a Sentence(S) or a Verb Phrase(VP). This is used as a feature for only NPs giving a strong indication if it is used as subject or object of the verb.

Parse Tree Path A parse tree path of a constituent of a sentence is the path of the constituent from the target word in the constituent parse tree which includes the intermediate nodes and the arrow directions. Figure 1.7 shows an example path between verb *eat* and noun phrase *He* as: $VB \uparrow VP \uparrow S \uparrow NP$

Position The position feature indicates whether the constituent occurs before or after the predicate invoking the frame.

Voice The voice of the sentence is a feature. Active or passive voice is identified with the help of 10 passive identifying patterns.

Head Word The head words of each constituent acts as a very useful feature.

1.6.2 Features for frame element boundary identification

Similar features are also used for frame element boundary identification namely head word, parse tree path and target word.

1.6.3 Probability estimation of a single role

In order to automatically label the semantic role of a constituent, we wish to estimate a probability distribution indicating how likely the constituent is to fill each possible role, given the features described above and the predicate, or target word, t:

$$P(r|h, pt, gov, position, voice, t)$$

This could be done by direct counts as:

$$P(r|h, pt, gov, position, voice, t) = \frac{count(r,h,pt,gov,position,voice,t)}{count(h,pt,gov,position,voice,t)}$$

But there were on an average 34 sentences per target word in the Framenet dataset used. In general, the data is sparse for estimating the conditional probability of a label given all the above features together. This is because, a particular combination of the above six features along with the target word occurs rarely in the dataset. To overcome this, conditional probabilities of the frame element labels given subsets of above features like $p(role \mid targetword)$, $p(role|path, targetword)$ *etc.* are computed. These are combined using different strategies like equal linear interpolation, EM linear interpolation, Geometric Mean, Back-off linear interpolation and Back-off geometric mean. This strategy gives a significant improvement in performance over the baseline approach of directly estimating the conditional probability of the labels given all the six features in the conditioning set.

1.6.4 Probability estimation of all the roles in the sentence

If we assume that the roles played by the different constituents of a sentence are independent of each other then the probability estimation of the previous section is enough to label the roles. But, it is trivial to note that this is just a simplifying assumption. If we relax this assumption, then we have to compute role assignment over the entire sentence r^*:

$$r^* = argmax_{r_{1...n}} P(r_{1...n}|t, f_{1...n})$$

Here, $f_{1...n}$ are the set of features as discussed above. Applying Bayes rule, removing elements not contributing to the argmax computation and assuming that features f_i are independent of each other given the target word t, we get the following equation:

$$r^* = argmax_{\{r_{1...n}\}} P(r_{1...n}|t) \prod_i P(f_i|r_i, t)$$

Applying Bayes rule again and removing constant part of numerator (not contributing to argmax) we get:

$$r^* = argmax_{\{r_{1...n}\}} P(r_{1...n}|t) \prod_i \frac{P(r_i|f_i,t)}{P(r_i|t)}$$

The frame element identification part can be incorporated in the above evaluation as follows:

$$r^* = argmax_{\{r_{1...n}\}} P(r_{1...n}|t) \prod_i \frac{P(r_i|f_i,fe_i,t).P(fe_i|f_i))}{P(r_i|t)}$$

1.6.5 Generalizing lexical semantics

The head word feature is observed to be the most useful. But due to large vocabulary of English, training on all possible head words is infeasible. Hence, to generalize our training from a small set of head words to other head words three techniques viz. automatic clustering, bootstrapping and making head word hierarchy using WordNet is described.

1.7 Extensions to Automatic SRL

[PWH$^+$04][PWH$^+$05] extend on this basic work. In the [PWH$^+$04], one vs rest Support Vector Machines(SVM) are trained for Automatic Semantic Role Labeling. The features used extend on the basic features combining features like:

- **Named Entities in Constituents** Named entities (PERSON, ORGANIZATION, LOCATION, PERCENT, MONEY, TIME, DATE) using Identi-Finder (Bikel et al,1999) added as 7 binary features
- **Head Word POS** Part of Speech tag of the headword
- **Verb Clustering** Predicate clustering to counter unknown predicates
- **Partial path** To avoid data sparsity parse tree paths in partial forms
- **Verb Sense Information** Word sense of the predicate
- **Head word of prepositional phrases** Replacing the preposition with the first noun as the head word
- **First and Last word/POS in constituent** Found to be discriminative
- **Ordinal constituent position** To avoid false positives of elements far away from predicate
- **Constituent Tree Distance** Finer way of depicting an existing feature
- **Constituent relative features** Features of parents and siblings
- **Temporal cue words** Temporal words not captured by NER
- **Dynamic class content** Hypotheses of at most two previous nodes belonging to the same tree

Some of these features were given by [SHWA03]

[PWH$^+$05] extend their own work by combining parses obtained from semantic parsers trained using different syntactic views like Charniak parser and Mini-par dependency parser.

[HY10] model word spans using an 80 state HMM model. Taking the hidden states as features in addition to the previous features an improvement is achieved in open domain SRL.

1.7.1 Other work

[ARR09] present an unsupervised argument identification algorithm for SRL. Using an unsupervised parser which generates unlabeled parse tree and POS tag annotation the algorithm is able to achieve 56% precision on the argument identification task.

1.8 Semi-supervised Semantic Role Labeling

Supervised learning methods deliver reasonably good performance. However, the reliance on labeled training data, which is both difficult and highly expensive to produce, presents a major obstacle to the widespread application of semantic role labeling across different languages and text genres. Even for English, despite the substantial annotation effort involved in the creation of FrameNet, the numbers of annotated instances vary greatly across lexical items. Also labeled data is scarce for individual predicates.

A better alternative is to use semi-supervised methods that make use of a small number of manually labeled training instances to annotate a large number of unlabeled instances which are similar to the training instances. Whereas manually labeled data are expensive to create, unlabeled data are often readily available in large quantities. The latter approach aims to improve the performance of a supervised Semantic Role Labeling system by enlarging its training set with automatically inferred annotations of unlabeled sentences. The key idea of this approach is to find novel instances for classifier training based on their similarity to manually labeled seed instances. The underlying assumption is that sentences that are similar in their lexical material and syntactic structure are likely to share a frame semantic analysis. The annotation of an unlabeled sentence can therefore be inferred from a sufficiently similar labeled sentence.

For example, Seed sentence:

$[Lee]_{Agent} punched [John]_{Victim}$[in the eye]$_{\text{Body part}}$.

Unlabeled sentences:

Bill will punch me in the face. I punched her hard in the head.

The unlabeled sentences are both structurally and semantically similar to the seed sentence. Now, in order to use these new sentences as training data we must somehow infer their semantic roles. We can probably guess that constituents in the same syntactic position must have the same semantic role, especially if they refer to the same concept (*e.g.*, body parts) and thus label in the face and in the head with the role Body part. Analogously, Bill and I would be labeled as Agent and me and her as Victim.

$[Bill]_{Agent}$will punch$[me]_{Victim}$[in the face]$_{\text{Body part}}$.

$[I]_{Agent} punched [her]_{Victim} hard$[in the head]$_{\text{Body part}}$.

1.8.1 Learning Method

This method needs a small seed corpus that has been manually annotated but not on a scale that is sufficient for high-performance supervised learning. For each sentence in the seed corpus, a number of similar sentences are selected from an unlabeled expansion corpus. These are automatically annotated by projecting relevant semantic role information from the labeled sentence. The similarity between two sentences is compared by measuring whether their arguments have a similar structure and whether they express related meanings (concepts). The seed corpus is then enlarged with the k most similar unlabeled sentences to form the expanded corpus.

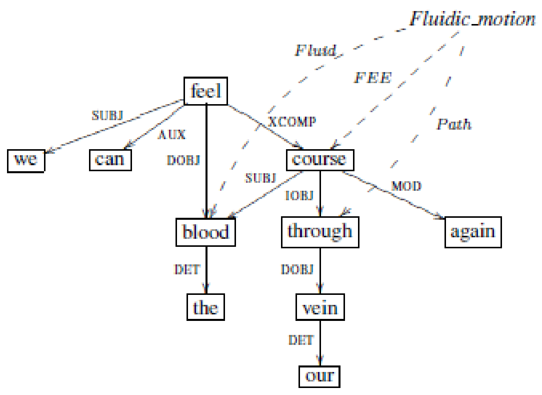


Figure 1.8: Labeled dependency graph

Lemma	GramRole	SemRole
blood	SUBJ	Fluid
vein	IOBJ_THROUGH	Path
again	MOD	—

Table 1.2: Predicate-argument structure for the verb course in Figure 1.8

Extracting Predicate-Argument Structures

The method operates over labeled dependency graphs. Figure 1.8 shows the labeled dependency graph for the sentence *We can feel the blood coursing through our veins again*. The frame is *Fluidic motion* and the roles are *fluid, path* and the verb, which is called the *frame evoking element (FEE)*.

Directed edges (without dashes) represent dependency relations between words, edge labels denote types of grammatical relations (*e.g.* SUBJ, AUX). The verbs, *i.e.* frame evoking elements, in the seed and unlabeled corpora are represented by their predicate-argument structure. Specifically, the direct dependents of the predicate course (*e.g.* blood or again in Figure 1.8) and their grammatical roles (*e.g.* SUBJ, MOD) are recorded. Prepositional nodes are collapsed, *i.e.*, the prepositions object and a composite grammatical role are recorded (like IOBJ_THROUGH, where IOBJ stands for prepositional object and THROUGH for the preposition itself). For each argument node, the semantic roles it carries are recorded, if any. All surface word forms are lemmatized.

An example of the argument structure information we obtain for the predicate course 1.8 is shown in Table 1.2.

Measuring Similarity

For each frame evoking verb in the seed corpus, a labeled predicate-argument representation similar to table 1.2 is created. All sentences from the unlabeled corpus containing the same verb is then extracted. Since a verb may invoke different frames with different roles and predicate-argument structure, all the extracted sentences are not suitable instances for adding to the training data. Therefore only sentences resembling the seed annotations must be selected.

The idea used for selection is that verbs appearing in similar syntactic and semantic contexts will behave similarly in the way they relate to their arguments. Estimating the similarity between two predicate-argument structures amounts to finding the highest-scoring alignment between them, *i.e.*, given a labeled predicate-argument structure p^l with m arguments and an unlabeled predicate-argument structure p^u with n arguments, find and score all possible alignments between these arguments. An alignment is an injective function

$$\sigma : M_\sigma \rightarrow \{1, ..., n\}$$

where

$$M_\sigma \subset \{1, ..., m\}$$

Thus we choose a subset of arguments from the labeled predicate-argument structure and map each of the arguments to some argument of the unlabeled predicate-argument structure on a one-one basis. It can be seen that the alignment function allows for partial alignments, *i.e.* there can be partial alignment on both sides.

Each alignment σ is scored using a similarity function sim(σ) defined as:

$$\sum_{i \varepsilon M_\sigma} \left(A.syn\left(g_i^l, g_{\sigma(i)}^u\right) + sem\left(w_i^l, w_{\sigma(i)}^u\right) - B \right)$$

where $syn(g_i^l, g_{\sigma(i)}^u)$ denotes the syntactic similarity between grammatical roles g_i^l and $g_{\sigma(i)}^u$ and

$sem((w_i^l, w_{\sigma(i)}^u)$ denotes the similarity between the head words w_i^l and $w_{\sigma(i)}^u$.

The goal is then to find an alignment such that the similarity function is maximized. This optimization problem is a generalized version of the linear assignment problem. The best alignment crucially depends on estimating the syntactic and semantic similarity between the arguments. $syn(g_i^l, g_{\sigma(i)}^u)$ is set to 1 if the relations are identical, set to some $a <= 1$ if the relations are of same type but different subtype and to 0 otherwise. The semantic similarity is measured with a semantic space model. The meaning of each word is represented by a vector of its co-occurrences with neighboring words. The cosine of the angle of the vectors representing w^l and w^u quantifies their similarity. The parameter A counterbalances the importance of syntactic and semantic information, while the parameter B acts as a threshold for the lowest acceptable similarity value for an alignment between two arguments is possible.

Figure 1.9 graphically illustrates the alignment projection problem. Here the aim is to project the semantic role information from the seed *blood coursing through our veins again* onto the unlabeled sentence Adrenalin was still coursing through her veins. The predicate course has three arguments in the labeled sentence and four in the unlabeled sentence. There are 73 possible alignments in this example. Each alignment is scored by taking the sum of the similarity scores of the individual alignment pairs (*e.g.* between *blood* and *be*, *vein* and *still* and so on). In this example, the highest scoring alignment is between *blood* and *adrenalin*, *vein* and *vein*, and *again* and *still*, whereas *be* is left unaligned. Note that only *vein* and *blood* carry semantic roles which are projected onto *adrenalin* and *vein*, respectively.

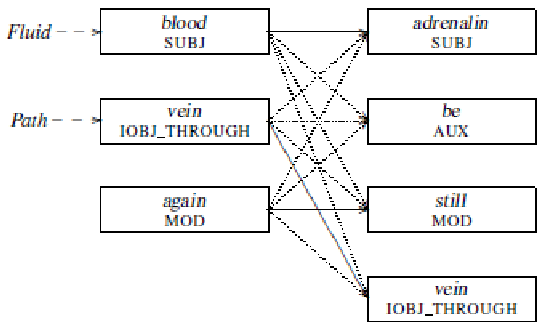


Figure 1.9: Labeled dependency graph

1.8.2 Projecting Annotations

Once the best alignment between seed sentence and unlabeled sentence is obtained, the roles are projected, resulting in a labeled sentence. These projections form the expansion corpus. For each seed sentence, *k* most similar neighbors are added to the training data. The parameter k controls the trade-off between annotation confidence and expansion size.

1.9 Statistical Method for UNL Relation Label Generation

Nguyen *et al.* [NI06] presents a statistical technique for relation generation. To the besy of my knowledge, this is the only attempt at UNL generation using statistcial techniques. This method only tackles the simpler problem of generating relation labels given the source and destination phrases. The system extracts various features representing the source and destination phrases and the relationships between them. The training step finds the conditional probability of each relation given the feature values. The system is described in detail in the following sections.

1.9.1 Feature Generation

The features used are similar to the ones described in Gildea *et al.* [DD02]. These are enumerated below.

- Phrase type: POS tag of the source and destination phrase. The Charniak parser tags is used.
- Head word: This is the root of the phrase in the syntactic parse.
- Voice: This is only applicable to verbs. For other features, the value is unspecified.
- Dependency Path: This is the path from the source phrase to the destination phrase in the dependency parse tree.
- Syntactic Cross Path: This is the path from the source phrase to the destination phrase in the syntactic parse tree.

1.9.2 Training

The number of relations for each feature vector is counted in the training data to estimate the conditional probability of each relation given the feature vector in the test data. Since data is sparse, the probabilities are also calculated for individual features.

1.9.3 Testing

The probabilities are combined through linear interpolation to get the final probability estimation. The predicted relation is an argmax over the probabilitiy values. The system achieves a best accuracy of 73.2% when tested on data supplied by the UNDL foundation.

1.10 Overview of UNL System at GETA

This section describes the UNL system developed at GETA, France by Etienne Blanc [Bla08][Bla11].UNL is an artificial language based on semantic graphs. It stems from the pivot language of one of the ATLAS II, a Japanese-English machine translation (MT) system. UNL is intended for much broader applications than just machine translation. UNL was created and is maintained by the former leader of the ATLAS team, Hiroshi Uchida. The link between UNL and various natural languages are the responsibilities of the 'language centers'. GETA is the language center for French. UNL has huge potential for dissipating information on the internet. The more difficult task of converting from the source language to UNL (enconversion) needs to be done once, irrespective of the number of target languages. The comparatively easier task of deconversion from the well structured non ambiguous UNL graph to natural language (deconversion) has to be done for each target language.

The UNL system was developed at GETA as part of the Ariane machine translation system. The system incorporates a French Enconverter and Deconverter. In addition, a UNL graph editor is provided. Sixteen languages such as English, Chinese, Arabic, Russian, Hindi are linked to French through Universal word dictionaries.

1.10.1 Universal Word Resources

The system uses three Universal Word resources

- The UNL Knowledge Base which constitutes a network structure where UWs are interconnected through relations of UNL.
- The UW++ dictionary developed at GETA
- The UW dictionary developed at CFILT, IIT Bombay having entries for Indian languages.

1.10.2 The Enconversion and Deconversion process

The process consists of three major steps:

- Analysis: The analysis steps elaborates a so called 'multilevel' tree. This tree reflects the syntagmatic structure of the input sentence, but bears 3 levels of information : the morphologic, syntactic and semantic ones.

- Transfer: The transfer step is mainly lexical : It translates the source words (UWs) into the target French words (lemmas). The UNL-French dictionary is used for this step. The morphologic and syntactic informations are no more relevant in the target language..

- Generation: The generation step builds up the multilevel target tree, and finally the target text.

1.10.3 Graph to Tree Conversion

Ariane translation system uses a tree structure for UNL. When translating from UNL graphs produced by other systems that uses a graph structure, the deconversion first process converts the graph into a tree using various rewrite rules. Some of the conversion rules are outlined below:

- Nodes with multiple parents: In this case, the relation of the child with one of the parents is reversed. For example, obj(parent, child) becomes invobj(child, parent)

- Closed circuit: Circuits do not exist in an UNL graph if directionality is considered. But if in the corresponding undirected graph, a closed circuit exists, a node is duplicated to create an undirected graph.

- Compound UWs: Compound UWs are replaced in the main graph with the corresponding subgraphs.

1.10.4 Deployment

The system is available as a mobile phone application and also through a web based interface. Interactive modes of lexical transfer and execution with trace is possible. In the interactive mode for lexical transfer, the possible French equivalents for a UW is displayed and the user has the option to select from the option. If no equivalent is found, the user may enter one, and the dictionary is automatically augmented.

The author also speaks about the possibilities of applying UNL in speech MT in the future.

1.11 Summary

Semantic Role Labeling is a fast developing area of research. It is a shallow level representation of semantics and has applications in large number of *Natural Language Processing* tasks. These include the Semantic Web, Information Extraction, Textual Entailment *etc.* With advances in both rule based and statistical techniques *Semantic Role Labeling* is a rapidly developing area in NLP.

Chapter 2

Dependency Grammar and Dependency Parsing

Dependency grammar is a theory that defines how the words in a sentence are connected to each other. The basic idea is that in a sentence all but one word is dependent on some other word. The word which is independent is usually the main verb of the sentence. Consider the following example :

Sentence: *A man sleeps.*

The main verb in the sentence is *sleeps* which is independent of all the other words and gives the central idea of the sentence. If the sentence is about a *sleep*, then there should be an *agent* or a *subject* of this action. Thus the word *man* is the *agent* or the *subject* who is sleeping and it depends on the word *sleeps*. We can also say that *man* fills the *verb-argument frame* for the verb *sleeps*. Now, in the sentence we are not talking about some specific *man* but about some indefinite *man*. This is denoted by the article *a*. Thus the word *a* is dependent on the word *man*, or in other words, *modifies* the word *man*. Thus, the dependency relations coming out of this simple relation is given in 2.1.

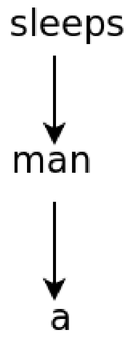


Figure 2.1: Dependency Relation

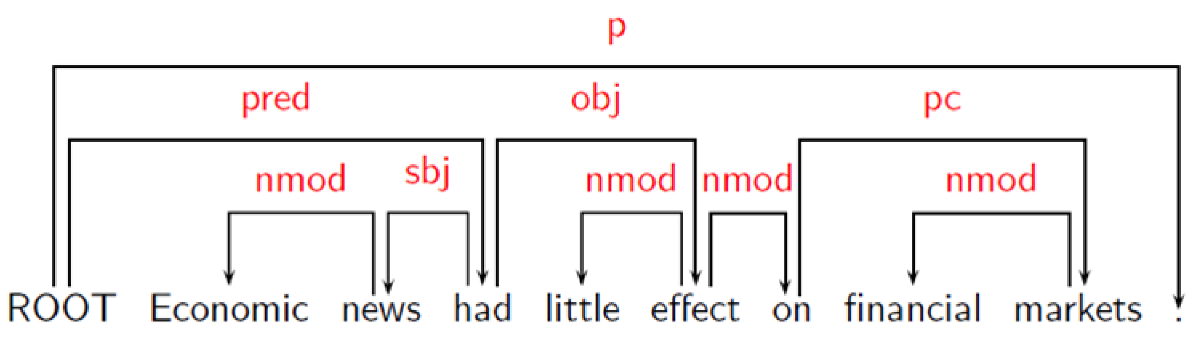


Figure 2.2: Projective example

2.1 Robinson's axioms

Robinson(1970) [Rob70] described four basic axioms for dependency grammar which govern the well-formedness of dependency structure. It says that in a dependency structure:

1. One and only one element is independent.
2. All others depend directly on some element.
3. No element directly depends on more than one element.
4. If A depends directly on B and some element C intervenes between them (in the linear order of the string), then C depends directly on A or B or some other intervening element.

The first axiom has the result that the dependency structure has a *root* node. The second axiom allows the dependency structure to be a single connected component. The third axiom is also called the *single-head* property says that every element can have at most one parent. Thus, the dependency structure can be a tree or a graph and different formalisms exist for both of them. The fourth axiom is the *projective* property, but many formalisms do not obey this axiom.

2.2 Projective and Non-projective dependency structures

A dependency structure is said to be *projective*, if it follows the fourth axiom of Robinson. A projective structure essentially means that no two edges of the graph cut-across each other. Let us look at an example 2.2.

As seen in the figure, no edges of the graph here cut-across each other. Algorithm designing for projective dependency structures is easier. But the problem with projective formalisms is, that languages with relatively free word order do not follow projectivity constraint. An example of a non-projective dependency structure is given in figure 2.3.

As can be seen, this sentence has a projective version, as seen in figure 2.4 if we modify the word order.

These three examples have been taken adapted from [Niv08].

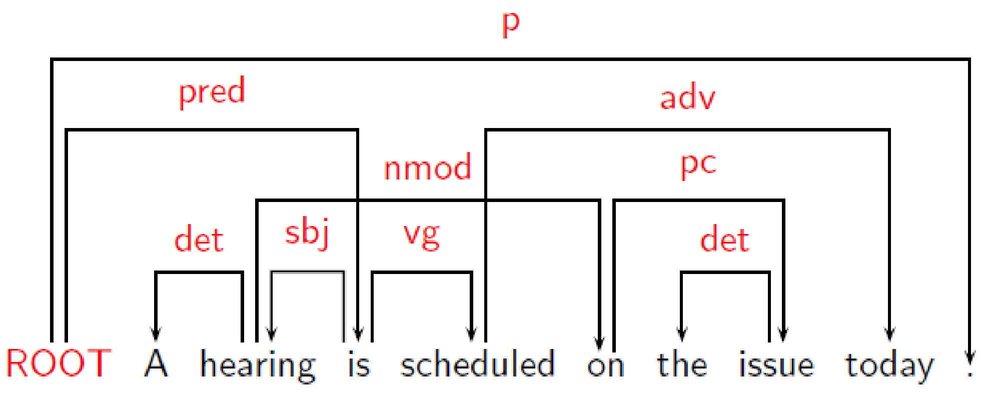


Figure 2.3: Non-projective example

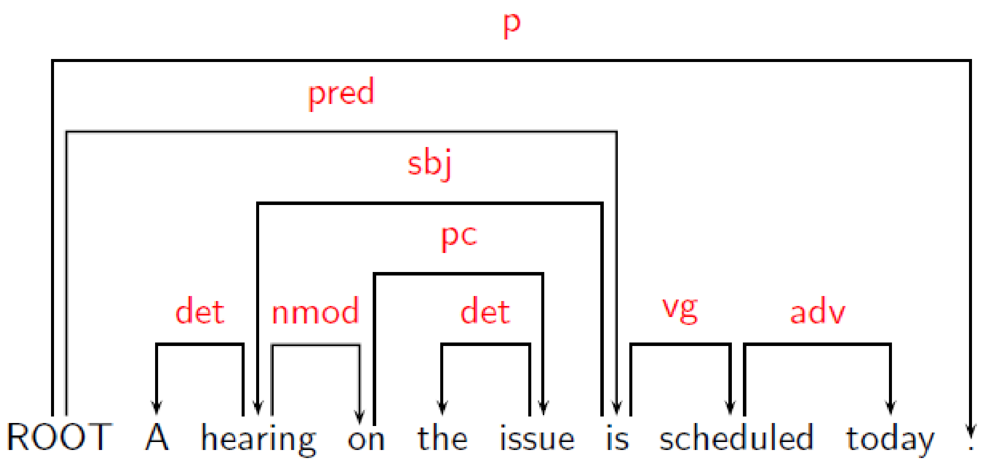


Figure 2.4: Projective version of the Non-projective example

2.3 Dependency Parsing Techniques

Dependency Parsing is the method of parsing sentences into *Dependency Structures*. In this section, we look at some of the prevalent *Dependency Parsing* techniques.

2.3.1 Data-based Dependency Parser

Data-driven dependency parsing uses machine learning from linguistic data to parse new sentences. In this report the supervised approaches will be discussed. The sentences used for machine learning are annotated with their correct dependency structures. The goal is to learn a good predictor of the dependency tree of a sentence given an input sentence. A model for this is M where M = (T, P, h), where T is the given set of constraints that helps in forming the structures for the sentence, p is a set of parameters to be learned from data and h is a fixed parsing algorithm.

2.3.2 Transition-based dependency parsing

Transition-based parsing system parameterizes a model to learn to predict the next transition given the input sentence and the parse history. The dependency trees are predicted using a greedy, deterministic algorithm.

A transition system generally contains a set of states, a set of rules to define transition of one state to another, an initial state and a set of final states. A simple stack-based transition system which uses a form of shift-reduce parsing will be explained. A configuration would be defined as a triple of stack, input buffer and a set of dependency arcs. A formal definition is:
Given a set of dependency types R, an input sentence S=$w_0 w_1 ... w_n$, the configuration c of the sentence S is defined as c = (σ, β, A), where

- σ is a stack of words w_i
- β is the input buffer
- A is a set of dependency arcs $(w_i, r, w_j) \in V_S R V_S$.

A configuration c contains partially processed words in the stack, remaining words to be processed in the buffer and the partially constructed dependency tree in the form of dependency arcs in the set A. σ and β are represented in the form of lists. The stack σ has its top at the right.
Initial configuration: $c_0(S) = ([w_0]_\sigma, [w_1, w_2, , w_n]_\beta, \phi)$.
Terminal configuration: $c_m(S) = (\sigma, []_\beta, A)$ for any σ and A.
Make w_0=ROOT and push it to stack and we reach configuration $c_0(S)$.

2.4 Summary

Dependency Grammar is one of the earliest language grammars which has regained it's importance with applications in Information Extraction and various Natural Language Processing tasks. A wide variety of dependency formalisms exist but most of them follow the basic axioms of Robinson. Dependency parsers like the Stanford Dependency Parser and the XLE parser are now available and the accuracy of these parsers is improving rapidly with the use of both statistical and rule-based techniques.

Chapter 3

Techniques for Corpus Based Learning

Manual encoding of linguistic information have often been challenged by annotated corpus based learning methods as a technique for providing linguistic knowledge to a system. Many rule based systems suffer from low recall rates due to linguistic knowledge acquisition bottleneck. Automatic linguistic information extraction from text corpus can overcome this and thus help create robust and highly accurate natural language processing systems. This chapter investigates various statistical and rule based techniques for learning from annotated corpus. With the availability of large quantities of corpus, corpus learning techniques have become a viable option. Some successful application of corpus-based techniques are in building parts-of-speech taggers, empirically assigning probabilities to grammar rules, computing statistical measures of lexical association, word sense disambiguation and machine translation.

3.1 Transformation-Based Error-Driven Learning

Although corpus-based approaches can successfully model various linguistic information, the linguistic insight into the model behavior is often lost in huge quantities of statistical data. For example, a hidden Markov model based parts of speech tagger tags words based on a model uses a huge database of P(tag |previous tags) and P(word |tag) as the prediction model which lack any linguistic intuition. This kind of indirect modeling can make it difficult to analyze, understand and improve the ability of these approaches to model underlying linguistic behavior. The point to be noted here is that unlike rule-based systems, corpus-based methods succeed without capturing the true complexities of knowledge. They achieve this due to the fact that complex linguistic phenomena can often be indirectly observed through simple epiphenomena.

Consider the following sentences for example.

- The frog kept an eye on the fly.
- The pilot will not fly the aircraft.

The word fly can be POS-tagged in the above sentences without linguistic insight into phrase structure by observing that a word that lies one or two words to the right of a modal verb is a verb whereas a word following a determiner is a noun.

As demonstrated above, simple stochastic n-gram taggers can obtain very high accuracy simply by observing the phenomenon in small neighborhoods without recourse to the underlying linguistic structure. The work by Brill [Bri95] identifies the problems associated with such

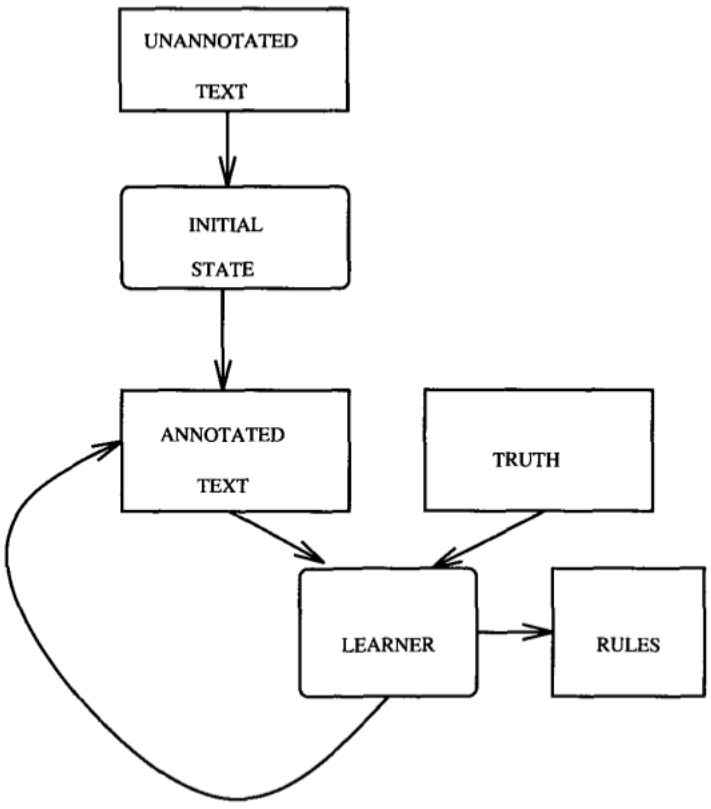


Figure 3.1: Transformation-Based Error-Driven Learning

blind dependence on corpus statistics. The understanding of the model generated by corpus based methods is central to detecting when the approximation model deviates from the actual linguistic phenomenon.

The author proposes a new approach called transformation-based error-driven learning. In this approach, the linguistic information that is learned is represented in a concise and comprehensive form. Thus it can be used to discover how learnt models can be coupled with the underlying linguistic phenomenon. The method has been applied to problems such as part-of-speech tagging, prepositional phrase attachment disambiguation, syntactic parsing, letter-to-sound generation and speech recognition.

3.1.1 Description

Figure 3.1 illustrates the working of transformation-based error-driven learning. The unannotated text is first fed to an initial-state annotator. The initial-state annotator can range from a naive system that randomly annotates the text to a manually designed annotation framework. For example, the initial-state annotator for a parts-of-speech (POS) tagger may assign the same or random POS tag to all words or it may assign the most common tag for each word. Similarly for syntactic parsing, the initial-state annotations may range from the output of a sophisticated parser to random tree structure with random nonterminal labels.

The annotated text is then compared to the truth. A manually annotated corpus serves as reference for truth. The goal is to learn transformations that takes the annotation closer to the

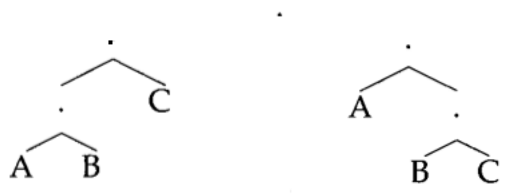


Figure 3.2: Bracketing Rewrite Rule

truth. The transformations are written in the form of rewrite rules. In addition, the transformation are also ordered in the sequence in which they may be applied. A transformation essentially consists of two components

- A rewrite rule, which specifies the effect of applying the transformation
- A triggering environment, which specifies the conditions under which the transformation may be applied

For example, in the POS tagging example taken earlier, the following transformation may be applied

- Rewrite rule: Change the tag from verb to noun
- Triggering environment: The previous word is a determiner

In case of parsing, the rewrite rule may be in the form of bracketing. Figure 3.2 illustrates such a rule. Here A, B and C are either terminals or nonterminals. Such a transformation may be applied to the initial-state parse tree ((John eats) rice) to obtain (John (eats rice)).

The above technique is essentially a greedy algorithm where at each step we choose the transformation that reduces the error by the greatest extent. For this purpose, a set of possible transformations is predefined. The algorithm terminates when no transformation can reduce the error further. The author also suggests simulated annealing and look-ahead window based learning as alternative methods. Figure 3.3 shows the operation of the technique. The space of transformations in this case is T1 to T4. The transformation T2 achieves the largest decrease in error so it is chosen as the first transformation and then T3 is chosen. The algorithm terminates after this since no further reduction in error can be achieved.

This algorithm can be applied to many different problems by specifying the following.

- The initial state annotator
- The set of transformations
- The objective function to be optimized

The order of application of transformations is important in many cases. It is also important to consider whether while applying a transformation, all triggering environments should be identified first and then the rewrite rules are applied or the application can be done one by one.

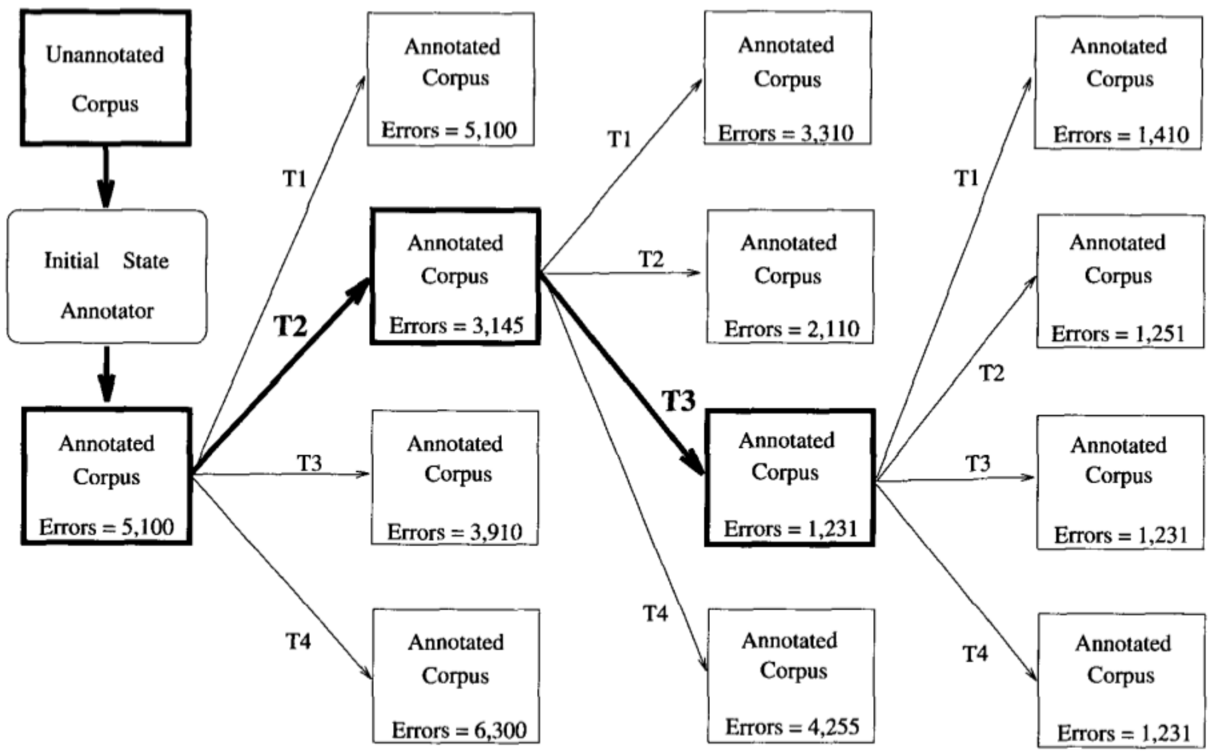


Figure 3.3: Example of Terminaton-Based Error-Driven Learning

3.1.2 Observations

The work also compares this learning method with decision trees and shows that decision tree classifications are a subset of the valid transformations using this method. However, the method is free from the data sparsity and over-fitting problems associated with decision trees since each transformation is applied on the entire training data set.

In addition, unlike decision trees, the effect of transformations at each step are reflected on the data before the next transformation is applied. Thus the later transformations can make a more informed decision.

3.2 Statistical Dependency Analysis

The task of finding word-word dependencies is closely related to the task of generating relations between words. The work by [YM99] uses a bottom up technique for generating word-word dependencies. Features are extracted from the parsed annotated training data and are used to learn SVM classifiers for dependency generation.

3.2.1 Parsing Actions

A parsing iteration is performed on the sentence from left to right. At any time, a parsing action is applicable to two neighboring words also called target nodes. The following parsing actions are applicable to the input sentences.

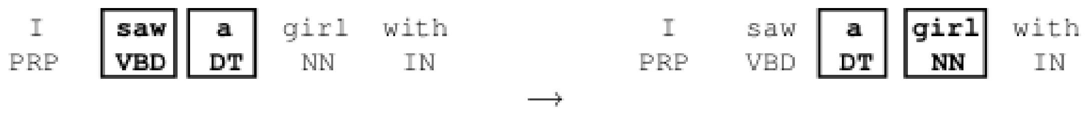


Figure 3.4: Example of Shift action showing states before and after the action

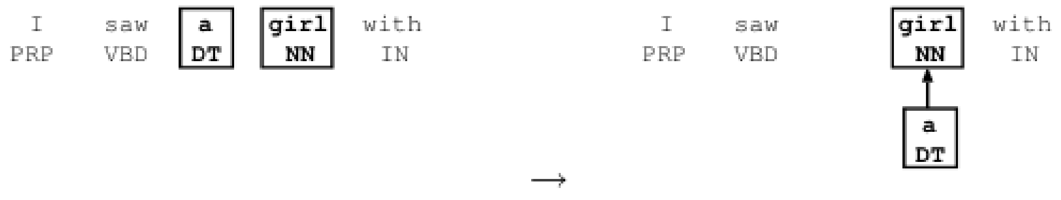


Figure 3.5: Example of Right action showing states before and after the action

- Shift: No dependency can be constructed between the nodes. The point of focus shifts right by one unit.
- Right: A dependency is constructed between the nodes with the left node becoming the child of the right node. The right frontier focus point is unchanged.]item Left: A dependency is constructed between the nodes with the right node becoming the child of the left node. The left frontier focus point is unchanged.

Figures 3.4 and 3.5 show the results of shift and right actions applied to target nodes. The actions right and left may be applied only when all dependencies of the node that becomes the child has been resolved. Once a node becomes a child, it is removed from further consideration.

3.2.2 Algorithm

The work describes an iterative algorithm for constructing the parse tree based on the above parser actions. The algorithm iteratively scans the sentence from left to right, choosing parsing actions from the contextual information around a node and the applying the selected parser action. After each action, the target nodes shift right by one unit. The algorithm terminates when only the root node is left or no more dependencies can be constructed. The choice of parser action is taken by training Support Vector Machines. This is discussed in the next section.

3.2.3 Classification

Support Vector Machines (SVMs) are trained on the features extracted. The choice of SVMs is made due to the following advantages over other classifiers.

- High generalization performance in high dimensional feature space: Since SVMs optimize based on maximum margin strategy, it theoretically guarantees low generalization error.

- Learning with combination of features using polynomial kernel functions: Kernel functions allow SVMs to deal with non-linear classification.

Since the choice of parser action is a multi class problem, three binary classifiers are constructed taking the actions pairwise: Left vs. Right, Shift vs. Left and Right vs. Shift. Majority voting among the three classifiers gives the chosen action.

The left context is defined as the indexes on the left of the target nodes and the right context is defined as the indexes on the right of the target nodes. The features used are of the form (p, k, v) where p denotes the position with respect to the target nodes (negative values denote left context, positive values denote right context), k denotes the type of the feature and v denotes its value. The types of features used for this task are discussed below.

- pos: Part of speech tag string
- lex: Word string
- ch-L-pos: POS tag string of the child node modifying to the parent node from left side
- ch-L-lex: Word string of the child node node modifying to the parent node from left side
- ch-R-pos: POS tag string of the child node modifying to the parent node from right side
- ch-R-lex: Word string of the child node modifying to the parent node from right side

For example, in figure 3.6, some of the features extracted are

- (-1, pos, NNS): The word to the left of "of" has a POS tag NNS
- (-1, lex, sellers): The word to the left of "of" is sellers
- (-1, ch-R-lex, the): The word modifying word to the left of "of" is the

The authors note that the task of training the SVM is quadratic to cubic in the number of training examples. Thus to reduce the training cost, the data set is divided into groups on the basis of the POS tag of the left target node. Thus a SVM is constructed for each POS tag of the left node and the appropriate SVM is chosen during dependency generation.

A kernel function is used to allow non linear combination of features. The kernel function chosen for this task is

$$(x^{'}.x^{''}+1)^2$$

.

3.3 Summary

The chapter presented two methods for corpus based learning. The first work provides a general architecture for developing a system which learns better rules through error analysis and feedback. The other work describes features that can be used for finding dependencies between words. These works provide strong motivation for an automated hybrid system.

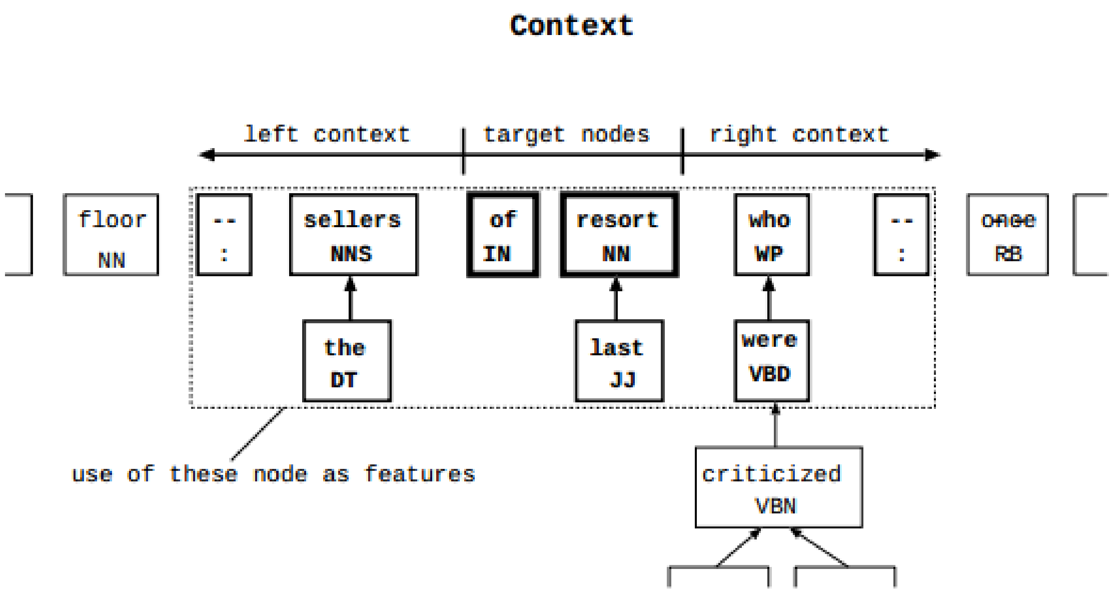


Figure 3.6: Example of Contextual Information

Bibliography

[ARR09] O. Abend, R. Reichart, and A. Rappoport. Unsupervised argument identification for semantic role labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 28–36. Association for Computational Linguistics, 2009.

[BFL98a] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pages 86–90, Montreal, Canada, 1998.

[BFL98b] Collin F Baker, Charles J Fillmore, and John B Lowe. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[Bla08] Etienne Blanc. Le langage UNL au GETA, November 2008.

[Bla11] Etienne Blanc. The Lexical Database of the French UNL Development Environment, February 2011.

[Bri95] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565, 1995.

[DD02] Gildea Daniel and Jurafsky Daniel. Automatic Labeling of Semantic Roles. In *Computational Linguistics*, 2002.

[Dut11] Subhajit Dutta. Semantics Extraction from Text, Stage 1 report. Master's thesis, IIT Bombay, 2011.

[Fel98] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[HY10] F. Huang and A. Yates. Open-domain semantic role labeling by modeling word spans. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 968–978. Association for Computational Linguistics, 2010.

[inf11] Inflection. `http://en.wikipedia.org/wiki/Inflection`, October 2011.

[KP02] Paul Kingsbury and Martha Palmer. From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1989–1993. Citeseer, 2002.

[KP03] P. Kingsbury and M. Palmer. Propbank: the next level of treebank. In *Proceedings of Treebanks and lexical Theories*, 2003.

[Lin94] D. Lin. Principar: an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 482–488. Association for Computational Linguistics, 1994.

[NI06] Dat PT Nguyen and Mitsuru Ishizuka. A statistical approach for universal networking language-based relation extraction. In *Proc. Int. Conf. on Research, Innovation and Vision for the Future*, pages 153–160. Citeseer, 2006.

[Niv08] J. Nivre. Sorting out dependency parsing. *Advances in Natural Language Processing*, pages 16–27, 2008.

[PWH$^+$04] S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL*, pages 233–240, 2004.

[PWH$^+$05] S. Pradhan, W. Ward, K. Hacioglu, J.H. Martin, and D. Jurafsky. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 581–588. Association for Computational Linguistics, 2005.

[Rob70] J.J. Robinson. Dependency structures and transformational rules. *Language*, pages 259–285, 1970.

[Roy11] Gourab Roy. Semantics Extraction from Text, Stage 2 report. Master's thesis, IIT Bombay, 2011.

[Sch05] Karin Kipper Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005.

[SHWA03] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 8–15. Association for Computational Linguistics, 2003.

[ST95] D.D.K. Sleator and D. Temperley. Parsing english with a link grammar. *Arxiv preprint cmp-lg/9508004*, 1995.

[YM99] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. *Machine Learning*, 34(1-3):151–175, 1999.