Predicate-Argument Structure

Predicate-Argument Structure (PAS) refers to the syntactic and semantic relationship between a predicate (typically a verb or verbal expression) and its arguments (typically noun phrases or other syntactic elements) within a sentence.

This structure helps in understanding how elements in a sentence are connected and how meaning is conveyed through these connections.

Key Concepts in Predicate-Argument Structure

- **Predicate:** The predicate is the central element that expresses an action, state, or relation in a sentence. It typically includes a verb and any accompanying adverbs or particles.
    - Examples:
    - "John ate pizza."
    - "She runs quickly."
- **Arguments**: Arguments are the participants or elements that are associated with the predicate and provide additional information about what the predicate relates to.
    - Examples: In "John ate pizza," "John" is the subject argument (Agent) and "pizza" is the object argument (Theme).
- **Syntactic Roles**: Syntactic roles describe the grammatical function of arguments relative to the predicate.
    - Examples:
    - Subject (Agent): The entity performing the action ("John" in "John ate pizza.")
    - Object (Theme): The entity affected by the action ("pizza" in "John ate pizza.")

- **Semantic Roles**: Semantic roles describe the underlying meaning or relationship of arguments with respect to the predicate.
    - Examples:
    - Agent: The doer or initiator of the action.
    - Theme: The entity that undergoes the action or state.
    - Experiencer: The entity experiencing a psychological state.

Importance of Predicate-Argument Structure:

- **Semantic Clarity**: Understanding PAS helps in clarifying the roles and relationships between elements in a sentence, leading to a clearer understanding of meaning.
- **Computational Linguistics**: In natural language processing (NLP), PAS is crucial for tasks such as parsing, information extraction, and semantic analysis.
- **Language Teaching and Learning**: PAS aids in teaching and learning grammar and syntax by explaining how different parts of a sentence function together.

Recourses: FrameNet

- **Semantic Frame Identification**:
    - FrameNet categorizes words and phrases into semantic frames, each representing a conceptual scenario or situation (e.g., "Eating," "Communication," "Motion").
    - These frames help NLP systems identify and disambiguate the intended meaning of predicates (verbs) in text.
- **Lexical Units (LUs):**
    - Each frame in FrameNet contains lexical units (LUs), which are words or phrases associated with that frame.

    - LUs provide a structured way to understand how predicates (verbs) are used in different contexts and what arguments (noun phrases) they take.

- **Frame Elements (FEs):**
    - FrameNet annotates frame elements (FEs) for each frame, representing the roles or arguments associated with the frame.
    - FEs correspond to syntactic and semantic roles within sentences, helping NLP systems analyze and extract information about predicate-argument relationships.
- **Semantic Role Labeling (SRL):**
    - FrameNet data can be leveraged for Semantic Role Labeling (SRL), a task in NLP that involves identifying and classifying the semantic roles of words or phrases in a sentence.
    - By using FrameNet's annotated frames and FEs, NLP models can accurately identify which words serve as predicates and which words are their arguments, thereby understanding the PAS in text.

Applications in NLP Tasks:

- **Information Extraction**: FrameNet aids in extracting structured information from text by identifying predicates and their associated arguments.
- **Text Understanding**: NLP systems can use FrameNet to improve text understanding capabilities by modeling how predicates and arguments interact to convey meaning.
- **Machine Translation**: FrameNet helps in disambiguating translation choices by providing semantic frames and roles that align with the intended meaning of predicates across languages.

Example Use Case:

- Consider the sentence: "The chef cooked a delicious meal."
- Frame: Cooking
- Lexical Unit (LU): cook
- Frame Elements (FEs):
  - Agent (The chef)
  - Instrument (None specified)
  - Patient (a delicious meal)
- In this example, FrameNet would provide structured information about the "Cooking" frame, including the roles played by "cook" (the verb) and its associated arguments ("the chef" and "a delicious meal").

Resources: PropBank

- PropBank (Proposition Bank) is another important resource in Natural Language Processing (NLP) that focuses on annotating predicate-argument structures in text.
- PropBank is a corpus-based resource that annotates verbs with their arguments and adjuncts, providing a detailed representation of how verbs are used in various contexts.
- It aims to capture the semantic roles (arguments) associated with verbs across different syntactic structures.
- **Key Features of PropBank**
  - **Annotation Scheme**
    - PropBank annotates verbs with specific semantic roles, known as "framesets," which include arguments and adjuncts associated with each verb.

    - Each frameset represents a distinct meaning or usage pattern for a verb, specifying the roles that arguments play in relation to the verb.

  - **Frame Elements (FEs):**

    - PropBank identifies and labels frame elements (FEs) for each frameset, representing the roles or semantic arguments associated with the verb.

    - FEs include roles such as Agent, Theme, Goal, Source, etc., depending on the verb and its context.

  - **Corpus Annotations:**

    - PropBank annotations are typically applied to large corpora of text, where verbs are manually annotated with their framesets and associated arguments.

    - These annotated corpora serve as valuable resources for training and evaluating NLP systems, particularly for tasks like Semantic Role Labeling (SRL).

- **Applications in NLP:**

  - **Semantic Role Labeling (SRL):** PropBank is used in SRL tasks to identify and label the semantic roles of verbs and their arguments in sentences.

  - **Information Extraction**: PropBank annotations aid in extracting structured information from text by identifying predicate-argument relationships.

  - **Machine Learning Models**: NLP models leverage PropBank data to improve performance in tasks like parsing, sentiment analysis, and machine translation.

- **Integration with Other Resources**:

  - PropBank complements other linguistic resources like FrameNet and WordNet by focusing specifically on verbs and their argument structures.
  - It provides a more fine-grained analysis of how verbs are used in context compared to more general semantic resources.

- **Benefits of PropBank in NLP:**
  - **Precision in Argument Identification**: PropBank annotations provide detailed information about the roles and relationships of verbs and their arguments, enhancing accuracy in NLP tasks.
  - **Domain Adaptability**: PropBank can be adapted to different domains and genres of text, allowing for specialized language processing applications.
  - S**tandardizatio**n: It offers a standardized framework for annotating and analyzing verb semantics, facilitating reproducibility and consistency in linguistic research and NLP applications.

Semantic Role Labeling (SRL) algorithm

- Semantic Role Labeling (SRL) algorithms in the context of Predicate-Argument Structure (PAS) aim to identify and classify the semantic roles (arguments) of predicates (verbs) within sentences.
- These algorithms play a crucial role in Natural Language Processing (NLP) tasks such as information extraction, question answering, and machine translation.

Semantic Role Labeling Algorithm for PAS:

- Input:
  - Sentence: A tokenized sentence.
  - Predicates: Pre-identified predicates (verbs).
- Dependency Parsing:

    - Perform dependency parsing to analyze the syntactic relationships between words in the sentence.
    - Extract dependency arcs (edges) that connect words and indicate syntactic dependencies (subject, object, etc.).
- Argument Identification:
    - For each predicate identified in the sentence:
    - Identify potential arguments (noun phrases, prepositional phrases, etc.) based on syntactic dependencies.
    - Use heuristics or machine learning models to determine candidate arguments that are likely to be associated with the predicate.
- Semantic Role Classification:
    - Classify each identified argument into semantic roles (Agent, Theme, Goal, etc.) based on its relationship with the predicate and contextual cues.
    - Apply linguistic patterns and features (syntactic position, dependency labels, etc.) to determine the appropriate semantic role for each argument.
- Output:
    - Generate structured representations (framesets) for each predicate, indicating its semantic roles and associated arguments.

Example Steps:

Consider the sentence: "John ate an apple."

Step 1: Dependency Parsing

- Dependency structure:
- ate (V) → John (NP)
- ate (V) → apple (NP)

Step 2: Argument Identification

- Identified arguments: ate (V): John (Agent), apple (Theme)

Step 3: Semantic Role Classification

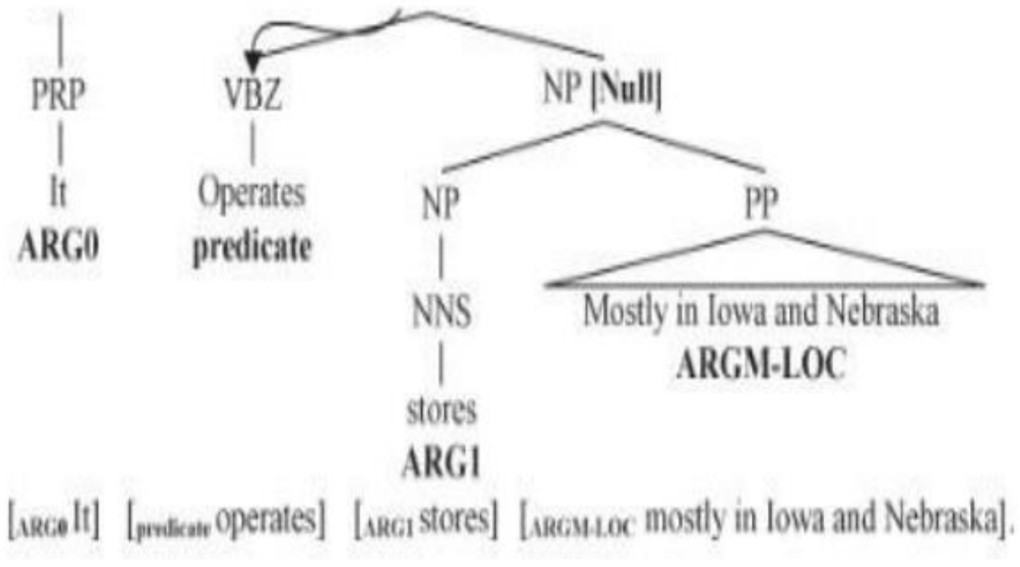
- Assign semantic roles:
- ate (V)
- Agent: John
- Theme: apple

Step 4: Output

- Representation for "ate"
    - Predicate: ate
    - Agent: John
    - Theme: apple

Parse Structure Grammar

- In the context of Parse Structure Grammar (PSG) applied specifically to Predicate-Argument Structure (PAS) in Natural Language Processing (NLP), features play a crucial role in capturing the syntactic and semantic relationships between predicates (verbs) and their associated arguments (noun phrases, prepositional phrases, etc.).
- These features enrich the representation of PAS and support various linguistic analyses.
- FrameNet marks word spans in sentences to represent arguments whereas PropBank tags nodes in a treebank tree with arguments.
- Since the phrase structure representation is amenable to tagging.
- Gildea and Jurafsky introduced the following features:
    - **Path**: This feature is the syntactic path through the parse tree from the parse constituent to the predicate being classified.



PRP
VBZ
NP [Null]
It
Operates
NP
PP
ARG0
predicate
NNS
Mostly in Iowa and Nebraska
ARGM-LOC
stores
ARG1
[ARG0 It] [predicate operates] [ARG1 stores] [ARGM-LOC mostly in Iowa and Nebraska].

- **Predicate:** The identity of the predicate lemma is used as a feature.
- **Phrase Type**: This feature is the syntactic category (NP, PP, S, etc.) of the constituent to be labeled.
- **Position:** This feature is a binary feature identifying whether the phrase is before or after the predicate.
- **Voice:** This feature indicates whether the predicate is realized as an active or passive construction. A set of hand written expressions on the syntax tree are used to identify the passive-voiced predicates.
- **Head Word:** This feature is the syntactic head of the phrase. It is calculated using a head word table.
- **Subcategorization:** This feature is the phrase structure rule expanding the predicate's parent node in the parse tree.

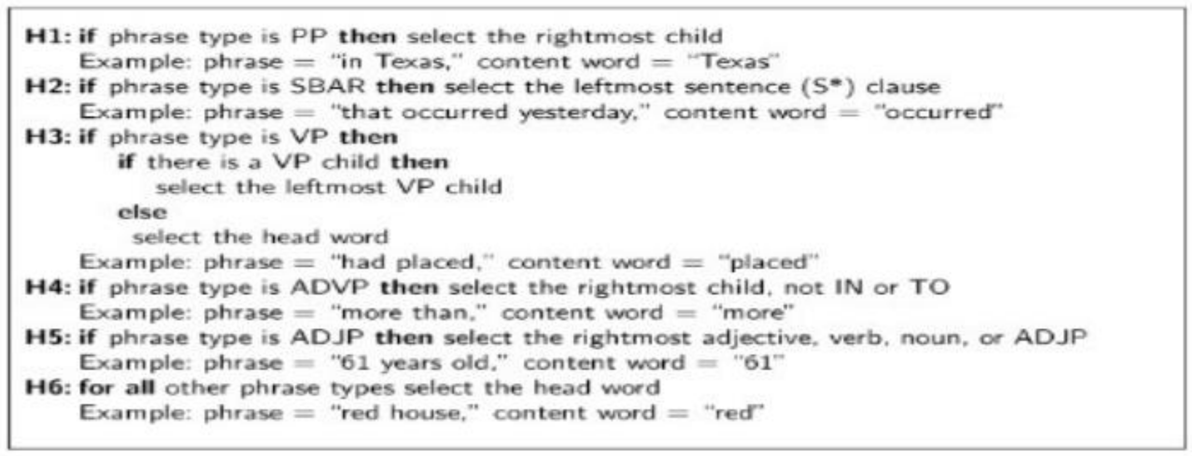
Verb Clustering:

- This predicate is one of the most salient features in predicting the argument class.
- Gildea and Jurafsky used a distance function for clustering that is based on the intuition that verbs with similar semantics will tend to have similar direct objects.

- For example: Verbs such as eat, devour and savor will occur with direct objects describing food.
- The clustering algorithm uses a database of verb-direct object relations.
- The verbs were clustered into 64 classes using the probabilistic occurrence model.

Surdeanu suggested the following features

Content Word: Since in some cases head words are not very informative a different set of rules were used to identify a so-called content word instead of using the head-word finding rules. The rules that they used are

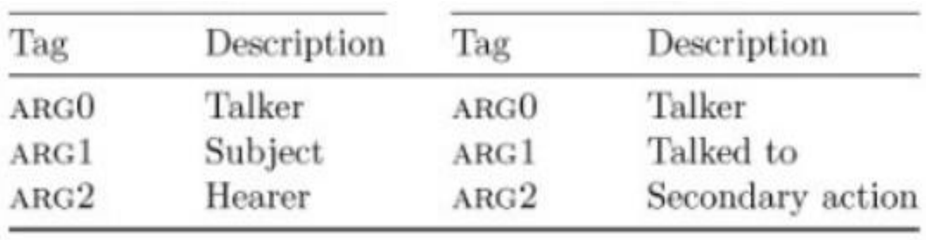


```
H1: if phrase type is PP then select the rightmost child
    Example: phrase = "in Texas," content word = "Texas"
H2: if phrase type is SBAR then select the leftmost sentence (S*) clause
    Example: phrase = "that occurred yesterday," content word = "occurred"
H3: if phrase type is VP then
        if there is a VP child then
            select the leftmost VP child
        else
          select the head word
    Example: phrase = "had placed," content word = "placed"
H4: if phrase type is ADVP then select the rightmost child, not IN or TO
    Example: phrase = "more than," content word = "more"
H5: if phrase type is ADJP then select the rightmost adjective, verb, noun, or ADJP
    Example: phrase = "61 years old," content word = "61"
H6: for all other phrase types select the head word
    Example: phrase = "red house," content word = "red"
```

List of content word heuristics

- POS of Head Word and Content Word: Adding the POS of the head word and the content word of a constituent as a feature to help generalize in the task of argument identification and gives a performance boost to their decision tree-based systems.
- amed Entity of the Content Word: Certain roles such as ARGMTMP, ARGM-LOC tent to contain time or place named entities. This information was added as a set of binary valued features.
- Boolean Named Entity Flags: Named entity information was also added as a feature. They created indicator functions for each of the seven named entity types: PERSON, PLACE, TIME, DATE, MONEY, PERCENT, ORGANIZATION.

- Phrasal Verb Collocations: This feature comprises frequency statistics related to the verb and the immediately following preposition.
- Fleischman, Kwon, and Hovy added the following features to their system:
    - Logical function: This is a feature that takes three values external argument, object argument, and other argument and is computed using some heuristic on the syntax tree.
    - Order of Frame Elements: This feature represents the position of a frame element relative to other frame elements in a sentence.
        - Syntactic Pattern: This feature is also generated using heuristics on the phrase type and the logical function of the constituent.
        - Previous Role: This is a set of features indicating the nth previous role that had been observed/assigned by the system for the current predicate.
        - Pradhan suggested using the following additional features:
        - Named Entities in Constituents: Named entities such as location and time are important for the adjunctive arguments ARGM-LOC and ARGM-TMP.
        - Entity tags are also helpful in cases where head words are not common.
        - Each of these features is true if its representative type of named entity is contained in the constituent.
    - **Verb Sense Information**: The arguments that a predicate can take depend on the sense of the predicate.
    - Each predicate tagged in the PropBank corpus is assigned a separate set of arguments depending on the sense in which it is used.
    - This is also known as the ***frameset ID***.
    - The table below illustrates the argument sets for a word.
    - Depending on the sense of the predicate “talk” either ARG-1 or ARG-2 can identify the hearer.



Tag	Description	Tag	Description
ARG0	Talker	ARG0	Talker
ARG1	Subject	ARG1	Talked to
ARG2	Hearer	ARG2	Secondary action

- **Features in Parse Structure Grammar (PSG)**

Syntactic Features:

1. **Dependency Relations:**
   - **Definition:** Specify the syntactic relationships between predicates and their arguments.
   - **Example:** In the sentence "John eats an apple":
     - eats (VERB) → John (NP) (Subject)
     - eats (VERB) → apple (NP) (Direct Object)
2. **Phrase Structure Rules:**
   - **Definition:** Define how words and phrases combine syntactically to form sentences.
   - **Example:**
     - S → NP VP: A sentence consists of a noun phrase followed by a verb phrase.

Semantic Features:

3. **Semantic Roles:**
   - **Definition:** Assign specific semantic roles (Agent, Theme, Goal, etc.) to arguments based on their relationship with the predicate.
   - **Example:**
     - Sentence: "John gives Mary a book."
     - Agent: John (the giver)
     - Recipient: Mary (the receiver)

        - Theme: book (the object being given)

4. **Frame Elements (FEs):**
    - **Definition:** Specify the semantic roles associated with specific predicates.
    - **Example:**
        - Predicate: "give"
        - Giver: John
        - Recipient: Mary
        - Theme: book

Lexical Features:

5. **Part-of-Speech (POS) Tags:**
    - **Definition:** Categorize words into lexical categories (Noun, Verb, Adjective, etc.).
    - **Example:**
        - John (NOUN), eats (VERB), apple (NOUN)
6. **Word-level Features:**
    - **Definition:** Describe properties of individual words, such as tense, number, gender, etc.
    - **Example:**
        - eats (VERB, Present tense), apple (NOUN, Singular)

Contextual and Discourse Features:

7. **Anaphora Resolution:**
    - **Definition:** Resolve references to previously mentioned entities within the discourse.
    - **Example:**
        - Resolving "he" in the sentence "John saw Peter. He waved."
8. **Discourse Relations:**
    - **Definition:** Capture relationships between sentences or utterances.
    - **Example:**

        - Establishing temporal sequence or cause-effect relationships between sentences.

Example Sentence Analysis:

Consider the sentence: "John eats an apple."

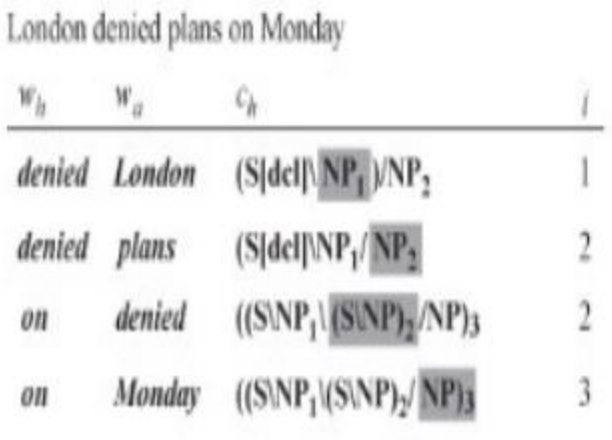
- **Syntactic Features:**
    - Dependency relations:
        - eats (VERB) → John (NP) (Subject)
        - eats (VERB) → apple (NP) (Direct Object)
    - Phrase structure rule: S → NP VP
- **Semantic Features:**
    - Semantic roles:
        - Agent: John
        - Theme: apple
    - Frame elements:
        - Agent: John
        - Theme: apple
- **Lexical Features:**
    - Part-of-Speech tags: John (NOUN), eats (VERB), apple (NOUN)
    - Word-level features: eats (VERB, Present tense), apple (NOUN, Singular)

Advantages of PSG Features in PAS:

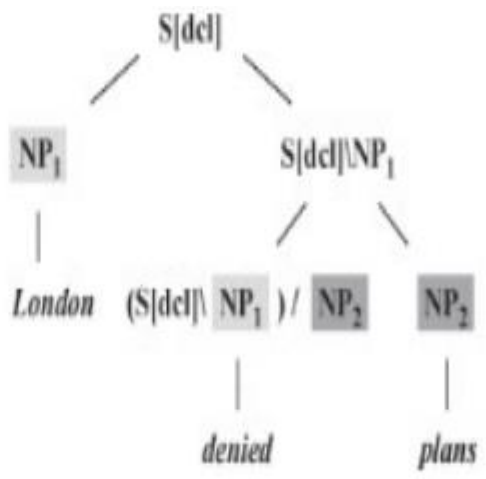
- **Structured Representation:** PSG features provide a structured and detailed representation of syntactic and semantic relationships within sentences.
- **Computational Modeling:** PSG frameworks with features support computational models for parsing, semantic role labeling, and other NLP tasks.
- **Interpretability:** The structured nature of PSG features enhances interpretability and understanding of linguistic structures.

Combinatory Categorial Grammar (CCG)

- Though the path feature is important for argument identification task, it is one of the sparse features and difficult to train or generalize.
- Dependency parsers generate shorter paths from the predicate to dependent words in the sentence and can be robust complement to the paths extracted from the PSG parse tree.
- Using features extracted from a CCG representation improves semantic role labeling performance on core arguments.
- CCG trees are binary trees and the constituents have poor alignment with the semantic arguments of a predicate.


London denied plans on Monday

w_h	w_a	c_h	i
denied	*London*	$(S[dcl]\backslash NP_1)/NP_2$	1
denied	*plans*	$(S[dcl]\backslash NP_1/NP_2$	2
on	*denied*	$((S\backslash NP_1\backslash (S\backslash NP)_2/NP)_3$	2
on	*Monday*	$((S\backslash NP_1\backslash (S\backslash NP)_2/NP)_3$	3



- Gildea and Hockenmaier introduced three features:
    - Phrase type: This is the category of the maximal projection between the two words, the predicate and the dependent word.
    - Categorial path: This is a feature formed by concatenating the following three values:
        - Category to which the dependent word belongs
        - The direction of dependence
        - The slot in the category filled by the dependent word
    - Tree Path: This is the categorical analogue of the path feature in the Charniak parse based system, which traces the path from the dependent word to the predicate through the binary CCGtree.

Combinatory Categorical Grammar (CCG) is a type of formal grammar used in natural language processing (NLP) to analyze the syntax and semantics of sentences.

It is particularly useful for understanding predicate-argument structures, which are essential for tasks such as parsing, semantic role labeling, and machine translation.

1. Categorical Grammar Basics:

Categorial grammar assigns categories to words and uses combinatory rules to derive the structure of sentences. Categories can be either simple (like N for nouns) or complex (like S\NP for a verb phrase that needs a noun phrase to form a sentence).

2. Lexical Categories:

Each word in a sentence is assigned a category that indicates its syntactic role. For example:

- John: NP (Noun Phrase)
- loves: S\NP/NP (A verb phrase that needs a noun phrase to its right and one to its left to form a sentence)
- Mary: NP

3. Combinatory Rules:

CCG uses combinatory rules to combine categories and build the structure of the sentence. Some common rules include:

- **Function Application:** Combines a function with its argument.
    - Forward application: X/Y Y -> X
    - Backward application: Y X\Y -> X

- **Function Composition:** Combines functions to form new functions.
    - Forward composition: X/Y Y/Z -> X/Z
    - Backward composition: Y\Z X\Y -> X\Z

4. Predicate-Argument Structure:

In CCG, the predicate-argument structure of a sentence can be derived by applying combinatory rules to the lexical categories of the words. For example:

- Sentence: "John loves Mary"
    - John: NP
    - loves: S\NP/NP
    - Mary: NP

  Using function application:

    - loves Mary: S\NP (Backward application: S\NP/NP NP -> S\NP)
    - John loves Mary: S (Forward application: NP S\NP -> S)

5. Semantic Representation:

CCG can be extended to include semantic representations, making it possible to map syntactic structures to their meanings. This involves assigning lambda calculus expressions to categories and combining them using the same combinatory rules.

Example:

For the sentence "John loves Mary":

- John: NP (λx. John(x))
- loves: S\NP/NP (λy.λx. loves(x, y))
- Mary: NP (λx. Mary(x))

Combining them:

- loves Mary: S\NP (λy.λx. loves(x, y)) (λx. Mary(x)) -> λx. loves(x, Mary)

- John loves Mary: S (λx. John(x)) (λx. loves(x, Mary)) -> loves(John, Mary)

Lexical Categories

1. "John" : NPNPNP
2. "loves" : (S\NP)/NP(S \backslash NP)/NP(S\NP)/NP
3. "Mary" : NPNPNP

Parse Tree Construction

1. **Lexical Insertion**: Each word is assigned its category.

   ```
   java
   Copy code
   John       loves           Mary
   NP      (S\NP)/NP           NP
   ```

2. **Forward Application**: The verb "loves" combines with its object "Mary".

   ```
   diff
   Copy code
           loves Mary
          ((S\NP)/NP) NP
   ---------------------->
               S\NP
   ```

3. **Backward Application**: The resulting S\NPS\backslash NPS\NP combines with the subject "John".

   ```
   diff
   Copy code
             John      loves Mary
              NP         S\NP
   ------------------------------->
                      S
   ```

Parse Tree Diagram

Below is a diagram of the parse tree for "John loves Mary" using CCG notation.

```
markdown
Copy code
            S
           / \
          /   \
         NP   S\NP
        John  /  \
            (S\NP)/NP NP
             loves   Mary
```

Explanation

1. **Lexical Categories**:

    - "John" is NPNPNP.
    - "loves" is (S\NP)/NP(S \backslash NP) / NP(S\NP)/NP.
    - "Mary" is NPNPNP.
2. **Combination**:
    - **Step 1**: "loves" ((S\NP)/NP(S \backslash NP) / NP(S\NP)/NP) combines with "Mary" (NPNPNP) using forward application, resulting in S\NPS \backslash NPS\NP.
    - **Step 2**: "John" (NPNPNP) combines with S\NPS \backslash NPS\NP (from Step 1) using backward application, resulting in SSS.

In summary, the parse tree demonstrates how the CCG framework allows for the composition of complex structures from simple lexical categories using combinatory rules, ultimately representing the predicate-argument structure of the sentence "John loves Mary".

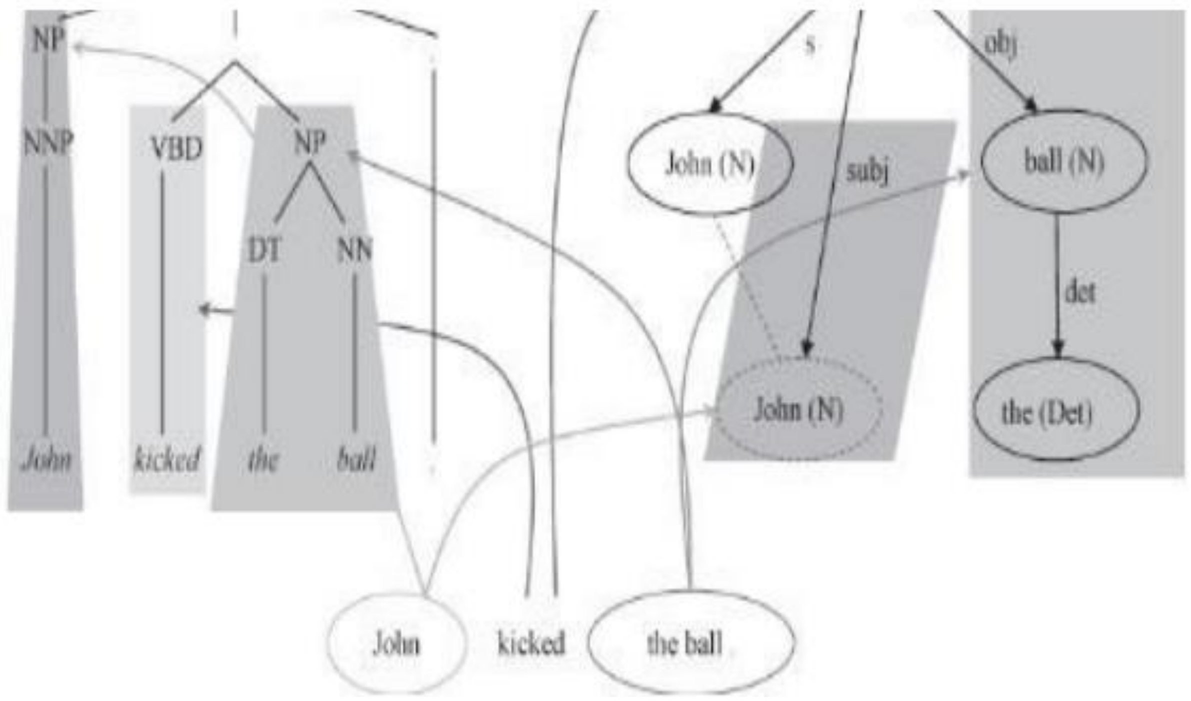
Tree-Adjoining Grammar (TAG)

- TAG has an ability to address long-distance dependencies in text.
- The additional features introduced by Chen and Rambow are:
    - ***Supertag path***: This feature is the same as the path feature seen earlier except that in this case it is derived from a TAG rather than from a PSG.
    - ***Supertag***: This feature is the tree frame corresponding to the predicate or the argument.
    - ***Surface syntactic role***: This feature is the surface syntactic role of the argument.
    - ***Surface subcategorization***: This feature is the subcategorization frame
    - ***Deep syntactic role***: This feature is the deep syntactic role of an argument, whose values include subject and direct object.
    - ***Deep subcategorization***: This is the deep syntactic subcategorization frame.
    - ***Semantic subcategorization***: Gildea and Palmer also used a semantic subcategorization frame where, in addition to the syntactic categories, the feature includes semantic role information.

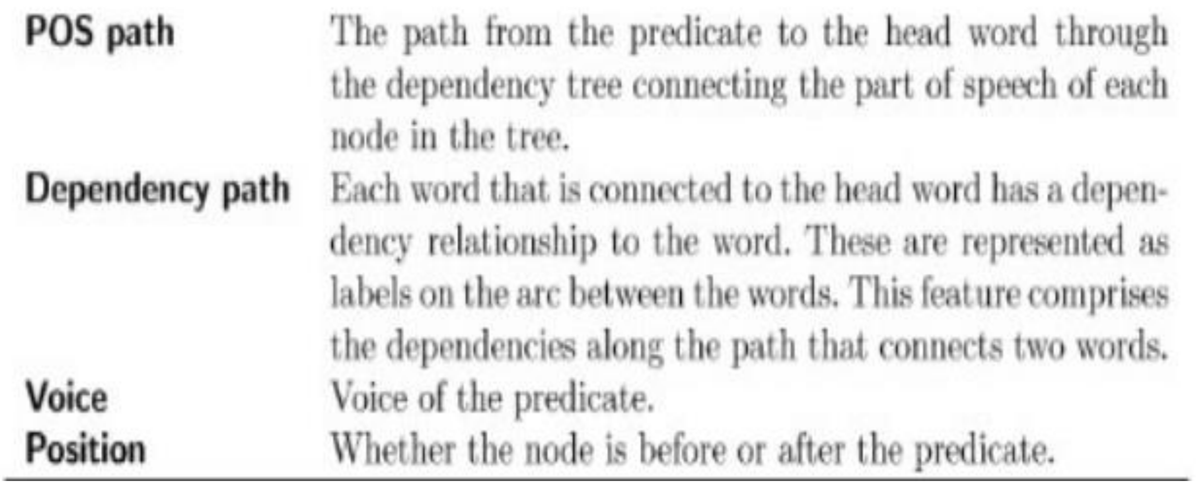
- A few researchers tried to use a tree kernel that identified and selected subtree patterns from a large number of automatically generated patterns to capture the tree context.
- The performance of this automated process is not as good as handcrafted features.

Dependency Trees

- One issue so far is that the performance of a system depends on the exact span of the arguments annotated according to the constituents in the Penn Treebank.
- PropBank and most syntactic parsers are developed in the Penn Treebank corpus.
- They will match the PropBank labeling better than the other representations.
- Algorithms which depend on the relation of the argument head word to the predicate give much higher performance with an Fscore of about 85.
- Hacioglu formulated the problem of semantic role labeling on a dependency tree by converting the Penn Treebank trees to a dependency representation.
- They used a script by Hwa, Lopez, and Diab and created a dependency structure labeled with PropBank arguments.
- The performance on this system seemed to be about 5 F-score points better than the one trained on the phrase structure trees.
- Parsers trained on Penn Treebank seem to degrade in performance when evaluated on sources other than WSJ.
- Minipar is a rule-based dependency parser that outputs dependencies between a word called head and another called modifier.
- The dependency relationships form a dependency tree.
- The set of words under each node in Minipar's dependency tree form a contiguous segment in the original sentence and correspond to the constituent in a constituent tree.



- The figure in the previous slide shows how the arguments of the predicate “kick” map to the nodes in a phrase structure grammar tree as well as the nodes in a Minipar parse tree.
- The nodes that represent head words of constituents are the targets of classification
- They used the features in the following slide



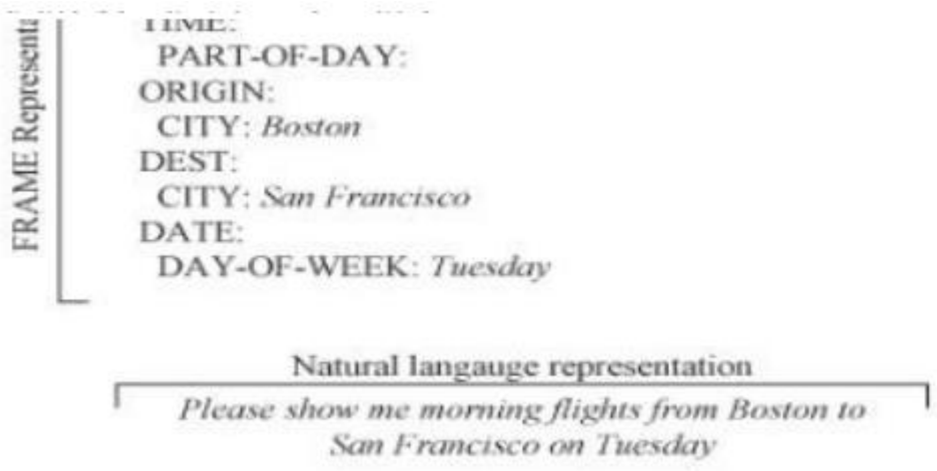
POS path	The path from the predicate to the head word through the dependency tree connecting the part of speech of each node in the tree.
Dependency path	Each word that is connected to the head word has a dependency relationship to the word. These are represented as labels on the arc between the words. This feature comprises the dependencies along the path that connects two words.
Voice	Voice of the predicate.
Position	Whether the node is before or after the predicate.

- An important question is how much does the full syntactic representation help the task of semantic role labeling?
- How important is it to create a full syntactic tree before classifying the arguments of a predicate?

- A chunk representation can be faster and more robust to phrase reordering as in the case of speech data.
- It was concluded that syntactic parsing helps fill a big gap using chunk based approach by Gildea and Palmer.

Meaning Representation

- The activity which takes natural language input and transforms it into an unambiguous representation that a machine can act on.
- This form will be understood by the machines more than human beings.
- It is easier to generate such a form for programming languages as they impose syntactic and semantic restrictions on the programs where as such restrictions cannot be imposed on natural language.
- Techniques developed so far work within specific domains and are not scalable.
- This is called deep semantic parsing as opposed to shallow semantic parsing.
- **Resources**
    - A number of projects have created representations and resources that have promoted experimentation in this area.
- **ATIS(Air Travel Information System)**
    - The Air Travel Information System (ATIS) is one of the first concerted efforts to build systems which build systems to transform natural language into applications to make decisions.
    - Here a user query in speech is transformed using a restricted vocabulary about flight information.
    - It then formed a representation that was compiled into a SQL query to extract answers from a database.
    - A hierarchical frame representation was used to encode the intermediate semantic information


FRAME Representa
TIME:
PART-OF-DAY:
ORIGIN:
CITY: Boston
DEST:
CITY: San Francisco
DATE:
DAY-OF-WEEK: Tuesday
Natural langauge representation
Please show me morning flights from Boston to San Francisco on Tuesday

- The training corpus of this system includes 774 scenarios completed by 137 people yielding a total of over 7300 utterances.
- 2900 of them have been categorized and annotated with canonical reference answers.
- 600 of these have also been treebanked.

- **Communicator**
  - ATIS was more focus on user-initiated dialog Communicator involved a mixed-initiative dialog.
  - Humans and machines were able to have a dialog with each other and the computer was able to present users real time information helping them negotiate a preferred itinerary.
  - Many thousands of dialogs were collected and are available through the Linguistic Data Consortium.
  - A lot of data was collected and annotated with dialog acts by CarnegieMellon university.
- **GeoQuery**
  - A geographical database called Geobase has about 800 Prolog facts stored in a relational databse.
  - It has geographic information such as population, neighboring states, major rivers, and major cities.
  - A few queries and their representations are as follows:
    - What is the capital of the state with the largest population?
    - Answer(C,(capital(S,C),largest(P,(state(S),population(S,P))))
      .

        - What are the major cities in Kansas?
        - Answer(C,(major(C),city(C),loc(C,S),equal(S,stated(kansas))))
    - The Ge0Query corpus has also been translated into Japanese,Spanish and Turkish.
- **Robocup:CLang**
    - RoboCup is an international initiative by the artificial intelligence community that uses robotic soccer as its domain.
    - A special formal language Clang is used to encode the advice from the team coach.
    - The behaviors are expressed as if-then rules.
    - Example: If the ball is in our penalty area, all our players except player 4 should stay in our half.
    - ((bpos(penalty-area our))(do(player-except our 4)(pos(half our))))

Systems

- Depending on the consuming application the meaning representation can be a SQL query, a Prolog query, or a domain-specific query representation.
- We now look at various ways the problem of mapping the natural language to meaning representation has been tackled.
- Rule Baes
- Supervised
- **Rule Based**
    - A few semantic parsing systems that performed very well for both ATIS and Communicator projects were rule-based systems.
    - They used an interpreter whose semantic grammar was handcrafted to be robust to speech recognition errors.
    - Syntactic explanation of a sentence is much more complex than the underlying semantic information.
    - Parsing the meaning units in the sentence into semantics proved to be a better approach.
    - In dealing with spontaneous speech the system has to account for ungrammatical instructions, stutters, filled pauses etc.

- Word order becomes less important which leads to meaning units scattered in the sentences and not necessarily in the order that would make sense to a syntactic parser.
- Ward's system, Phoenix uses a recursive transition networks (RTNs) and a handcrafted grammar to extract a hierarchical frame structure.
- It re-evaluates and adjusts the values of the frames with each new piece of information obtained.
- The system had the following error rates:
  - 13.2% for spontaneous speech input of which
  - 4.4% speech recognition word-error rate
  - 9.3% error for transcript input

- **SUPERVISED**
  - The following are the few problems with rule-based systems:
  - They need some effort upfront to create the rules
    - The time and specificity required to write rules restrict the development to systems that operate in limited domains
    - They are hard to maintain and scale up as the problems become more complex and more domain independent
    - They tend to be brittle
      - As an alternative statistical models derived from hand annotated data can be used.
      - Unless some hand annotated data is available statistical models cannot deal with unknown phenomena.
    - During the ATIS evaluations some data was hand-tagged for semantic information
    - Schwartz used that information to create the first end-to-end supervised statistical learning system for ATIS domain.
    - They had four components in their system:
      - Semantic parse
      - Semantic frame
      - Discourse

- Backend
  - This system used a supervised learning approach with quick training augmentation through a human in-the-loop corrective approach to generate lower quality but more data for improved supervision.
  - This research is now known as natural language interface for databases (NLIDB).
  - Zelle and Mooney tackled the task of retrieving answers from Prolog database.
  - The system tackled the task of retrieving answers from a Prolog database by converting natural language questions into Prolog queries in the domain of GeoQuery.
  - The CHILL (Constructive Heuristics Induction for Language Learning) system uses a shift-reduce parser to map the input sentence into parses expressed as a Prolog program.
  - A representation closer to formal logic than SQL is preferred for CHILL because it can be translated into other equivalent representations.
  - It took CHILL 175 training queries to match the performance of Geobase.
  - After the advances in machine learning new approaches were identified and existing were refined.
  - The SCISSOR (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representation) system uses a statistical syntactic parser to create a Semantically Augmented Parse Tree (SAPT).
  - Training for SCISSOR consists of a (natural language, SAPT, meaning representation) triplet.
  - KRISP (Kernel-based Robust Interpretation for Semantic Parsing) uses string kernels and SVMs to improve the underlying learning techniques.