Chapter 1. Finding the Structure of words

Human language is a complicated thing. We use it to express our thoughts and through language, we receive information and infer its meaning.

Linguistic expressions are not unorganized, though. They show structure of different kinds and complexity and complexity and consist of more elementary components whose co-occurence in context refines the notions they refers to in isolation and implies further meaningful relations between them.

In this chapter, we explore how to identify words of distinct types in human languages, and how the internal structure of words can be modeled in connection with the grammatical properties and lexical concepts the words should represent. The discovery of word structure is morphological. parsing words and their components:

words are defined in most languages as the smallest linguistic units that can form a complete utterance by themselves. The minimal parts of words that deliver aspects of meanings to them are called morphemes.

Tokens:

Sentence divided into segments, punctuations and words, eleminating white space

Suppose, for a moment, that words in English are delimited only by whitespace and punctuation

example: will you read the newspaper? will you read it? I won't read it.

lexemes:

The content of the concept provding by the lexical analyizer using set of alternative forms in the algorithm. The lexemes also consider words in the context called lemma's

example: Did you see him? I didn't see him. I didn't see anyone

Presents the problem of tokenization of didn't and the investigation of the internal structure of anyone.

Morphemes:

Morphemes properties of words and their components are usually called segments. The parts of speech of word are called morphemes or morphs

Morphological theories differ on whether and how to associate the properties of word forms with their structural components.

These components are usually called segments or morphs.

Example: 1. Whether will+you-read this the-newspapers?

2. Whether will+you-read+it? not will I-read+it

Typology:

Morphological typology divides into group of languages that phenomena provides prevelent characters.

Isolating(or) analytic:

languages include no or relatively few words that would comprise more than one morpheme.

Synthetic: Synthetic languages can combine more morphemes in one word and are further divided into agglutinative and fusional languages.

Agglutinative: Agglutinative languages have morphemes in one word associated with only a single function at a time

Fusional: language are defined by their feature-per-morpheme ratio higher than one.

Concatenative: language linking morphs and morphemes one after another.

Issues and challenges.

Morphological parsing tries to eliminate or alleviate the variability of word forms to provide higher-level linguistic units whose lexical and morphological properties are explicit and well defined. It attempts to remove unnecessary irregularity and give limits to ambiguity both of which are present inherently in human language.

- Irregularity
- Ambiguity
- Productivity

- Irregularity:

Morphological parsing is motivated by the quest for generalization and abstraction in the world of words. Immediate descriptions of given linguistic data may not be the ultimate ones. due to either their inadequate accuracy or inappropriate complexity, and better formulations may be needed. The design principles of the morphological model are therefore very important.

Irregular words like if their an sentence provide indirect meaning made abstractions for irregular morphology.

- Ambiguity:

Morphological ambiguity is the possibility that word forms can be understood in multiple ways in the concept.

- word, sentence, pharse are ambiguous words contains multiple meanings.

Types of Ambiguity:

Lexical Ambiguity: word has more than one meaning

Example: 1. Book (noun) → Text, novel, Artical

2. Back (noun) → financial, river, transactions

3. I can play cricket

give me that can

Syntatic Ambiguity:

To present in sentence structure specify possible arrangements of word in sentence.

this Ambigous words depends on grammatic roles

- Abhi saw the man with book
- Abhi saw the man carrying book
- Abhi saw the man through the book

Symantic Ambiguity:

Each word in the sentence have more than one meaning

- If a sentence has 10 words each having three meaning associated.
- number of possible interpretations

example:- 10 words

in in in $3*3*3*3*3*3*3*3*3*3 = 59049$

Madu loves his cat
deepu loves too
deepu also loves cat

Anaphoric Ambiguity:

to identify meaning of the sentence poovides parts of speech in the sentence and identify

when, where, by whom occurance of the sentence what was said

example: 1. Monkeys eat banana,
when they wake up
Here they means what?
- monkeys

2. Monkeys eat banana,
when theny are ripe
Here they means what?
- Banana.

Pragmantic :

understandmg speakers intension and also known as Informative ambiguty

example: you're late (criticize)

you're good (compliment)

Challenges of Ambiguty:=

1. Elongated words:

I am soo sorry - to longwords

It was too yummy

Shortcuts: please = pls

By the way = BTw

what = wat

ok = K

Ellipsis :

example: peter worked hard and passed exam, kelvin too

Interpritation.

Peter worked hard, peter passed exam

Kelvin worked hard, kelvin passed exam

Kelvin worked hard and also passed exam

Ambiguity punctuations:

women, without her man, is nothing

women ! without her, men is nothing

Productivity:

1. In India we have INLTK (India natural language tool kit) API (Application programming interface)

From Inltk import Setup

Tokenize:

from Inltk.Inltk import tokenize

tokenize (text, 'code-of-language')

Out:- Native language — code.

Hindi	hi
Punjabi	pa
Gujarathi	ga
Telugu	te
English	En
urdu	ur
malayalam	ml
Tamil	Tm

Code mixed language: combination of languages

example: English + Hindi — strpt: latn — code: hi-en

= (Hindi + Eng) Hinglish

Tamil + English → Tanglish — latn — tm-en

Malayalam + English → manglish — latn — ml-en

Get embeded Vectors:

If we give ur then the code converted to urdu and gives output as urdu

This return array of "Embeded Vectors"

```
from inltk.inltk import get_embedding_vectors
vectors = get_embedding_vectors(text,'code_of_language')
vectors = get_embedding_vectors('Bharath','hi')
```

Output: 2112

```
len(vectors)
```

Out:- 4

Predict 'n' words

```
from inltk.inltk import predict_next_words
predict_next_words(text, n, 'code_of_language')
```

Identify the language

```
from inltk.inltk identify_language
reset_language_identifying_models
identify_language(text)
identify_language("भारत")
```

Output:- hindi_hi

Bharath

In Indian language the product of the NLP is INLTK.

Command: pip install inltk (caps)

Here INLTK runs on CPU as a desired behaviour most of the learning models in product supported languages.

API

To set up INLTK in the CPU before that to run and import INLTK Application

Command :- from inltk.inltk import setup

- To import native languages and code of the language

```
from inltk
setup('code_of_language')
```

→ Morphological models:

Morphology: It is study of structure of word and word formation

Morpheme: The smallest meaning unit of a word that has meaning

Type of morphemes:

1. Free morphemes
    1. Lexical morphemes
    2. Functional morphemes
2. Bound Morphemes
    1. Derivational
    2. Inflectional

Free morphemes: The word can stand by themselves means it has own meaning

Eg:- Boy, girl, car, beauty.

lexical morpheme: Classify the words using pos.

Eg:- Nouns, Verbs and adjectives

Functional Morphemes: Built the class of the words using POS that have grammatical functions

Eg:- prepositions: of, to
conjuctions: And, But
Determiners: this, that
Pronouns : I, you
Verbs : Is, can

Bound Morphemes: This words cannot standalone By themselves cannot exist independently

Eg:- Boy+s
↓
Bound morpheme

mouses+s = mouses.

Derivational Morpheme: It changes the category of the word. and their are some exceptions. changes the grammatical category contains two types:

1. Class changing
2. Class maintaining

1. Class changing

Eg:- Driver Drive Beaty Beatiful
↓ ↓ ↓ ↓
noun Verb noun Adjective

2. Class maintaining.

Eg:- Adjust Adjustment (IIIrd person)
↓ ↓
Verb noun

Help Helpful
↓ ↓
Verb Adjective

Inflectional morpheme:

That are used to indicate grammatical functions of a word and provide tenses to the sentence.

Eg:- Reach + ed (past tense)
Eat + en (past participate)
Happ + ier (-er) (Comparative)
Happrest (superlative) (-est)

Dictionary lookup:

Morphological passing is a process by which word forms of a language are associated with corresponding linguistic desenptions In this context, a dictionary is understand As a data structure that directly enables obtaining some precomputed results, In our case word Analysis the data structures can be shared Lookup Operations are relatively simple as usually quick Dictionaries can be Implemented for instance, as-lists, binary search trees, has tables etc.

Unification-Based morphology:

It have been inspired by advances in various formal linglistic framework aiming as enabling complile gramatical discriptions of human languages.

A finite-state morphological models, both-surface and lexical forms are by themselves unstructed strings of atomic symbols In higher-level approches. Imguistic information is expressed by more approriated data

P: $\emptyset \rightarrow \{\psi\}$ p: form $\rightarrow$ {content}

functional morphology:-

This group of morophology models includes not only the ones following the methodology of functional morphology but even those related to it defines its modify using principles

of functional programming and type theory

It implementation are intended to be refused as programming libraries capable of handing the complete morphology of a language and to be incorporated into various kinds of applications

I: lexeme → {parameter} → {form}

D: lexeme → {parameter} → {lexeme}

L: context → {lexeme}

Morphology Induction:

We have focused on finding structures of words in diverse language supposing we know what we are looking for we have not considered the problem of discovering and inducing word structure without the human insight

There are several challenging issues about deleting words. structure just from the forms and their context.

They are ambiguity and irregularity in morphology.

In order to improve the changes of statical inference, lel learning of morphologyly for multiple language is proposed by resulting in discovery of abstract morphemes.

finding the structure of documents:

In human language, words and sentences do not appear randomly but usually have a structure. for example, combinations of words form sentences - meaningful grammatical units, such as statements, requests, and commands. Likewise, in written text, sentences form paragraphs - self-contained units of discourse about a particular point or idea. sentences may also be related to each other by explict discourse connectives such as therefore.

Automatic extraction of structure of documents helps subsequents natural language processing (NLP) tasks; for example, parsing, machine translation, and semantic

Sentence Boundary Detection:

Sentence boundary detection (also called sentence segmentation) deals with automatically segmenting a sequence of words tokens into sentence units. In written text in English and some other languages, the beginning of a sentence is usually marked with an uppercase letter, and the end of a sentence is explicity marked with a period (.), a question mark (?), an exclamation mark (!) or another type of punctuation.

- It was the best of times
- It was the worst of times
- It was the age of wisdom
- It was the age of foolishness

Division of tokens

'It', 'was', 'the', 'best', 'of', 'times', 'worst', 'age', 'wisdom', 'foolishness'.

Here we removed repeated words, white spaces and punctuations

- It was the best of times [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
- It was the worst of times [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
- It was the age of wisdom [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
- It was the age of foolishness [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

Topic Boundary Detection:

Topic segmentation (sometimes called discourse or text segmentation) is the task of automatically dividing a stream of text or speech into topically homogeneous blocks. That is, given a sequence of (written or spoken) words, the aim of topic segmentation is to find the boundaries where topics change.

Topic segmentation is an important task for various language-understanding applications, such as information extraction and retrieval and text summarization.

Append the parts of speech

> Import corpus

```
nlp = corpus ("package name");
doc = nlp (para)
"This is nlp document ' Document contains some paragraph".
list = nlp (list)
```

Out: ["'this', 'is', 'nlp', 'document', 'Document', 'contain', 'some',
'paragraph'"]

Out: "'This',
'is',
'nlp',
'document',
'Document',
'contain',
'some',
'paragraph'"

In: sent = list (doc.sents)

Out: 'This', 'is', 'nlp', 'document',
'document', 'contains', 'some', 'paragraph']

In: len (sents). [:. length of the sentence]

In: print (sents)

Out: 2

Table formation of text:

In: text: "Apple is looking at buying U.K starts for $1 billion"

In: pip install Table

In: from table import table;

In: Table.columns header = ["'text', 'pos', 'TAG', 'Explain Tag', 'shape',
'is_stop'"]

In: for token in doc:
Table.rows.append ([token.text, token.pos_, token.tag_,
explain (token.tag_ token.shape, token.is_stop])

Descriminative classification model:

In: print (Table)

Out:

Text	POS	TAG	Explain_Tag	Shape	Is_stop
Apple	Noun	xxx	Sing- Noun	xxxx	0

Methods:

Sentence segmentation and topic segmentation have mainly been considered as a boundary classification problem. Given a boundary candidate (between two word tokens for sentence segmentation and between two sentences for topic segmentation), the goal is to predict whether or not the candidate is an actual boundary (sentence or topic boundary).

In this we have two methods

- Generative
- Descriminative

Generative

The most commonly used generative sequence classification method for topic and sentence segmentation is the hidden markov model (HMM).

Naive Baye's Algorithm

Eg: frequency table.

S.No	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Descriminative classification model:

Descriminative classifiers aim to model [illegible] The most

likelihood table.

frequency table
↓
likelihood table
↓
Probability
↓
decision

Weather	yes	No	
overcast	5	0	$\frac{5}{14}=0.35$
Rainy	2	2	$\frac{4}{14}=0.29$
Sunny	3	2	$\frac{5}{14}=0.35$
	$\frac{10}{14}=0.7$	$\frac{4}{14}=0.29$	

By applying Baye's theorm

for yes

$$P\left[\frac{yes}{sunny}\right] = \frac{P\left[\frac{sunny}{yes}\right] * P(yes)}{P(sunny)}$$

$$P\left[\frac{sunny}{yes}\right] = \frac{3}{10} = 0.3$$

$$P(sunny) = \frac{5}{14} = 0.35$$

$$P(yes) = \frac{10}{14} = 0.7$$

$$\Rightarrow \frac{0.3 \times 0.7}{0.35} = \frac{0.21}{0.35} = 0.60$$

For No

$$P\left[\frac{No}{sunny}\right] = \frac{P\left[\frac{sunny}{No}\right] * P(No)}{P(sunny)}$$

$$P\left[\frac{sunny}{No}\right] = \frac{2}{4} = 0.5$$

$$P[No] = \frac{4}{14} = 0.29$$

$$P[sunny] = \frac{5}{14} = 0.35$$

$$= \frac{0.5 \times 0.29}{0.35} = \frac{0.145}{0.35} = 0.41$$

$$P\left[\frac{yes}{sunny}\right] > P\left[\frac{No}{sunny}\right]$$

Descriminative classification model:

Descriminative classifiers aim to model $P(y_i/x_i)$ of directly. The most important distinction is that whereas class densities, $p(x/y)$, are model assumptions in generative approaches, such as naive Bayes, in discriminative methods, discriminant functions of the feature space define the model.

eg:
```
Import numpy as np;
Import pandas as pd
dataset = pd.read_csv ("/Social Network_Ads.csv")
Dataset.head()
```

o/p:

	UserId	RegNo	G1	Age	EstSal	Purchased
0		156250	M	19	19000	0
1		156251	M	35	20,000	0
2		156252	F	26	43,000	0
3		156253	F	27	57,000	0
4		156254	M	19	76,000	0

```
Dataset.isna().Sum
```

o/p:

UserID	0
RegNo	0
gender	0
Age	0
Est Sal	0
Purchased	0

```
x = data.iloc [0:2; 2:4].values
y = data.iloc [0:0, 0:4].values
```

out x = [0:0; 19, 19000]
[0:0; 35, 20000]
[0.0; 26; 25000]
[0.0; 27, 30000]
[0.0; 29, 35,000]

out y = [0, 0, 0, 0, 0]

Regression:-

Import logistic Regression

(x.data, y.data, x-age, y- Purchased) .values

Overall x & y data is displayed.

Discriminative sequence classification methods.

In segmentation tasks, the sentence or topic decision for a given example (word, sentence, paragraph) highly depends on the decision for the examples in its vicinity. Discriminative sequence classification methods are in general extensions of local discriminative models with additional decoding stages that find the best assignment of labels by looking at neighbouring decisions to label an example.

It is used to remove repeated data, irrelavent data, noisy data, unwanted data.

Sequence classification model

↓

Pre-processing model

$$S_1 = \sum_{x \in c_1} (x - \mu_1)(x - \mu_1)^T$$

$$c_1 = \{(4,1), (2,4), (2,3), (3,6), (4,6)\}$$

$$c_2 = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$$

$$c_1 = \left[\frac{x_1+x_2+x_3+x_4+x_5}{5}, \frac{y_1+y_2+y_3+y_4+y_5}{5}\right]$$

$$= \left[\frac{4+2+2+3+4}{5}, \frac{1+4+3+6+6}{5}\right] = [3,4] = \mu_1$$

$$c_2 = \left[\frac{9+16+9+8+10}{5}, \frac{10+8+5+7+8}{5}\right] = [8.4, 7.6] = \mu_2$$

Apply coherent martices

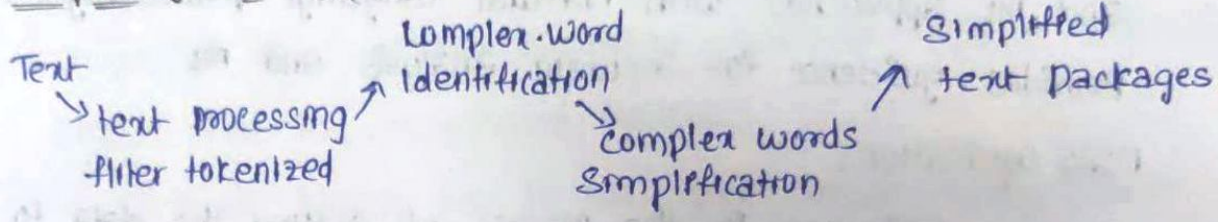
$$(x - \mu_1)(x - \mu_2)^T$$

$$[x - \mu_1] = [1 \quad -1 \quad -1 \quad 0 \quad 1]$$

Generate predications:

Finally the output is the predictions that the ML model will make on unknown data to understands information with in the context are related to each other.

Complexity of Approaches



Example: import nltk.data
nltk.data.load ('corpora')
from nltk.corpus import dataset
Vocabulary.categories()

In linguistics, complexity is a charateristic of a text but there are multiple measures and hence multiple implied definations in practice. In NLP, the measures are useful for descriptive statistics.

Complexity measures:

most popular ways of answering text complexity: how readable your text is (textual readability) and how rich it is (textual richness)

Performances of the Approaches:

For sentence segmentation in speech, performance is usually evaluated using the error rate (ratio of number of errors to the no of example). F1-measure (The harmonic mean of recall is defined as the ratio of the no of correctly retured sentence boundaries to the

$[x - \mu_2] = [9 - 8.4, 6 - 8.4, 9 - 8.4, 10 - 8.4]$

$= [0.6, -2.4, 0.6, -0.4, 1.6]$

Hybrid Approaches:

The Hybrid approach combine best of rule based and ML approches with Natural language processing. It provides guidence for accurate analysis and ML

Data aquisation:

Collection is the process of getting the data to the model data can come from valid resources like data websites, data where houses, life strings of the data like audio and vedio

cleaning and preprocessing:

Raw of data in most cases contains information that may not help in the analysis therefore after data aquisation is completely perform some cleaning and preprocessing that includes like removing out layers (outliers)

Data analysis and Interepret:

It is important at the begining of the analysis it reveals the information that means orgaining the data to predit to analysis process and predicate the data using patterns and algorithm after this analysis process to interact the data for training set. To model for machine learning process.

Building ML model:

Once data is ready for analysis process it start building machine learning model. Here the model is to provide plenty of data to learn and algorithm uses to learn about patterns.

no.of correctly returned sentence boundaries to the no. of all automatically estimated sentence boundaries.

filtering spam emails, language model, text-generate For sentence segmentation in text, reasearches have reported error rate results on a subset of tree the wall street journal corpus of about 27,000 sentences, Mikheev reports that wall street that his rule-based system.

If we search hoc it will give results many similar words using named entity recognisation.

UNIT-II

Syntax Analysis:

Parsing uncovers the hidden structure of linguistic input. In many application involving natural language, the underlying predicate-argument structure of sentence can be useful. The syntactic analysis of language provides a means to explicity discover the various predicate argument dependencies that may exist in a sentence.

Parsing Natural Language:

In a text-to-speech application, input sentence are to be converted to a spoken output that should sound like it was spoken by a native speaker of the language.

Parsing:

modeling the data in tree format

Parse → Latin word

Passing → Tree → Named entity recognization

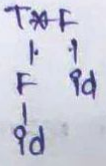
We have two types of parsing

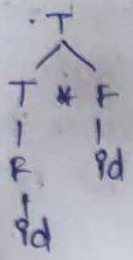
1. Bottom up approch
2. top down approch

Example:

E → T (Expression)
T → T*F (Travese)
F → T (Field)
F → Id (Field)
Id * Id (Identifier)

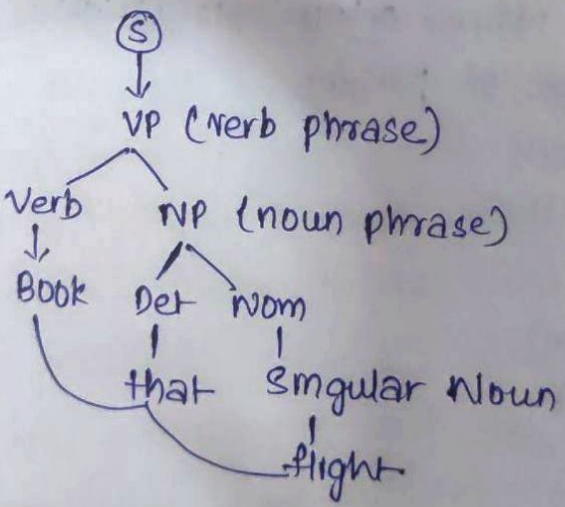
1. Id * Id → F * Id
2. F * Id → T (Id replaced with F)
   1
   Id
3. T * Id
   1
   F
   1
   Id

4.


5.


Top down approch.

S → VP
VP → Verb or NP
NP → Det or Nom
Det → That
Nom → Singular Noun
Singular noun → flight

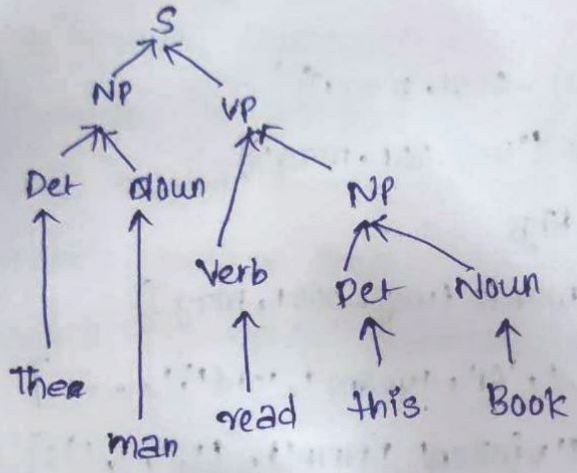


It is also known as recursive parsing starts from the "start" symbol and transforming into the input symbol mean parsing the nodes or extension of the links from top to bottom

Bottom up parsing

It is used to construct a parts tree from an input string. Parsing starts with the input symbol and construct the parts tree input string (bottom) up to the start symbol (top)

Bottom up approach



Tree bank: A Data Driven Approach to Syntax

To construct a tree using lexical analysis and maintaing combination of parts of speech to the data. Here parser is a Compiler that used to break data into smaller elements in lexical analysis phase. A parser takes input sequence of elements produce Output in the form of tree using text to speech application

Construct parts tree top down and bottom up to the following sentence using example role.

Tree bank.nlp

```
#download required files from nltk
import nltk
nltk.download('treebank')
True
```

```
# check what's in the treebank
from nltk.corpus import treebank
print(tb.fileids())
print(len(tb.fileids()))
```

out: ['wsj-0001.mrg', 'wsj-0002.mrg', ... 'wsj-009.mrg.'99']

```
# see words
print(tb.words('wsj-0001.mrg'))
print(len(tb.words('wsj.001.mrg')))
# see word with tags
print(tb.tagged_words(wsj-0001.mrg))
```

out: ['pierre', 'vinken', ',', '61', 'years', 'old', ',', ...]

[('pierre', 'NNP'), ('vinken', 'NNP'), (',', ',');]

```
# See the parsed sentence tree
print(tb.parsed_sents('wsj-0001.mrg')[0])
print("*********")
print(tb.parsed_sents('wsj-0001.mrg')[1])
```

out: (s(NP (pierre .NNP) (vinken NNP))

(ADJP (NP (Det 61) (NS years) (N old))

(VP (VB will) (VB join)

(NP (Det the) (NN board)

(NP as) (VB a) (Det Non-executive) (NN.Director)]

Pierre, vinken 61 years old will join

The board as a Noun-execution director

* * * * * * * *

[s (NP (NNP vinken)

(NP 's) (NN chairman)

(VB of) (Det the)

(NNP Daten) (VB publishing)

(NN group)]

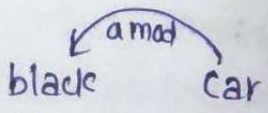
Representation of syntactic structure.

- Syntax Analysis using Dependency Graphs

The main philosophy behind dependency graphs is to connect a word - the head of a phrase - with the dependents in that pharse. The notation connects a head with its dependent using a directed connection

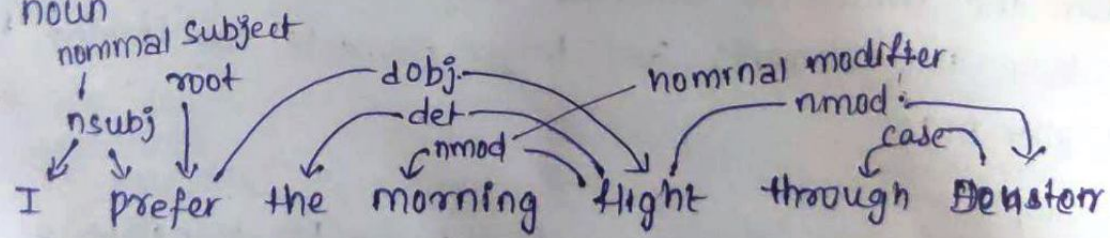
It is aprocess to analyze the grammartical structure and find out related words as well as the type of the relationship between them

Sample dependency relation between two words



There exists a relationship between car and black because black modifies the meaning of car. Here "car" acts as the "head" and "black" is a "dependency" of the "head" The nature of relationship here is "amod" stands for "Adjective modifier" It is an "adjective" or an "adjective phrase" that modifies a noun

ex:



The syntatic structure of sentence is discribed solely interms of the words and a set of directed bmary gramatical relations.

The relations between every word of the sentence is indicated using directed are in a typed dependency structure

- Here, the root the tree "prefer" form head of above sentence
- The relationship between any 2 word is marked a dependency tag. for instance, the word "flight" modifies meaning the noun "Denvor"
- you can notic a dependency from 'flight → Denvor' where the flight is head Denver is child or dependent.
- It is denoted by nmod which is nomial modifier

I ⇒ nsubj

Prefer ⇒ root

The ⇒ det ⇒ flight

morning ⇒ nmod ⇒ flight

flight ⇒ dobj ⇒ prefer

through ⇒ case ⇒ denever

Denever ⇒ nmod ⇒ flight

Syntax Analysis using phrase structure Trees,

A phase structure syntax analysis of the sentence derives from the traditional sentences diagrams in that partitions of a sentence into elements and larger elements are formed by merging smaller once

Example

- It tooks me more than two hours to translate a few Pages of english.

The sentence identify pos tag and the root node provide Proper noun pharse (ansubj – ansubject) and linked with using grammar tree

The another example sentence "the team is not perfoming well in the match"

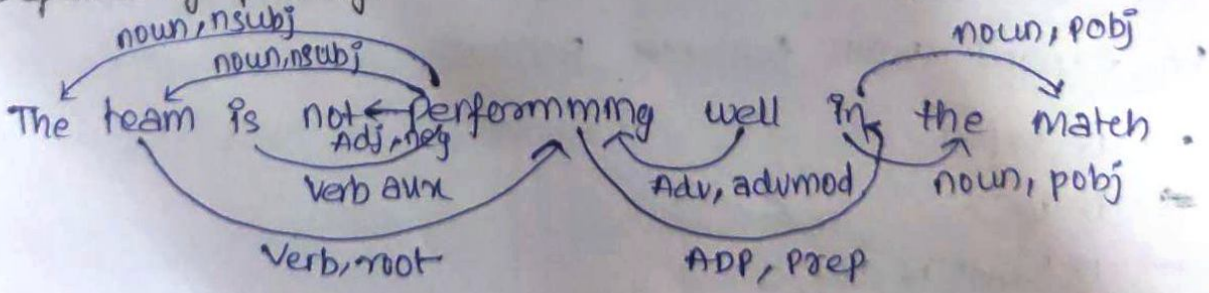
To Identify pos tag and dependency parsing and also provide Phrase structure tree.

Program:

```
import spacy
nlp = spacy.load("en")
doc = nlp("The team is not performming well in the match")
for token in doc:
    Print (str(token.text), str(token.pos_), str(token.dep_),
           str(token.head_))
Print()
```

```
>>> The , Noun , nsubj ⇒ performming
    team , Noun , nsubj (root) ⇒ performing
    is, Verb , aux ⇒ performing
    not , ADV, neg ⇒ performing
    Performing , verb, root ⇒ team
    well , ADV , advmod ⇒ performing
    In, ADP , Prep ⇒ performing
    The , Noun , pobj ⇒ In
    Match , Noun , pobj ⇒ In
```

Dependency parsing:



Pharse structure tree

(S (NP (N The) (N. team),
(VP (V Is) (ADV not),
(VP (V performming)
(ADV well)
(ADP Pn),
(NP (N The) (N match)))))

Parsing Algorithm:

To Given an Input sentence a parser produces an Output analysis of that sentence. Here parser do not need an explict grammar consider the following simple sentence can be derive (a and b or c) using and, or symbols

To start symbol with N, N → 'a'/'b'/'c'

Here '→' the symbol represents a sequence of steps called derivation

this sentence actions seperated with symbols and append with N

N → 'N' 'and' N

N → N 'or' N

Here the sentence derived from the start symbol N to expand with sentence actions

⇒ N 'or' N

⇒ N 'or' c

⇒ N 'and' N 'or' c

⇒ N 'and' b 'or' c

⇒ a 'and' b 'or' c

form N → 'a'/'b'/'c'

N → N 'and' N

N → N 'or' N

N → a and b or c

The sentence to expand with right most sequence form we arrage the derivation following rule.

Right side Rule

a and b or c

⇒ 'N' and b or c # N → a

⇒ 'N' and 'N' or c # N → b

⇒ 'N' and 'N' or 'c' # N → 'N' and 'N'

⇒ 'N' or 'c' # N ⇒ c

⇒ N or 'c' 1

⇒ N or N # N → 'N' or N

⇒ N

Shift-Reduce parsing:

~~Pars~~ To build a parser, we need to create an algorithm that can perform the steps in the preceding rightmost derivation for any grammar and for any input string. Every CFG turns out to have an automation that is equivalent to it, called a pushdown automaton

Parse Tree	Stack	Input	Action
		a and b or c	init
a	a	and b or c	Shift a
(N a)	N	and b or c	reduce N → a
(N a) and	N and	b or c	Shift and
(N a) and (N b)	N and b	or c	Shift b
(N (N a) and (N b))	N	or c	reduce N → b
(N (N a) and (N b)) or	N or	c	Shift or

(N(Na) and (Nb)) or c	N or ∅	reduce N→a shift c
(N(Na) and (Nb) or (Nc)	N or N	reduce N→c
(N (N (Na) and (Nb)) or (Nc))	N	reduce N→N or N
(N(N(Na) and (Nb)) or (Nc))	N	Accept!

Shift reduce parsing algorithm for the input "a and b or c" for the grammar rules using and append with N symbol

To construct partstree following section

a and b or c

(Na) · # using N→a

(Na) and

(Na) and b

(Na) and (Nb) # use N→b

(N(Na) and (Nb)) # use N→(Na) and (Nb)

(N (Na) and (Nb)) or

(N (Na) and (Nb)) or c

(N (Na) and (Nb)) or (Nc)) # N→c

(N (N.(Na) and (Nb) or (Nc))) # use M→(N(Na) and (Nb) or (Nc))

Allocation.

To store the data in the memory to represent the size by using array

a[5] =

1	2	3	4	5
0	1	2	3	4

Index 0 - starts

Index 4 - ends

Hyper Graphs: and chart parsing:

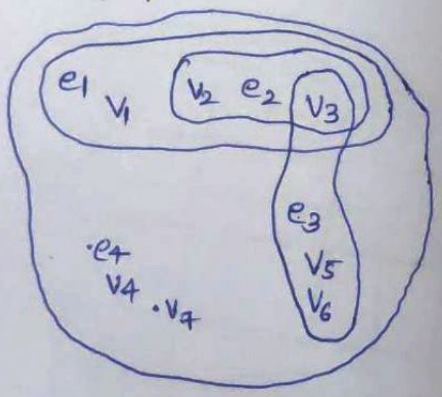
Shift-reduce parsing allows a linear time parse but requires access to an oracle. For general CFGs in the worst case, such a Parser might have to resort to backtracking, which means reparsing the input, which leads to a time that is exponential in the grammar size in the worst case.

Hyper graph is the graph of in which an edges can join any no.of vertices Here we have two types

1. Unidirected graph
2. Directed graph.

Unidirected graph:

Ex.



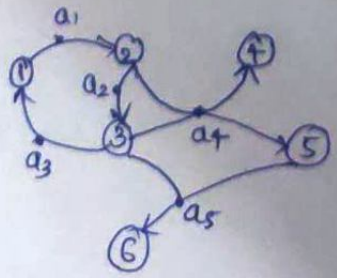
$e_1 = \{v_1, v_2, v_3\}$

$e_2 = \{v_2, v_3\}$

$e_3 = \{v_3, v_5, v_6\}$

$e_4 = \{v_4\}$

Directed graph:



$n = \{1, 2, 3, 4, 5, 6\}$

$E = \{a_1, a_2, a_3, a_4, a_5\}$

$a_1 = \{(1,2)\}$, $a_2 = \{(2,3)\}$

$a_3 = \{(3,1)\}$, $a_4 = \{(2,4)\}$

$a_5 = \{(5,6)\}$

Hyper graph parsing:

Example: Input grammar rules

$N \rightarrow$ 'N' and 'N'

$N \rightarrow$ 'a' | 'N' or 'N'

$N \rightarrow$ 'a' | 'b' | 'c'

$N \rightarrow NN'$

$N^{\wedge} \rightarrow$ 'and' N

$N^{\cdot} \rightarrow N\ NV$

$N \rightarrow$ 'or' N

$N \rightarrow$ 'a' | 'b' | 'c'

$N\ [0,5] \rightarrow N[0,1]\ N^{\wedge}[1,5]$

$N\ [0,3] \rightarrow N\ [0,1]\ N^{\wedge}[1,3]$

$N^{\wedge}[1,3] \rightarrow$ 'and' $[1,2]\ N[2,3]$

$N^{\wedge}[1,5] \rightarrow$ 'and' $[1,2]\ N[2,5]$

$N[0,5] \rightarrow N[0,3]\ NV[3,5]$

$N[2,5] \rightarrow N[2,3]\ NV[3,5]$

$NV[3,5] \rightarrow$ 'or' $[3,4]\ N[4,5]$

$N[0,1] \rightarrow$ 'a' $[0,1]$

$N[2,3] \rightarrow$ 'b' $[2,3]$

$N[4,5] \rightarrow$ 'c' $[4,5]$

CKy Algorithm. (Coicke, Kasami, younger)

Rules

S → NP VP [0.8]

NP → Det N [0.3]

VP → V NP [0.2]

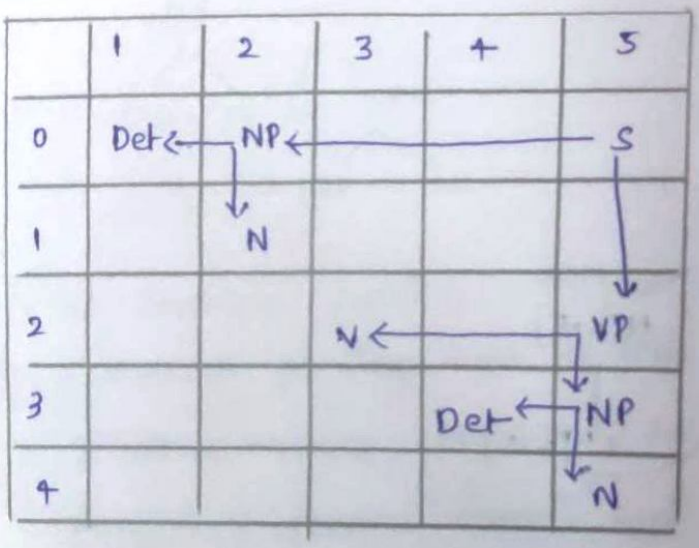
V → Includes [0.05]

Det → The [0.4]

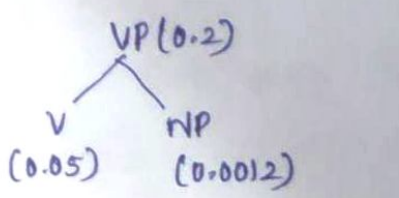
Det → a [0.4]

N → meal [0.01]

N → flight [0.02]

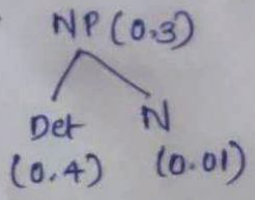


. Probabilistic cky



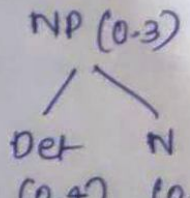
= 0.2 × 0.05 × 0.0012

= 0.000012



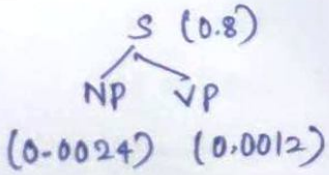
= 0.3 × 0.4 × 0.01

= 0.0012

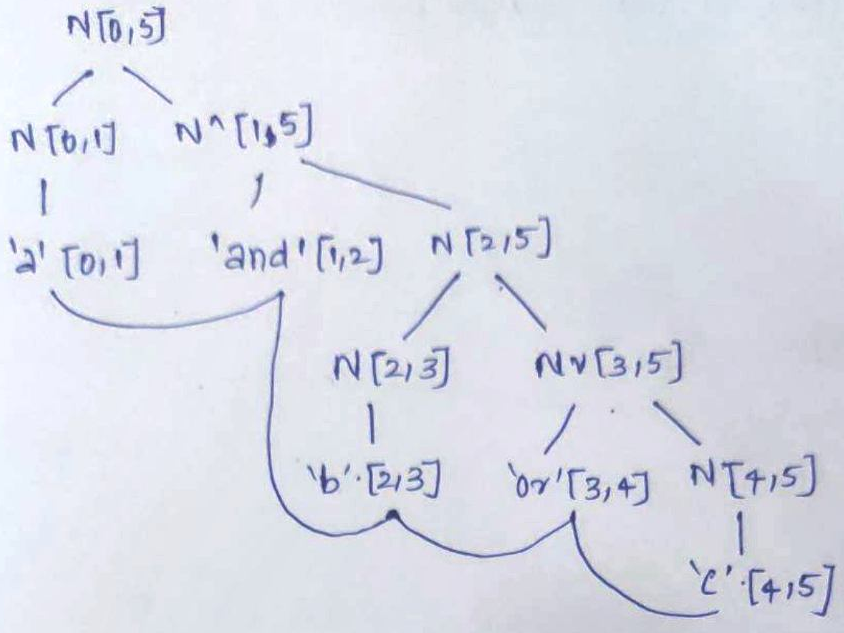


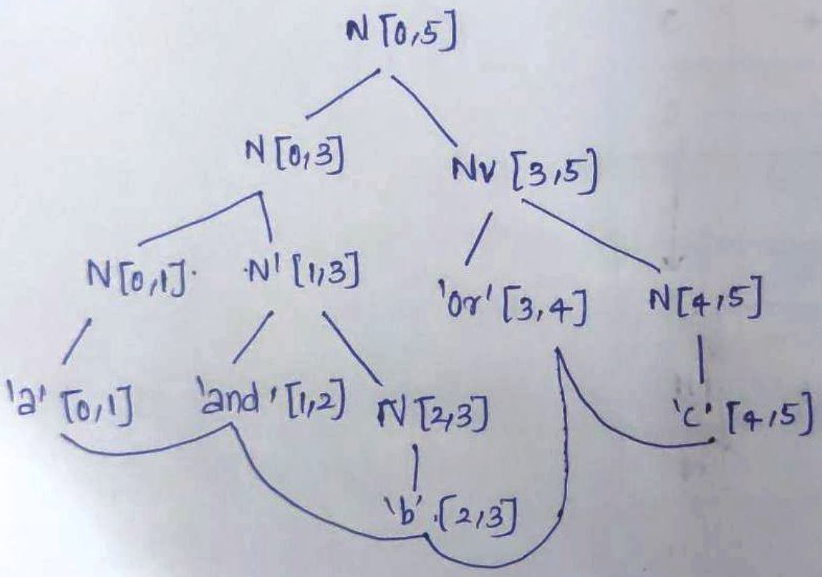
= 0.3 × 0.4 × 0.02

= 0.0024



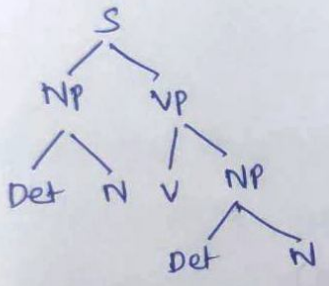
0.8 × 0.0024 × 0.000012


N[0,5]
N[0,1]
N^[1,5]
'a' [0,1]
'and' [1,2]
N[2,5]
N[2,3]
NV[3,5]
'b' [2,3]
'or' [3,4]
N[4,5]
'c' [4,5]


N[0,5]
N[0,3]
NV[3,5]
N[0,1]
N'[1,3]
'or' [3,4]
N[4,5]
'a' [0,1]
'and' [1,2]
N[2,3]
'c' [4,5]
'b' [2,3]

	1	2	3	4	5
0	Det 0.4	NP 0.0024			S 2.30
1		N			
2			V 0.05		VP 0.000022
3				Det 0.4	NP 0.0012
4					N 0.01

Tree



The sentence parsing with set of grammar rules to construct parsing tree in this algorithm context free grammar rules converted to CFG by using CNF rules (Comskey Normal form) and it is parsing in matrix form or chart table

CNF Rules:

Here using same CFG rules but some of the rules are changes

eg:- NP → The combination of direct terminal and non-terminal changes to direct terminal to non terminal

NP → Det N

NP → Det noun

PP → chages to

NP → Det nom

nom — noun PP.

Spanning Tree Algorithm:

- Minimum Spanning Trees and dependency parsing.

F ← [ ] → function

T ← [ ] → constant tree

Source [ ] → final score

for each v ∈ V

Min Edge ← arg t score [e] → select min edge weight

F ← F ∪ MinIEdge

for each e = (u,v) ∈ E do → stop condition

If T = (V, F)

else

return it spanning tree

c ← a cycle in F

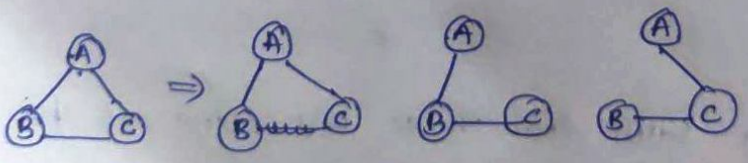
G ← contract (G, c) (G, root, score)

T ← EXPAND (T, c) → Expanded graph

Return T

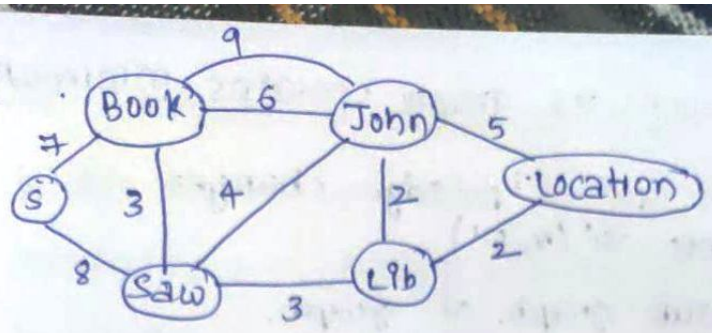
Minimual Spanning tree

1. Their is no connection or loop forming in the spanning tree



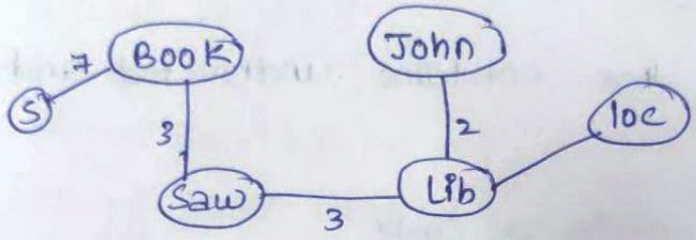
minimum spanning tree based on connected graph in directed graph to choose minimum weight of edge in the graph

eg:-



1. Select the root node is'
2. Remove all the loop in the graph.

3.



trace the tree with minimum weight if we have same weight we can consider two edges without forming loop

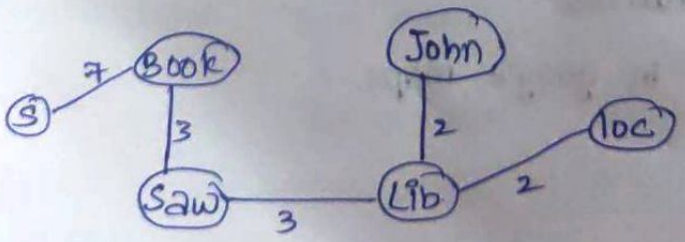
$7+3+3+2+2=17$ [minimum weight]

2nd method.

Arrange all the edges assending order of weight

{John, Lib} {Lib, Loc} {Book, saw} {saw, Lib} {saw, John}
2 2 3 3 4

{John, Loc} {Book, John} {S, Book} {S, saw} {Book John}
5 6 7 8 9



$7+3+3+2+2=17$

- Every undirected and connected graph contains minimum one spanning tree → edge changes.
- Spanning tree represents $G'(N, E')$
- Spanning tree is a sub graph of graph.

Properties:

- In directed graph, consider minimum edge weight of graph
- the rule of spanning tree contains undirected and Connected graph
- Spanning the never contain a cycle
- If it contains cyclic graph it can divided into unclydic possible graph
- possible to have more than minimum spanning tree the weight of same edge are same in the graph.

Applications:

- routing protocols
- civils networks
- clustering process
- comunication b/w group
- to calculate distance b/w nodes
- Telecommunication networks
- Finding shortest paths in google maps.

Models for Ambiguity Resolution in parsing.

In this section we focus on the modeling aspect of Parsing: how to design features and ways to resolve ambiguity in parsing.

Probabilistic Context-Free-Grammar.

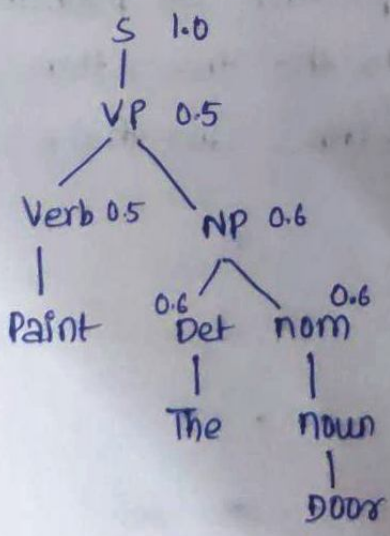
$S \rightarrow NP\ VP\ 0.8$
$S \rightarrow VP\ 0.2$ } 1.0

$NP \rightarrow Det\ noun\ 0.6$
$NP \rightarrow proper\ Noun\ 0.2$
$NP \rightarrow prep\ noun\ 0.1$ } 1.0
$NP \rightarrow Det\ NP\ 0.1$

$VP \rightarrow verb\ NP\ 0.5$
$VP \rightarrow verb\ 0.3$
$VP \rightarrow VP\ PP\ 0.2$ } 1.0

$PP \rightarrow pre\ p\ NP$ } 1.0

Paint the Door



$P(D) = 1.0 \times 0.5 \times 0.5 \times 0.6 \times 0.6 \times 0.2$
$= 0.018$

Assigning probabilities to possible parser of a sentence it is also known as probabilistic context free grammar In this assign probabilities most likely parses and tagging the POS provide final resultent to each phrase is "1" means : Partition each phrase with assigning value to possibilities of the phrase and final resultant is the output

Like Find ⇒ Assign ⇒ Return .

- output : PCFG

Example : The sentence

"paint the door with the hole"

Grammar Rules:

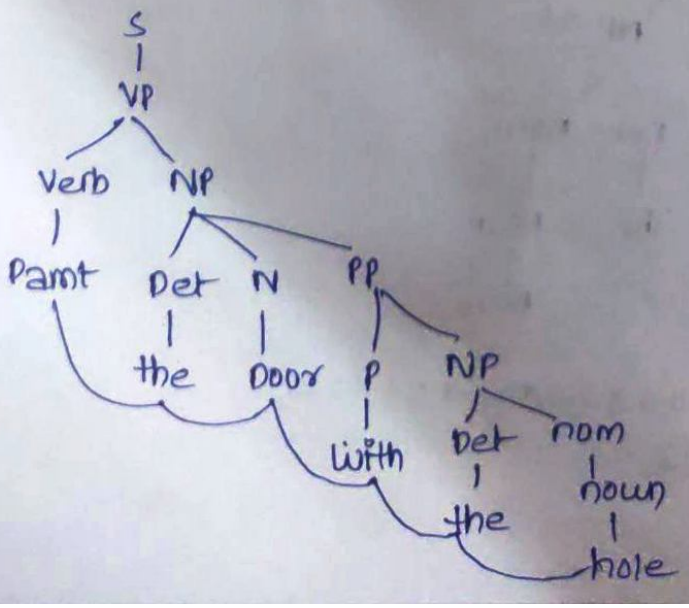
Noun → paint / Door / hole / bird

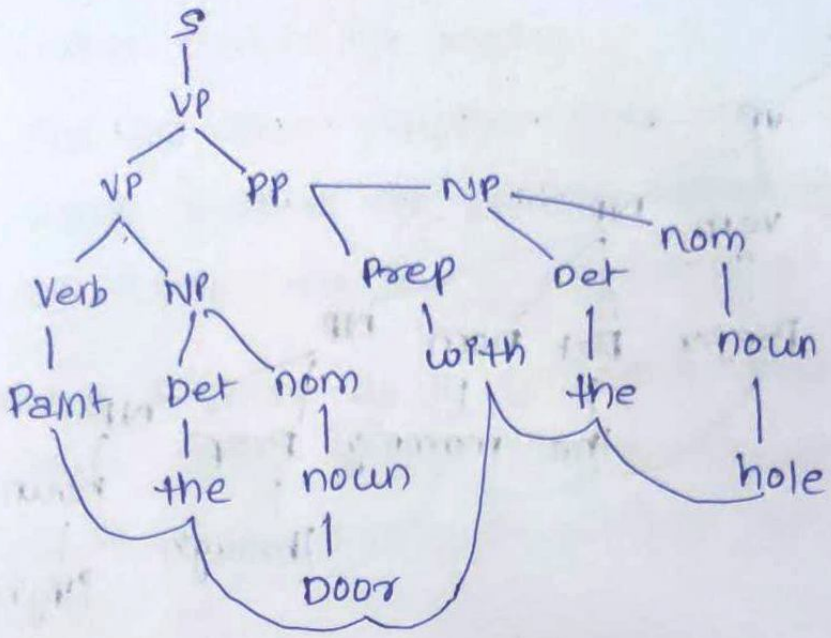
Verb → sleeps / sings / open / saw / paint

Prep → from / with / on / to

Pronoun → she / he / they

To construct parsing tree with product of probability values and caluclate derivation to the tree . Here to construct CFG then two possibilities are their.





NP → Det nom 0.3
NP → p noun 0.2
NP → pronoun 0.2
NP → Det N P 0.3

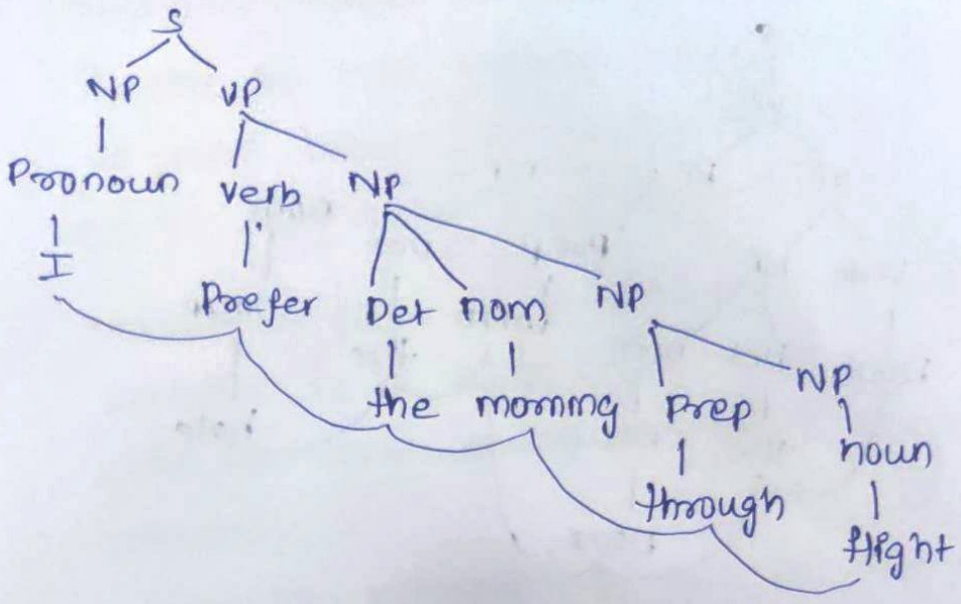
} 1.0

PCFG = P(D1) + P(D2)
= Result

The probability context free grammar and possibility

I prefer the morning through flight

S → NP VP
NP → Det nom
NP → Det nom NP
NP → Prep NP
NP → pronoun
VP → verb Np



The giraffe dreams.

S → NP VP

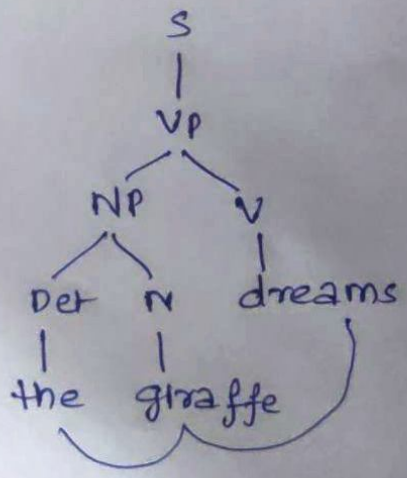
NP → Det N

VP → V NP

det → the / a / an

n → giraffe / apple

V → dreams / cats



Generative models for parsing.

To find the most plausible parse tree, the parser has to choose between the possible derivations each of which can be represented as a sequence of decisions.

Modeling a parse as a generative process allows the partstree to be broken into manageable parse. for which the corresponding probabilistic can be reliable estimated.

Probability estimation is easy for this sorts of models but choosing how to breakup the parse is something of a black art.

PCFG provide a redimate solution to this statical parsing Problem.

Representation.

1. The set of parts of speech tags
2. whether to pass lexical heads of the tree.
3. whether to replace words with their morphological stems

Decomposition:

1. The order in which to generate the tree.
2. In order of decision made in generating the tree
3. These decisions do not have to correspond to parsing decisions.

Assumptions:

1. Each rule in generative model

$$P(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

where $q(\alpha \rightarrow \beta)$

The probability for rule using $\alpha \rightarrow \beta$

$q(VP \rightarrow Vt\ NP) = 0.4$

$S \Rightarrow NP\ VP$ 1.0

$VP \rightarrow Vi$ (verb) 0.4

$VP \rightarrow V\ NP$ 0.4

$VP \rightarrow VP\ PP$ 0.2

$NP \rightarrow DT\ NN$ 0.3

$NP \rightarrow NP\ PP$ 0.7

$PP \rightarrow P\ NP$ 1.0

Lexicon

$Vi \rightarrow$ sleeps 1.0

$Vt \rightarrow$ saw 1.0

$NN \rightarrow$ man 0.7

$NN \rightarrow$ woman 0.2

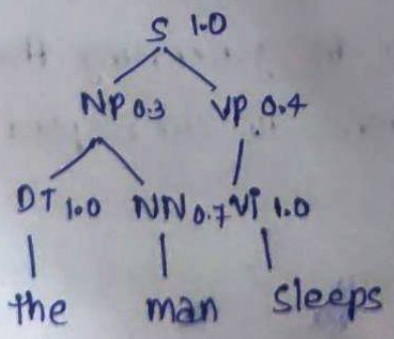
$NN \rightarrow$ telescope 0.1

$DT \rightarrow$ The 1.0

$IN \rightarrow$ The 0.5

$IN \rightarrow$ In 0.5

The man sleeps



$1.0 \times 0.3 \times 0.4 \times 1.0 \times 0.7 \times 1.0$

$= 0.084$

Generative Rule

$$P(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

Where $q(\alpha \rightarrow \beta)$

NP → DT 0.3
VP → Vi 0.4
VP → VP PP 0.2

Derivation	Rule	Prob
S	NP VP	1.0
NP VP	NP → DT NN	0.3
DT NN VP	DT → the	1.0
The NN VP	NN → man	0.7
The man VP	VP → Vi	0.4
The man Vi	Vi → sleeps	1.0
The man sleeps		

$P(t) = 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0$

$= 0.084$

Discriminative model for parsing:

Discriminative model is a class of logistic model used for classification or regression

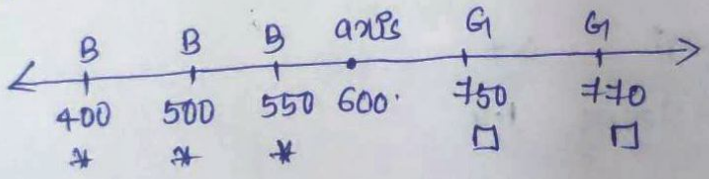
It makes predictions based on conditional probability on the data the distribution of data set to return by using example letters determined representing CIBIL score in HDFC Bank customers.

- Here axis point is known as linear regression boundary

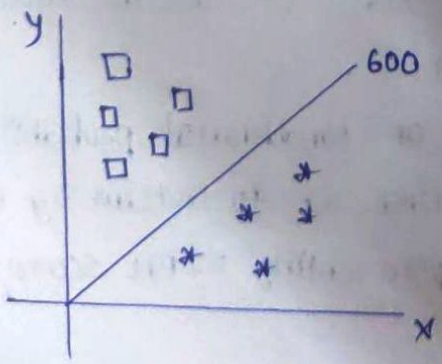
S.NO	CIBIL	Decision
1	500	B
2	550	B
3	750	G
4	770	G

Regression line estimate Average value in a given data set. It slightly differentiate sufficent and insufficent predictions in the data set.

Here dataset contains another dimension the differentiate by using x and y axis line is known as quantail regression or necessary condition analysis. In this situation regression created bounary



Years	S.NO	CIBIL	Decision
2	1	500	B
4	2	550	B
3	3	750	G
5	4	770	G



The data set contains collection of sufficent and insufficent predicates and regreate using axis boundary in existing graph. (14)

Multilingual issues :

Tokenization, case and encoding.

Tokenization:

It is a process of converting a sequence of characters into tokens. It performs lexical analysis contains lexer or tokenizer These generally combine with a parser together analyse the syntax of program.

lexical syntax:

The regular expression alphabets consists of single characters of source code

Grammatical syntax:

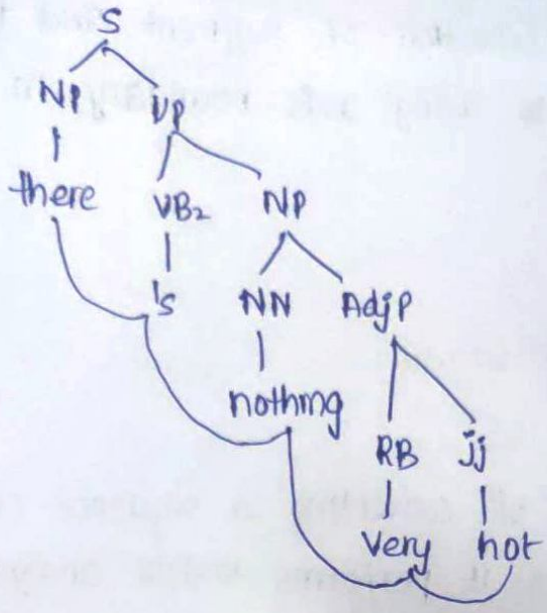
It is usually a context free language whose alphabet consist of tokens produced by lexer.

Let us determine example sentence

eg:- there is nothing very hot

tokenization :

```
S((NP There)
  (VP (VB2's)
    (NP (NN nothing)
      (Adj (RB very)
        (JJ hot)))))
```



Tokenization classify sentence into smallest units Tokenize produces a tree instead of grammar being return in natural language in terms of individual characters in the parsing

Word segmentation

this is a technique to divide set of words in a sentence we generally compute a score for each word to signify importance in the sentence. this used in information retrival and text mining.

The segmentation measures frequency of the word in a sentence this highly depends on the sentence length.

Eg: S_1 : earth is the third planet from the sun

S_2 : jupiter is the largest planet

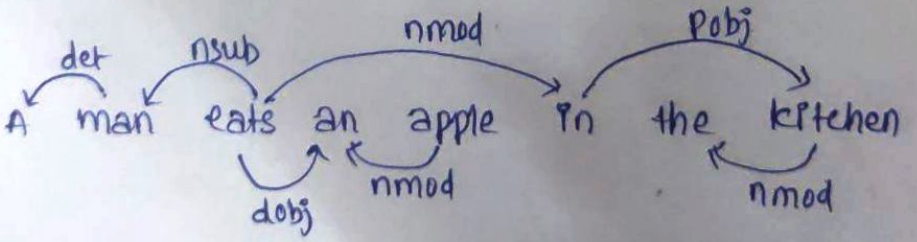
word	S_1	S_2	frequency
earth	1/8	0/5	log (2/1) = 0.3010
is	1/8	1/5	log (2/2) = 0
the	2/8	1/5	log (2/3) = -0.1760

third	1/8	0/5	log(2/1) = 0.3010
Planet	1/8	1/5	log(2/2) = 0
from	1/8	0/5	log(2/1) = 0.3010
Sun	1/8	0/5	log(2/1) = 0.3010
Jupiter	0/8	1/5	log(2/1) = 0.3010
largest	0/8	1/5	log(2/1) = 0.3010

Dependency parsing analysis shows syntatic dependencies need to be aware of the morphemes with in the words eg:- morpheme boundaries with in a boundary are shown using '+'

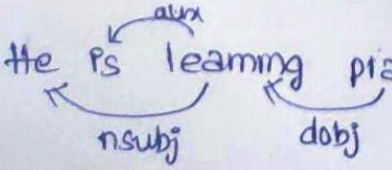
Morphology :

In many languages the notion of splitting up tokens using spaces is problematic because each word can contain several components, called morphemes, such that the meaning of a word can be thought of as composed of the combination of the meaning of the morphemes.



the + kitchen ⇒ NP ⇒ Det N

A + man ⇒ NP ⇒ Det N



learning + piano

VP = V + N

UNIT - 3

Semantic Analysis

→ NLP is a field which aims to give machine the ability to understand natural language.

→ semantic analysis is a sub topic of it.

→ semantic analysis means giving exact meaning of text.

→ Role of semantic analyser is to check text for meaningfulness.

Semantic Analysis

- meaning of individual word (lexical semantics)
- combination of words. (sentences)
  - Relationship exist between words
  - The word order
  - context
  - semantic structure
  - Real world knowledge

Composition Semantics

It involves how word combines to form larger meaning.

→ It says that meaning of each word matters & Syntax and way in which the sentence are constructed plays an important role as well.

Ex: I like you and you like me are same words but meaning are different

I → subject	you → sub
y → obj	me → obj

→ lexical semantic + composition semantic = Semantic analysis

→ called principle of compositionality.

Basic Building blocks of semantic system

Entities: It represents the individual such as particular person, location, etc.

ex: Rama, Chennai.

Concept: It represents general category of individuals like person, vehicle, animals

Relations: Relation between entities and concept.

ex: chennai is a location.

Predicate :- It represent Verb structure.

ex: case grammar and semantic rules.

Case grammar :- A form of grammar in which structure of sentence

Approaches :-

→ First order Predicate logic FOPL

→ Semantic Nets

→ case grammar.

FOPL → It convert sentence into logical form

ex:- 1) Jack loves jill

loves (Jack, jill)

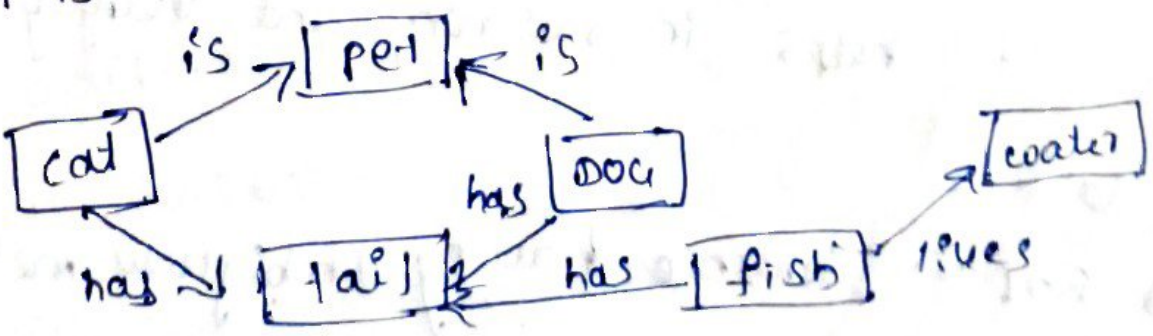
2) suman takes Engg or pharmacy.

takes (suman, Engg) or takes (suman, pharm).

Symantic Nets :-

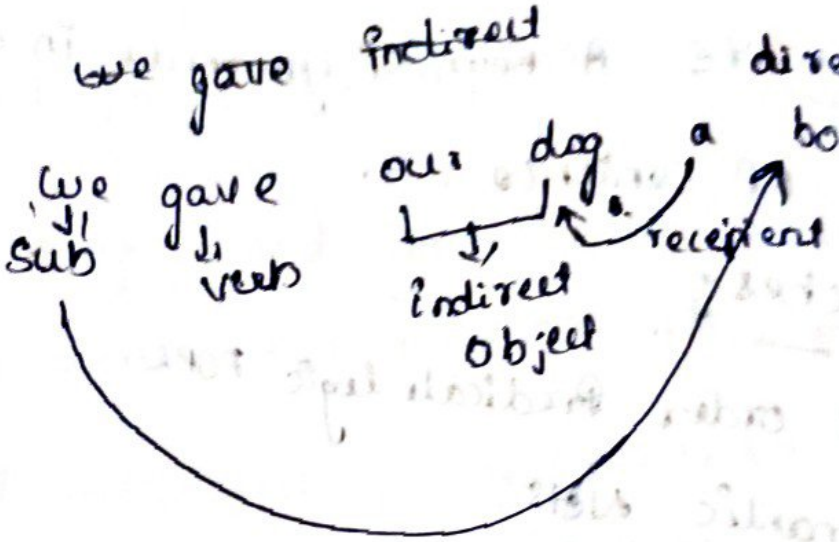
→ It shows Semantic Relations b/n concept.

It is used form knowledge Representation.



Case grammar :-

Case grammar :- structure of sentence is analyzed



Early semantic Analysis Important means in real word Posts, reviews, feedbacks, likes etc likes purpose using ~~###~~ analysis.

Semantic Interpretation:

→ It helps in linking the linguistic elements of a sentence to the non linguistic elements.

→ It helps in representing unambiguous data at lexical level.

→ It helps in reasoning and verifying common data

→ Explain sentence having ambiguous meanings, It should account

a) resolve the ambiguities of words in context

b) Identify meaningless but syntactically well formed sentences

c) Identify syntactically or transformationally unrelated paraphrases of a concept having the same semantic content

Details about the Relations:

Hyponymy: It illustrates the connection b/n a generic term is known as hypernym, while the occurences are known as hyponyms.

Homonymy: It may be described as words with the same spelling or form but diverse and unconnected meanings.

polysemy: polysemy is a term or phrase that has a different but comparable meaning. To put it another way, polysemy has the same spelling but various and related meanings.

⇒ Antonymy: It is the relationship between two lexical items that includes semantic components that are symmetric with respect to an axis.

meronomy: It is described as a logical arrangement of letters and words indicating a component portion of or member of anything.

Structural Ambiguity:

as talks of structure, we generally refer to the syntactic structure

word sense disambiguation

It is an automatic process of identifying the context of any word, in which it is used in the sentence. In natural language one word can have many meanings.

Eg: The word 'light' could be meant as not very dark or not. This is done by word sense.

Relationship Extraction:

In a sentence, there are a few entities that are co-related to each other. Relationship extraction is the Semantic relationship b/n these entities. In a sentence,

"I am learning mathematics",

2 entities 'I' & 'mathematics' the Relation b/n them 'learn'.

predicate argument structure

Predicate argument structure is based on the function features of lexical items (most often verbs). The function features determines thematic roles to be played by the other words.

Ors The police detained the suspect at the scene of the crime

who did what to whom at where?

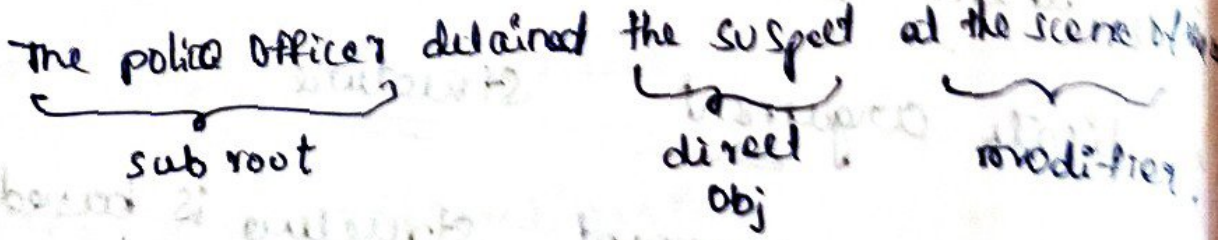
↑ ↓ ↓ ↓

The police officer detained the suspect at the scene of the crime

understanding a sentence = knowing "who what (to whom, when, where, why ---)

Verbs correspond to predicates (what are their arg (and modifiers) identify who did to whom, where, when, why, etc).

Syntatic parsing:



Meaning Representation:

Verbs describe events or states

Tom broke the window with a rock

The window broke.

The window was broken by Tom / by a rock

we could translate Verbs to (logical) predicates: Sub = first arg, obj = second arg

does not capture that similarities in meaning.

break (Tom, window, rock)
break (window)
break (window, Tom)
break (window, rocks).

semantic / Thematic Roles

Verbs describe events or states

Tom broke the window with a rock
The window broke
The window was broken by Tom / by a rock.

Roles: refer to participants of these events:

Agent (who performed the action) : Tom
patient (who was the action performed on) : window
Tool/ Instrument (what was used to perform the action) : rock

Semantic Roles (agent, patient) are different to Grammar Rules (Sub, Obj).

Ex: John broke the window (or) with a rock
Agent patient Instrument

Break (Agent, patient).
Break (Agent, patient, instrument)

System Paradigms

1. System Architecture :

(a) Knowledge based : As the name suggests, these systems use a predefined set of rules of a knowledge base to obtain a solution to a new problem.

(b) Unsupervised : These systems involve the methods . It is based on the fact that words that are related to each other can be found in the definations.

(c) Supervised methods :

Sense - annoteted corpora are used to train machine . The Rules of the grammer will have an extra field to hold semantic attachments .

Predicate Argument structure

It is an verb - centered approach

ex: I want Turkish food NP want NP

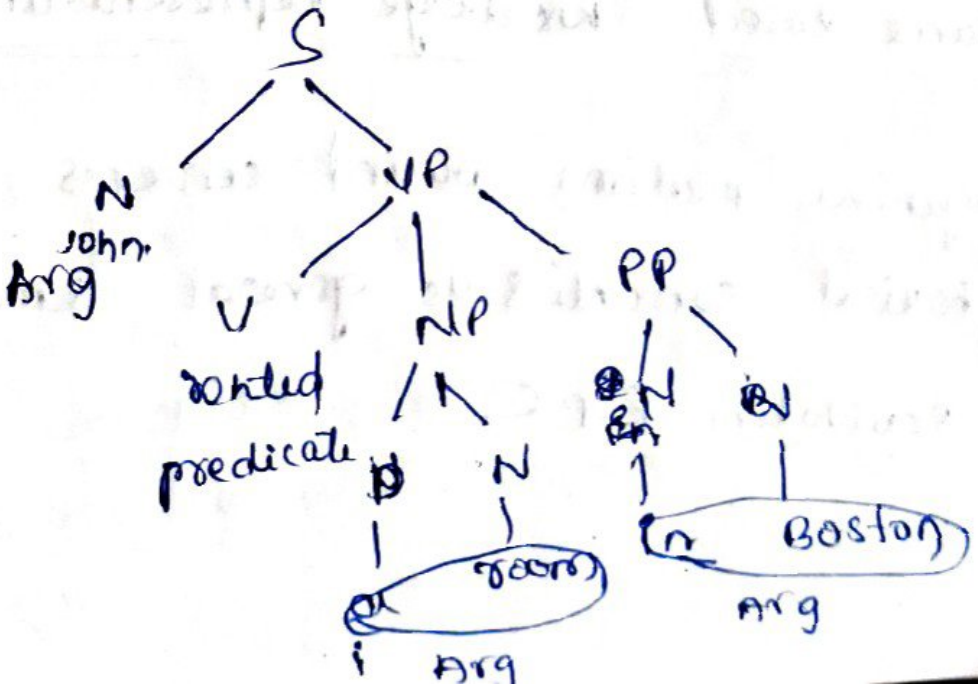
I want to spend less than five dollers NP want VP

I want it to be close by here NP want NP InfVP

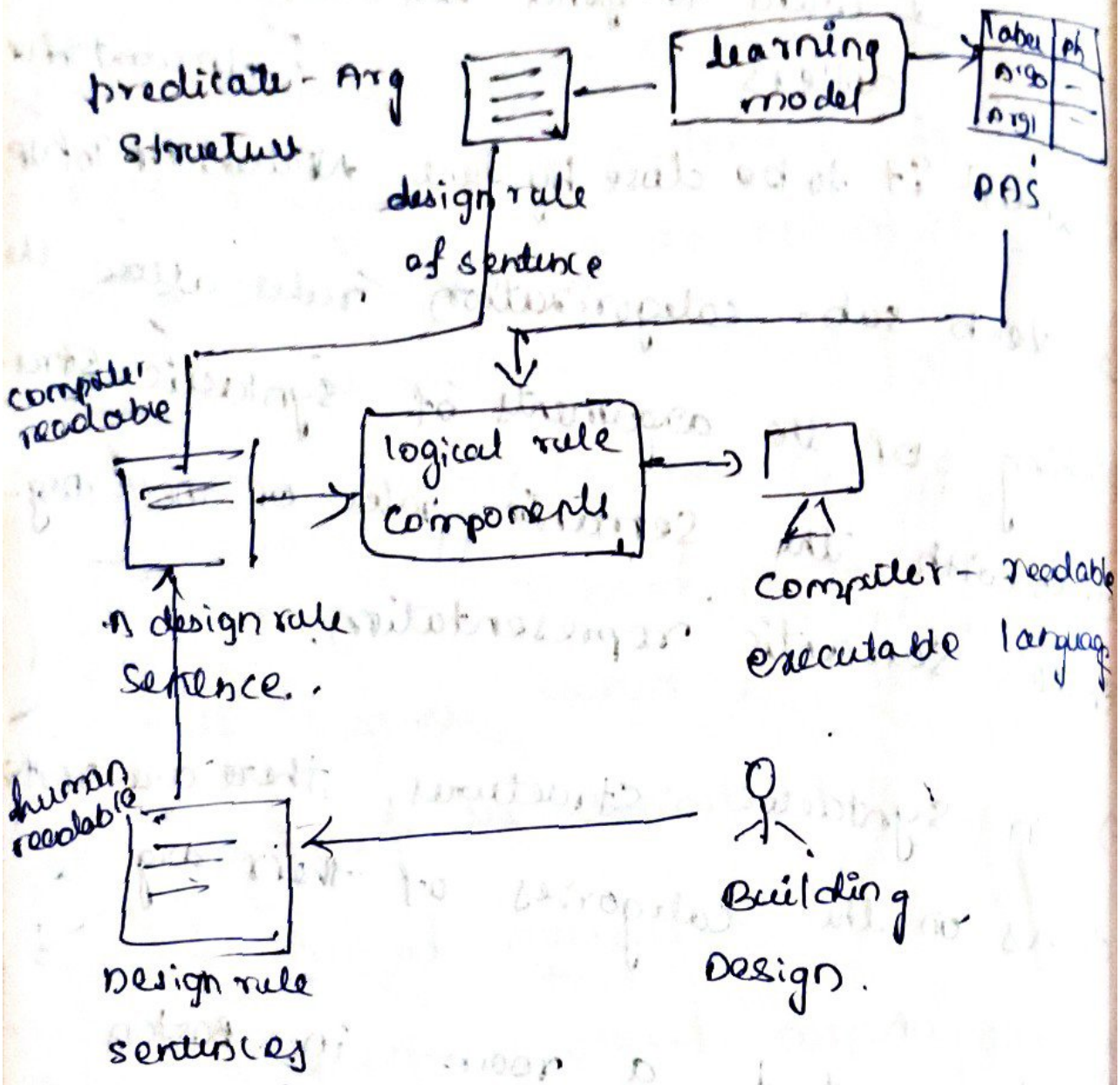
→ verb sub-categorization rules allow the linking of the arguments of syntactic structures with the semantic roles of these arg. in the semantic representation.

→ In syntactic structures, there are restrictions on the categories of their arg.

ex: John rented a room in Boston



Representation

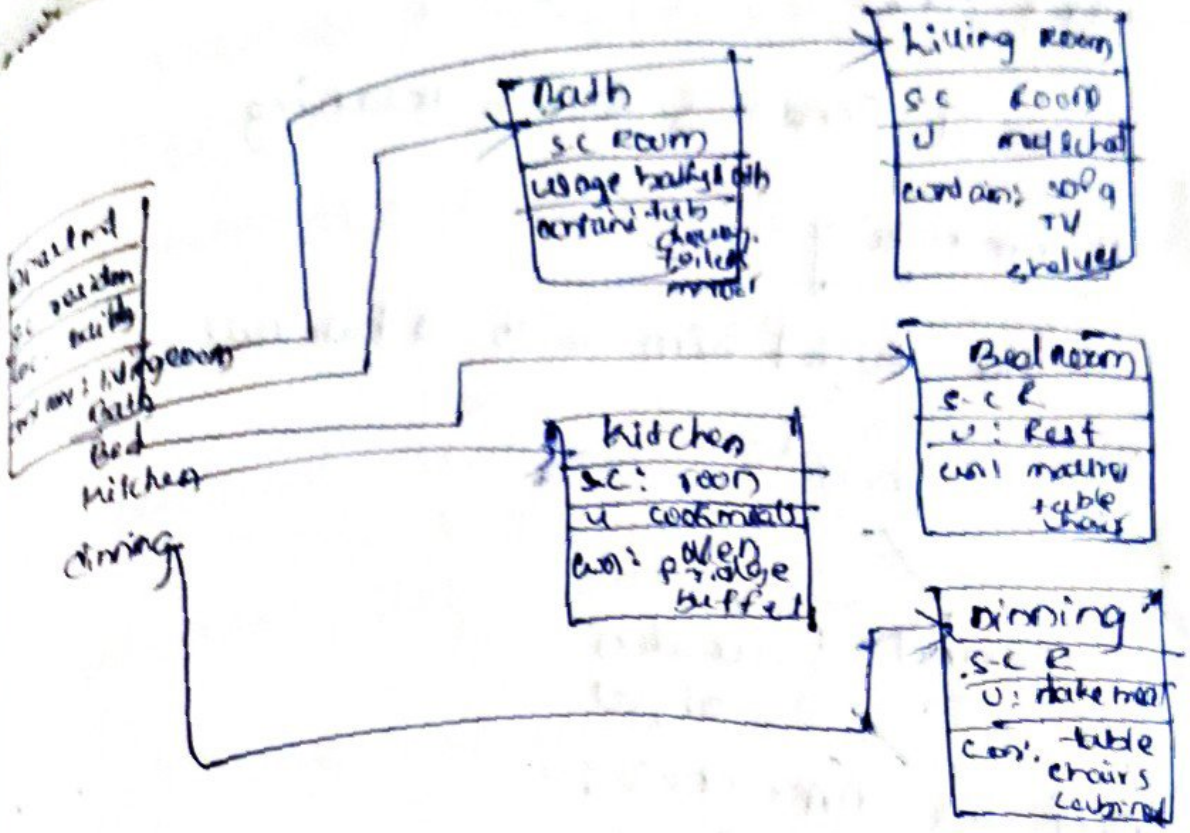


Frame Based knowledge Representation

→ organize relations around concepts

→ Lexical semantics is general semantic

→ equivalent POP C.



Frequency, Speed, time

→ John broke the window — breaker
→ The window broke — broken thing
→ John always breaking things — breaking frequency

→ we ate dinner — eater
→ we already ate — eaten thing
→ the pies were eaten up quickly — eating speed.

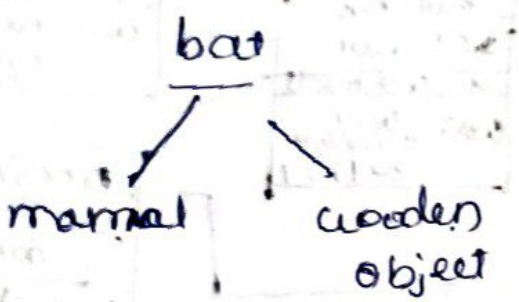
→ open vo: — opener.
→ Some left the door open — opened thing.
→ John opens the window — opened time.
at night

word sense:

word sense & sense meaning

Ambiguity

She killed him with a base ball.



Different Approches:

→ knowledge Based Approch:

Resources required

1). Raw corpora

2) machine readable dictionare

ex: Thdword.

how its done?

The bank can guarantee deposits will eventually cover failure.

let use example of the word "bark"

one meaning is → outer covering of the tree

2nd meaning → Sound made by a dog.

so here same word different meaning.

ex: 1 sentence: "cinnamon comes from the bark of the cinnamon tree".

2 sentence: "the dog barked at the sharger"

this sentence is passed to an algorithm for sentiment analysis. "bark" and "barked" might mean the same meaning.

so, we can understand that, same words can mean differently based on the usage of the word in a particular sentence. the usage of words defines about their meaning. But the problem lies that, in NLP, while dealing with text data, we need some way to interpret the different words with different meaning.

WSD Applications:

WSD has many applications in various text processing and NLP fields.

→ WSD can be used alongside lexicography. Much of the modern lexicography is corpus based. WSD, used in lexicography can provide

significant textual indicators.

→ WSD can also be used in Text mining and Information Extraction tasks. As the major purpose of WSD is accurately understand the meaning of a word.

→ In a security point of view, a text system should be able to understand the difference b/n a coal "mine" and a land "mine".

→ WSD used for information Retrieval purposes. IRS works through text data primarily based textual info: knowing the relevance of using a word in any sentence.

Challenges in word sense Disambiguation

WSD faces a lot of challenges and problems.

→ the most common problem is the difference b/n various dictionaries of text corpus.

different applications need different alg and that is often a challenge for WSD.

A problem also arises is that words cannot be divided into discrete meanings. words often have related meanings and this causes a lot of problems.

Systems: Rule based
Dictionary- and knowledge-based methods

These methods rely on text data like dictionaries, Thesaurus, etc. It is based on the fact that words that are related to each other can be found in the dic. nitions. The popularly used lesk method.

Supervised methods:

Sense-annotated corpora are used to train machine learning models. But, a problem that may arise is that such corpora a very tough and time consuming to create.

Semi-supervised methods:

Due to the lack of such corpus, word sense disambiguation algorithms use semi-supervised methods. The process starts with a small amount of data, which often manually created.

unsupervised methods:

unsupervised methods pose the great challenge to researchers and NLP professionals. A key assumption of these models is that similar meanings and senses occur in a similar context. They are not dependent on manual efforts hence can overcome the knowledge acquisition deadlock.

unsupervised Lesk Algorithm:

Lesk Alg is a classical word sense disambiguation alg introduced.

The lesk alg is based on the idea that words in a given region of the text

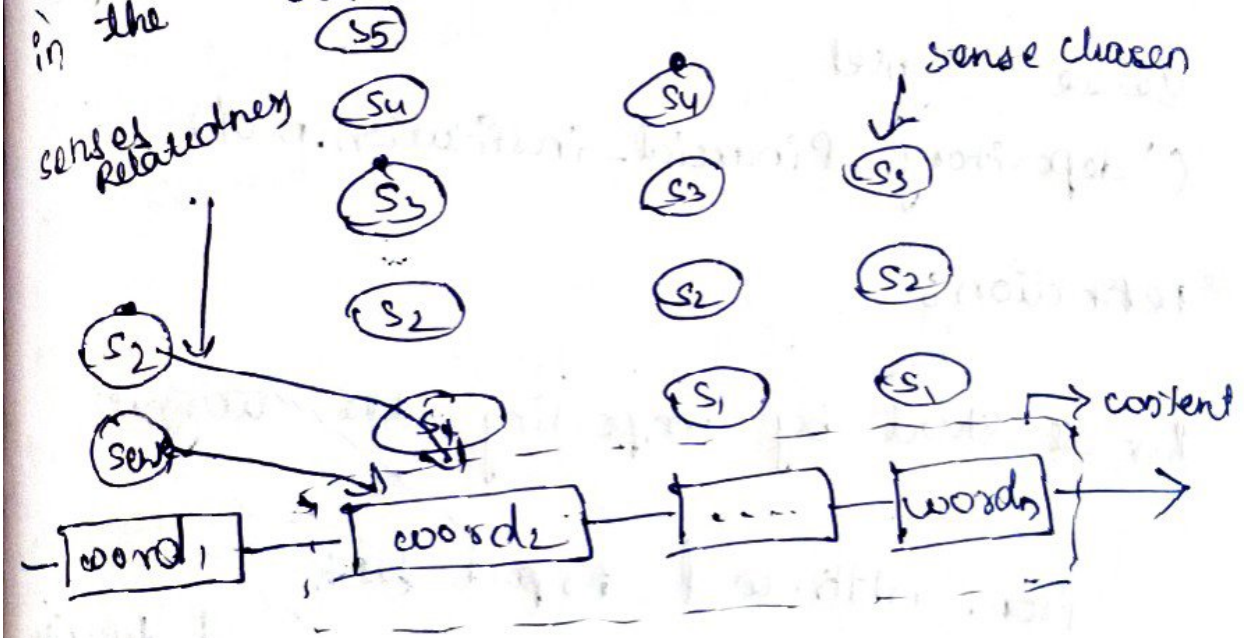
will have a similar meaning.

for ex: in the sentences below, the word "bank" ha has different meanings based on the content of the sentence.

Text 1 = 'I went to the bank to deposit my money'

Text 2 = 'The river bank was full of dead fish'

→ It is basis on the hypothesis that words m used together in text are related to each other and that the relation can be observed in the definations of words and their



Text

We First install the library "pywsd"

python implementation of wordsense disambiguation (WSD)

pip install pywsd

from pywsd.lesk import simple_lesk

sentences = ['I went to the bank to deposit my money',
'The river bank was full of dead fish']

print ("context-1:", sentences[0])

answer = simple_lesk (sentences[0], 'bank')

print ('sense:', answer)

print ('definition:', answer.definition())

out: context 1:

I went to the bank to deposit my money

sense - synset

('depository_financial_institution.n.01')

definition:

Let us start by importing the libraries.

from nltk.wsd import lesk

from nltk.tokenize import word_tokenize

ex:

a1 = lesk (word_tokenize ('this device is used to jam the signal'), 'jam')

print (a1, a1 . definition())

a2 = lesk (word. tokenize ('I am stuck in a traffic jam'), 'jam')

print (a2, a2. definition())

o/p:

synset ('jamming.n.01') deliberate radiation or reflection of electromagnetic energy for the purpose of disrupting enemy use of electronic devices or systems.

synset ('jam.v.05') get stuck and immobilized

first; it is meant to stop the correct signal.

and second implies a traffic jam.

ex2:

b1 = lesk (word_tokenize ('Apply spices to the chicken to season it'), 'season')

print (b1, b1. definition())

o/p: ('season.v.01') lend flavour to

'season' is used for cooking sense of view.

Ex: b2 = lesk (word_tokenize ('India receives a lot of train in the rainy season'), 'season')

print (b2, b2.definition())

O/p: Synset ('season.n.01') a period of the year marked by special events or activities in some field.

" geographic season.

4) c1 = lesk (word_tokenize ('The current water'), 'current')

print (c1, c1.definition())

O/p: Synset ('stream.n.02') suggestive of running water.

- c1 = lesk (word_tokenize ('The current time is 2 Am'), 'current')

print (c1, c1.definition())

O/p :- Synset ('current.a.01') occuring now or belonging to the present time

```
from nltk.corpus import wordnet
syn = wordnet.synsets('hello')[0]
print("synset name: ", syn.name())
print("In synset meaning: ", syn.definition());
print("In synset example:", syn.examples)
```

o/p synset name: hell.n.01

synset meaning: an expression of greeting

synset ex: ['every morning they exchanged polite hellos'].

word.net.synset(word) can be used to get a list of synsets. This list can be empty (if no such word is found) or can have few elements.

hypernyms and hyponyms:

Hypernyms: more abstract terms

Hyponyms: more specific terms

Both come to pickles as Sysets are organized in a structure similar to that of an inheritance tree.

supervised Rule-

Algorithm lesk :

procedure : Simplified_Lesk(word, sentence) ret

best sense of word.

1) best_sense ← most frequent sense of wo

2) max_overlap ← 0.

3) context ← set of word in sentence.

4) for all sense ∈ senses of word do.

5) signature ← set of words in gloss and examples of sense.

6) overlap ← COMPUTEOVERLAP (sign, con

7) if overlap gt max_overlap then

max_overlap ← overlap

best_sense ← sense

end if

end for

return best_sence.

Supervised : Ironically, the simplest form of word sense disambiguating systems. The supervised approach which tends to transfer all the complexity to the machine learning

classifiers: probably the most common and high-performing classifiers are support vector machines (SVMs) and maximum entropy (MaxEnt) classifiers. many good-quality, freely available distributions of each are available and can be used to train word sense disambiguation models. Typically, because each lemma has a separate sense inventory, it is almost always the case that a separate model.

features: we discuss a more commonly found subset of features that have been useful in supervised learning of word sense.

lexical context: This features comprises the words and lemmas of words occuring in the entire paragraph or a smaller window.

part of speech: This features comprises the POS information for words in the window surrounding the word that is being good.

Bag of words content: This features compri
the POS information for words in the windo
A threshold is typically tuned to include
most informative words in the larges conte

local collocations: local collections are an ex
sequence of phrases near the target word
provide semantic context for disambiguation.
-ally, a very small window of about the t
tokens on each side of the target word.

ex: He did bought a box of nails from the
where
hardware store.

here $c_{1,1}$ → bought would be word form.
from the
$c_{1,8}$ → store would be the string.

syntactic relations: If the phrase of the
sentence containing the target word availa
then we can use syntactic features.

Topic features: The broad topic, or domain,
of the article that the word belongs to

is also a good indicator of what sense of the word.

voice of the sentence: This ternary feature indicates whether the sentence in which the word occurs is apassive, semipassive, or active sentence.

presence of subject/object: This binary feature indicates whether the target word has a subject or object. Given a large amount of training data, we could also use the actual lexeme.

sentential complement: This binary features indicates whether the word has a sentential complement.

prepositional phrase adjunct: This feature indicates whether the ~~target~~ word has a sentential complement.

prepositional phrase adjunct: This feature indicates whether the target word has a prepositional phrase.

Algorithm : Rules for selecting Syntactic rele
as features.

1) if w is noun then
2) Select parent head word (h)
3) Select part of speech of h
4) select voice of h.
5) Select position of h (left, right)
6) else if w is a verb then
7) select nearest word l to the left of w su
that w is the parent head word of l.
8) select nearest word r to the right of
such that w is the parent head word of
9) Select part of speech of l.
10) Select part of speech of r.
11) select part of speech of w.
12) Select voice of w.
13) else if w is a adjective then
14) select parent head word (h)
15) Select part of speech of h.
16) end if.

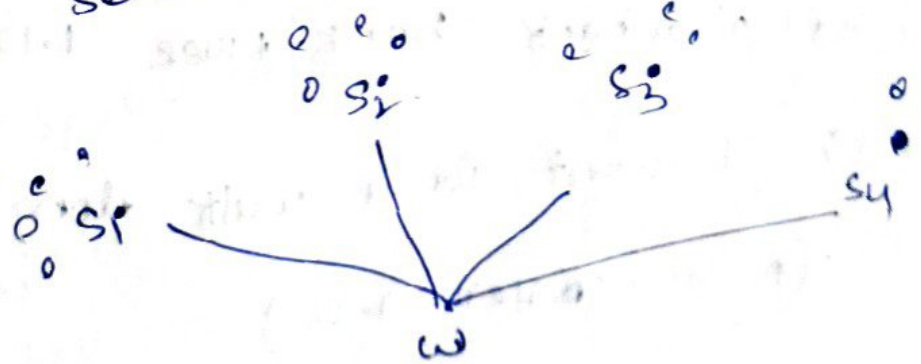
Named entity: This feature is the named entity of the proper nouns and certain types of common nouns.

wordnet: wordnet synsets of the hypernyms of head nouns of the noun phrase arguments of verbs and prepositions.

path: this feature is the path from the target verb to verb's arg.

subcategorization: The subcategorization frame is essentially the string formed by joining the verb phrase type with that of its children.

unsupervised: progress in word sense disambiguation is stymied by the dearth of labeled training data to train a classifier. For every sense of each word in a given language.



training phase

→ features are extracted from the corpus

→ An ordered decision list of the form (feature value, sense, score) is created.

→ The score of a feature f is the log-likelihood ratio of the sense given the feature as,

$$score(s_i) = \max_f \log\left(\frac{P(s_i/f)}{\sum_{j \neq i} P(s_j/f)}\right)$$

Decision lists

The decision list for the word 'bank'

feature	prediction	score
a/c with the bank	bank/fin	4.33
standing in bank	bank/fin	3.35
bank of blood	bank/supply	2.48
work in bank	bank/fin	2.33
The left river bank	bank/river	1.12

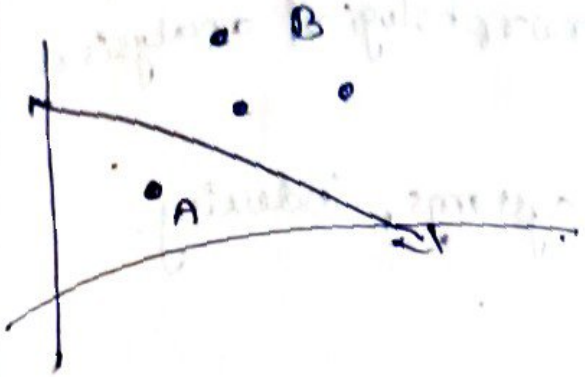
Test: I went for a walk along the river bank

(Noun - river - 1.12)

Support Vector Machines

e.g: If a word has 4 senses

sysn	A	B
1	s_1	s_2 s_3
2	s_2	s_1 s_4
3	s_3	s_1 s_4
4	s_4	s_2 s_3



The distance gives the confidence score for each sysn.

Program of WSD

Supervised:

Lemmatization: It converts plural form of words to singular and present tense to past-nce.

was → be

mice → mouse

meeting → meet

```
import spacy.cli
spacy.cli.download('en_core_web_sm')
import spacy
nlp = spacy.load('en_core_web_sm')
```

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single

item. lematization & stemming
it brings context to the words. so it links
words with similar meaning the one
lemmatization does morphological analysis of
words.

Applications: retrival systems, indexing.

ex: 'rocks' : rock
corpora : corpus
better : good

import nltk
nltk.download('wordnet')
from nltk.stem import porterstemmer
from nltk.corpus import wordnet
paragraph = " I have three visions for India

import nltk
from nltk.stem import PorterStemmer
from nltk.stem import porterstemmer, lemmatizer
porter = PorterStemmer()
porter = lemmatizer()
print ("Porter Stemmer")
print (porter.stem ("cats"))
print (Porter.stem ("trouble"))
("troubling")

stemming:

```
print (porter.stem ("troubled"));
print ("lemmatizer");
print (lematizer.stem ("cats"))
```

("trouble")
("troubling")
("troubled")

all: porter stemmer
cat
trouble
troubl
troubl
Lemmatizer stemmer
cat
troubl
troubl
troubl

porter stemmer uses suffix stripping to produce stems. It is giving the root (stem) of the word "cats" removing the 's'

Stemming: It is process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma.

PorterStemmer() It includes the implements the PorterStemming technique.

ex: we construct an instance of Porter. stemmer() and use the porter alg to stem the list of words.

→ Snowball Stemmer: This method utilizes an instance of more precise and is referred to as "English Stemmer". It is some what faster and more logical than original Porter Stemmer.

ex: nltk.stem import SnowballStemmer
Snowball = SnowballStemmer(language='english')
words = ['generous', 'generate', 'generously', 'generation']
for word in words:
print(word, "--->", Snowball.stem(word))

Op:
generous --> generous
generate --> general
generously --> generous
generation --> general.

Software: several software programs are made available by the Research Community for word sense disambiguation, ranging from similarity measure modules to full disambiguation systems. It is not possible to list all of them here so we list a selected few.

Step1: Preprocessing

- for each sense of a word w, determine the synsets of wordnet in which it appears. For each such synset, determine monosemous words included in that synset.

Step2: search

- form search phrases using the following procedures in order of preference.

1) → If they exist, extract monosemous synonyms.

2) Select each of the unambiguous parsed Constituents search phrase.

3) After passing the gloss, replace all the words with a new operator and use a query.

4) use only the head phrase combined words in the synset using AND operator

Steps: Post processing

keep only those sentence in which the word under consideration belongs to the same pos as the selected sense. and delete the others.

Predicate - Argument Structure:

Shallow Semantic parsing, or what people
now a days known today as semantic role labeling is
the process of identifying the various arguments
ments of predicates in a sentence. The linguistic
community has been debating for a few
decades over the constituents the set of arguments
and what the granularity of such arguments
label should be. for various predicates.

Resources:

Unit-IV

Predicate - Argument structure

shallow semantic Parsing, or what is popularly known today as semantic role labeling. is the process of identifying the various arguments of predicates in a sentence.

Resources:

The late 1990s saw the emergence of two important corpora that are semantically - tagged. one is Framenet and the other is - Propbank. These resources have begun a transition from a long tradition of predominantly rule-based approaches toward more data-oriented approaches. These approaches focus on tranforming linguistic insights into features, rather than into rules, and letting a machine learning framework. use these In such resources. Framenet is based on the theory of frame semantics

Framenet :

Frame net corpus is a lexical database on English that is both human- and machine-readable, based on annotating examples of how words are used in actual texts.

framenet based on a theory of meaning called Frame Semantics, deriving from the work of charles J. Fillmore and colleagues.

-goals : It is convert semantic knowledge into something that is machine readable for use in NLP systems etc.

→ A semantic frame is a structure used to define the semantic meaning of a word.

→ For ex: there are abstract frames (such as the "replacement" frame), as well as specific frame (such as the "Apply-heat" frame).

Replacement Frame

→ Pat replaced [old The curtains] [new with wooden blinds].

→ The Framenet lexicon contains one lexical unit for the association of the verb replace with the Replacement frame.

Apply - heat Frame

Boil [Food The rice] [duration for 3 minutes] [medium in water] Then drain.

Here Boil is associated with the Apply-heat Frame to form another lexical unit. Some additional possible associations with Apply - heat include : char, fry, grill, microwave, etc.

Inherited Frames

Frames is also be inherited from other frames, a good example is the transportation frame.

frame (transportation)
frame-elements (mover(s), vehicle, ...)
scene (mover(s) move along path by means)
frame (driving)
inherit (trans)
frame-elements (driver (=mover), vehicle (=means)
Rider(s) (=mover(s)), cargo (=moved(s))
scene (driver starts vehicle, drive controls
vehicle, drives, stops vehicle)
frame (RIDING)
inherit (trans)
frame-elements (Rider (=mover(s)), vehicle (=means)
scene (Rider enters vehicle,
Vehicle carries Rider along path,
Rider leaves vehicle).

Here driving and "Riding" refering the transportation frame.

propbank: From the start, propbank was developed with the idea of having a training data for machine learning-based semantic role labeling system in mind. It requires that all arguments to a verb be syntactic

Constituents to a verb be syntactic constituents and different senses of a word are only distinguished if the differences bear the arguments.

→ propbank: each verb has its own Roles

→ propbank more used, because it's layered over the treebank (and so has greater coverage plus parses)

Ex: ① sense: shift from one thing to another

Arg0: cause of shift

Arg1: thing being changed

Arg2: old thing

Arg3: new thing

② sense: move downward

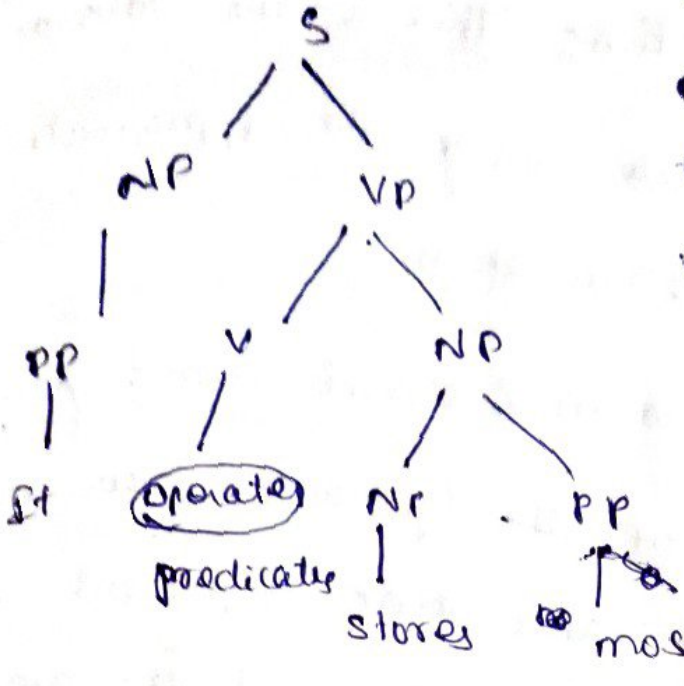
A1: thing falling

A2: extent distance fallen

A3: start point

A4: end point

Ex: It operates stores mostly in country



Argo: It
Arg1: operates
Arg2: stores
Arg3: mostly in country

[Argo It] [predicate operates] [Arg2 stores]
[Arg-loc mostly in country].

systems: WSD, little research has gone into learning predicate-argument structures from unannotated corpora, perhaps because it is closer to the actual applications and has been more or less absorbed in the area of information extractions.

Argument identification: This is the task of identifying all and only the parse constituents

that represents valid semantic arguments of a predicate.

Argument classification: This is the task of identifying all and predicate, assign the appropriate valid semantic arg label to them.

Argument identification & classification: This task is a combination of the previous two tasks where the constituents that represent arg of a predicate are identified and the appropriate arg label is assigned to them.

Syntactic Representation:

As we have seen, PropBank was created as a layer of annotation on top of Penn TreeBank style phrase structure trees and in some of the early work in recovering PropBank annotations, Gildea and Jurafsky [113] added argument labels to parses obtained from a parser trained on PennTreeBank.

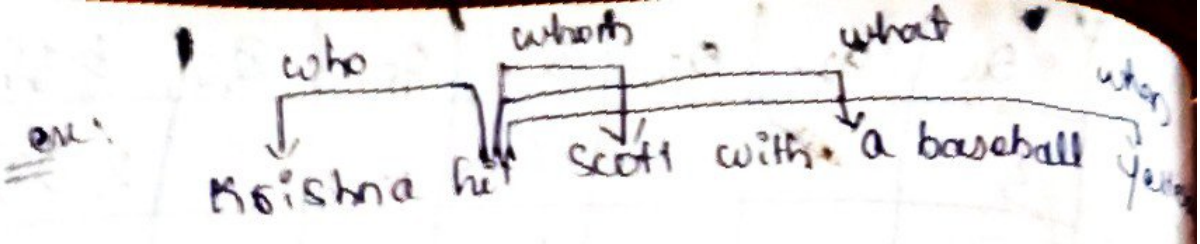
Algorithm: the semantic role labeling (SRL)

procedure: SRL (sentence) returns best semantic role labeling

Input: Syntactic parse of the sentence.

1) generate a full syntactic parse of the sentence
2) Identify all the predicates
3) for all predicate ∈ sentence do.
4) extract a set of features for each node in the tree relative to the predicate
5) classify each feature vector using the model created in training
6) select the class of highest scoring classifier
7) Return best Semantic role labeling.
8) end for.

Phrase structure Grammar (PSG)



ex:

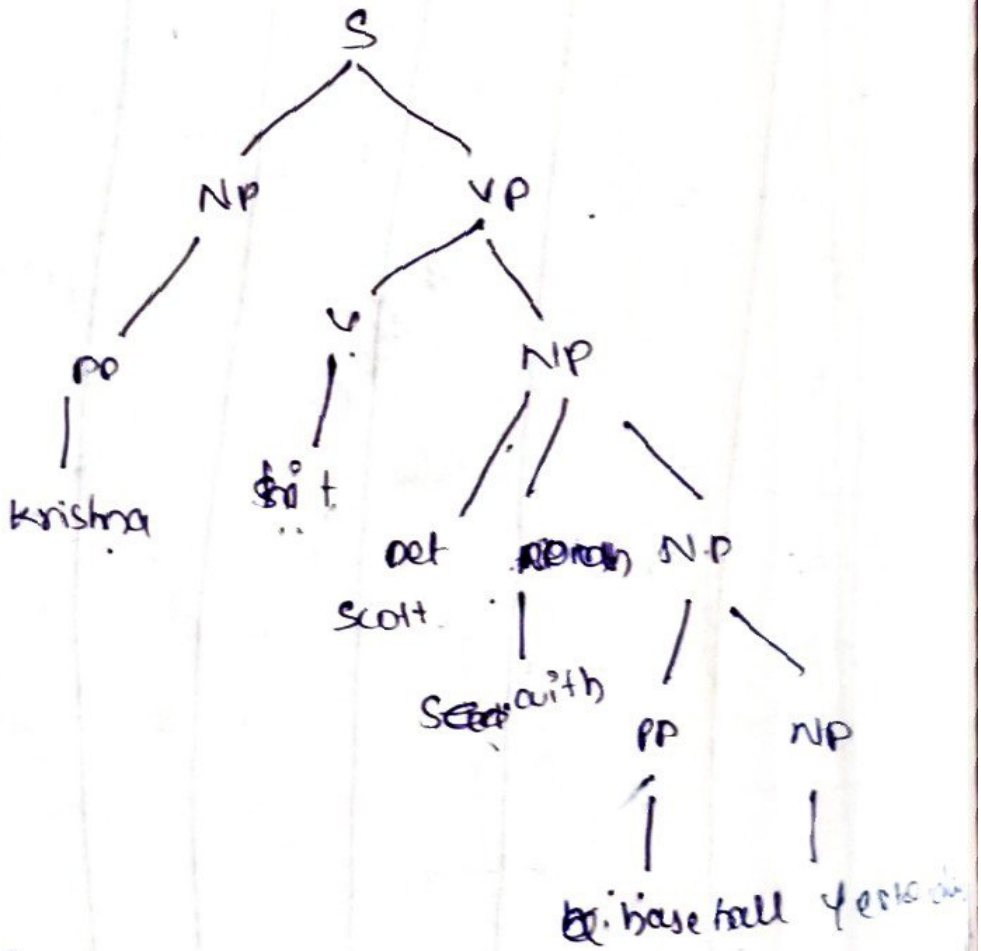
→ who hit Scott with a baseball

→ whom did krishna hit with a baseball

→ what did krishna hit Scott with

→ when did krishna hit Scott with a baseball.

Analysis :



Variations

Krishna hit Scott with a baseball

Krishna → agent, hitter

Scott → thing hit

baseball → instrument

phrase structure grammar (PSG): Framenet marks word spans in sentence to represent arg, whereas PropBank tags nodes in a tree-bank tree with arguments. Because several high-quality statical parsers existed that could produce phrase structure tree.

path: this feature is the syntactic path through the parse tree from the parse constituent to the predicate being classified.

Predicate: The identity of the predicate lemma is used as a feature.

phrase type: This feature is a Syntactic category (NP, PP, S, etc).

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.

Head word: this feature is the syntactic head of the phrase. It is calculated using a head word table.

sub categorization: This feature is the phrase structure rule expanding the predicate's parent node.

verb clustering: The predicate is one of the most salient features in predicting the arguments class. given various syntactic/semantic constructions that a predicate can appear in, any amount of hand-tagged training data.

content word: Because the head word feature of some constituents, such as PP and

restricts the development to system that operate in limited domains.

(iii) they are hard to maintain & scale up as the problem becomes more complex & more domain independent

(iv) ~~They~~ They tend to be brittle

Software:

Not a lot of software programs are available for older, more rule based system, but the following are available for download

WASP [http://www.cs.utexas.edu/~ml/wasp/]

CHILL [http://www.cs.utexas.edu/~ml/chill.html]

UNIT- IV

PREDICATE - ARGUMENT STRUCTURE

Shallow semantic Parsing, or what is popularly known today as semantic role labelling is the process of identifying the various arguments of predicates in a sentence.

Resources:

The late 1990s saw the emergence of two important corpora that are semantically tagged. One is permanent & the other is propbank. These resources have begun a transition from a long tradition of predominantly rule-based approaches toward more data-oriented approaches. These approaches focus on transforming linguistic insights into features, rather than into rules, and letting a machine learning framework. use this in such resources. Framenet is based on the theory of frame senatics.

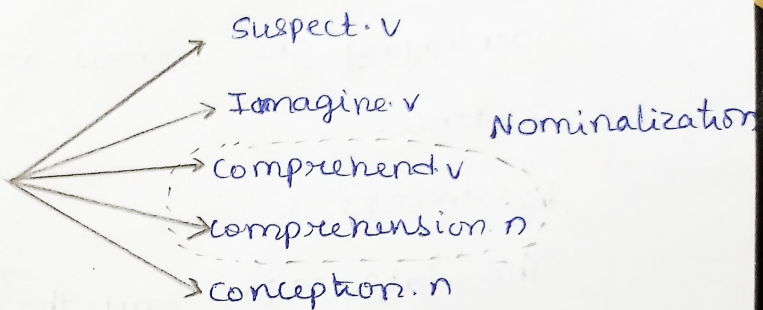
Frame Net:

Frame net corpus is a lexical database on English that is both human and machine-readable based on annulating example of how words are used in actual texts. Frame net based on a theory or meaning called Frame semantics, deriving from the work of charles *

Fill more and colleges.

FrameNet Example

Frame: Awareness	
cognizer	Manner
content	Expressor
Evidence	Role
Topic	Phenomenon
Degree	Practice



Goals:

It is convert semantic knowledge into something that is machine readable for use in NLP systems etc.

→ A semantic frame is a structure used to define the semantic meaning of a word.

→ For ex: There are abstract frames (such as the "Replacement frame), as well as specific frame (such as the "Apply-heat" frame)

Replacement Frame

⇒ pat replaced [old The curtains] [new with wooden blinds].

⇒ The Framenet lexicon contains one lexicon unit for the association of the verb place with the replacement frame.

Apply-heat frame.

Boil [food the rice] [duration for 3 minutes]

[medium in water] then train.

Here Boil is associated with the apply-heat frame to form another lexical unit. Some additional possible associations with apply-heat include: char, fry, grill, wave, etc

Inherited Frames:

Frames is also be inherited frame other frames, a good example is the transportation frame.

Ex: Frame (transportation)

frame-elements (mover(s), means, path)

Scene (mover (s) move along path by means)

frame (Driving)

inherit (Trans)

frame-elements (Driver (= mover) · vehicle (= mean),

Rider (s) (= mover (s)), cargo (= mover(s))

Scene (Driven starts vehicle, drive controls

vehicle Driver · stops · vehicle)

frame (RIDING)

Inherit (Trans)

Frame elements (Rider (= mover (s)), vehicle (= mean))

Scene (Rider enters vehicle,
vehicle carriers Rider along path,
Rider leaves vehicle)

PROPER BANK (PROP BANK):

From the start prop bank was developed with the idea provide training data to machine learning based on the semantic rules. It requires all arguments to a verb & different senses of a word distinguished by the argument.

→ In sentence each verb follows CFG rules it layered over the tree bank.

Ex: Sense: shift from one thing to another

(1)

Arg0 : consider shift

Arg1 : change (replace)

Arg 2 : old thing

Arg 3 : new thing

(2) Sense : move downward

A1 : thing falling

A2 : extent distance fallen

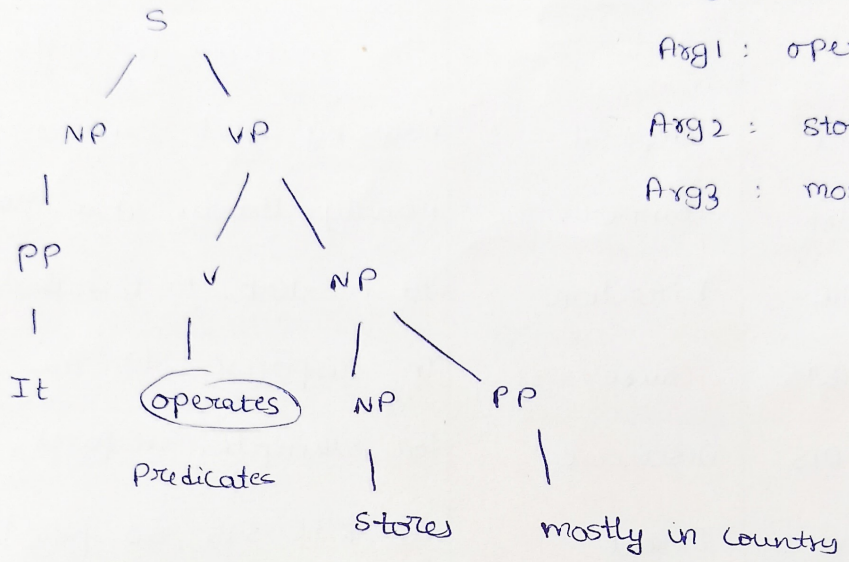
A3 : start point

A4 : end point

List of adjunctive arguments in PropBank

Tag	Description	Example
ARGM-LOC	Locative	the museum in Westborough, Mass.
ARGM-TMP	Temporal	now, by next summer
ARGM-MNR	Manner	heavily, clearly, at a rapid rate
ARGM-DIR	Direction	to market, to Bangkok
ARGM-CAU	Cause	In response to the ruling
ARGM-DIS	Discourse	for example, in part, similarly
ARGM-EXT	Extent	at $38.375, 50 points
ARGM-PRP	Purpose	to pay for the plant
ARGM-NEG	Negation	not, n't
ARGM-MOD	Modal	can, might, should, will
ARGM-REC	Reciprocals	each other
ARGM-PRD	Secondary Predication	to become a teacher
ARGM	Bare ARGM	with a police escort
ARGM-ADV	Adverbials	(none of the above)

Ex: It operates stores mostly incountry



Argo : It

Arg1 : operates

Arg2 : stores

Arg3 : mostly in country

[Argo It] [medicate operates] [$_{Arg2}$ stores]

[Arg-hoc mostly in country]

Systems:

WSD, little research has gone into learning medicate-argument structures from ohannotated corpora, perhaps because It is -closer to the actual applications and has been more or less absorbed in the area of information extractions.

Argument Identification.

This is the task of identifying all and only the pause constituents. That represents valid semantic arguments of the predicate.

Argument Classification:

This is the task of identifying all predicate, assign the appropriate value semantic arg label to them.

Argument Identification & Classification:

This task is a combination of the previous few tasks where the constituents that represent arg of a predicate are identified and the appropriate arg label is assigned to them.

Syntactic Representation:

As we have seen, propbank was created as a layer of annotation on top of Penn TreeBank style phrase structure trees and in same of the early work in recovering propBank annotations, Gildea and jwafsky [113] added argument labels to parses obtained from a parser trained on penn Tree Bank.

Algorithm: The Semantic role labeling (SRL)

procedure: SRL (sentence) returns best semantic role labeling.

Input: Syntactic parse phase of the sentence.

1) Generate a full syntactic parse of the sentence.

2) Identify all the predicates

3) for all predicate ϵ sentence do

4) Extract a set of features for each node in the tree relative to the predicate

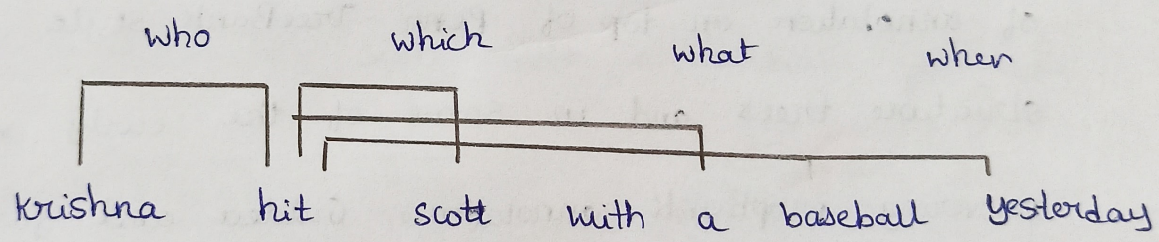
5) classify each feature vector using the model created in training

6) select the class of highest scoring classifier.

7) Return best semantic role labelling

8) end for

Ex:



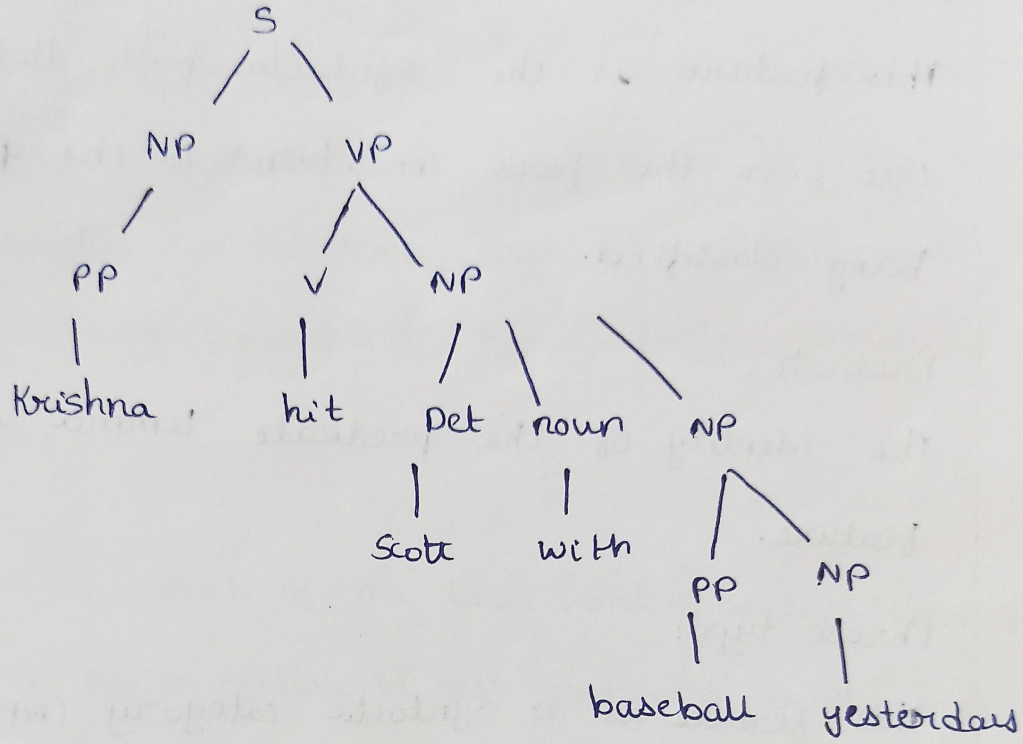
→ Who hit scott with a base ball

→ whom did Krishna hit with a baseball

→ What did Krishna hit scott with

→ When did Krishna hit scott with a baseball.

Analysis:



Variations:

Krishna hit scott with a baseball yesterday

agent, hitter	thing hit	Instrument

PHRASE STRUCTURE GRAMMAR (PSG):

Frame net marks and spans in sentence to represent org, propBank tags nodes in a tree. Bank tree with arguments because several high quality statical parsers existed that could produce phrase structure tree.

Path :

This feature is the syntactic path through the parse tree from the parse constituent to the predicate being classified.

Predicate :

The identity of the predicate lemma is used as a feature.

Phrase type:

This feature is a syntactic category (NP, PP, S, etc)

Position:

This feature is a binary feature identifying whether the phrase is before or after the medicate.

Voice :

This feature indicates whether the predicate is realised as an active or passive construction.

Head Word :

this feature is the syntactic head of the phrase - It is calculated using a head word table

Sub categorization :

This feature is the phrase structure rule expanding the medicates parent node.

Verb clustering:

The medicate is one of the most salient features in predicting the argument clans given various Syntactic. Semantic constructions that a predicate can appear in, any amounts of hand-tagged training data.

word:

Adding part of speech of the head word & content word of a constituent as a feature to help generalize in the task of arg.

H_1: if phrase type is PP then select the right most child.

Ex: Phrase = "in Texas", content word = "Texas"

H_2: if phrase type is SBAR then select the leftmost sentence ("S") clause.

Example: phrase = "that occured yesterday"

Content word = "occurred"

H_3: if phrase type is VP then

if there is a VP child

else

Select the head word

Ex:

phrase = "had placed", content word = "placed"

H_4: If phrase type is ADVP then select the right most child, not IN or TO

Ex: phrase = "more than", content word = "more"

H5: if phrase type is ADVP then select the rightmost adjective, verb, noun or ADVP

Ex: phrase 4 = "61 years old", content word = "61"

H6: for all other phrase types the head word

Ex: phrase = "red house" content word = "red"

Named entities in constituents:

Reported a performance improvement on classifying the semantic role of the constituent by using the presence of a named entity in the constituent.

verb sense information:

The arguments that a predicate can take depend on the sense of the predicate. Each predicate on the sense in which it is used. the two sense of predicates talk in Propbank corpus.

talk.01

Tag	Description
Arg 0	talker
Arg 1	Subject
Arg 2	hearer

talk.02

Tag	Description
Arg0	talker
Arg1	talked to
Arg2	Secondary action.

Clause-based path variations:

position of the clause node (S, SBAR) seems to be an important feature in argument identification. Accordingly, experimented with four clause-based path feature variations.

→ Replacing all the nodes in a path other than clause (S, SBAR) seems to be an important feature in arg identification.

→ Replacing all the nodes in a path-other than clause nodes with an (+). for Ex: the path NP ↑ S ↑ VP ↑ SBAR ↑ NP ↑ VP ↓ VBD becomes NP ↑ S ↑ VS ↑ V ↑ ↓ VBD

Path n-grams:

This feature category decomposes a path into a server of trigrams. Ex:

path NP ↑ S ↑ VP ↑ SBAR ↑ NP ↑ VP ↑ VBD becomes

NP ↑ S ↑ VP, S ↑ VP ↑ SBAR, VP ↑ SBAR ↑, NP,

SBAR ↑ NP ↑ VP, and so on. Shorter paths were padded

hills

Predicate Context:

predicate sense variations. Two words before and two words after user added as features.

Punctuations:

for some adjunctive arguments punctuations plays an importan

role. this set of features captures whether punctuations amends immediately.

Combinatory categorial Grammar (CCG):

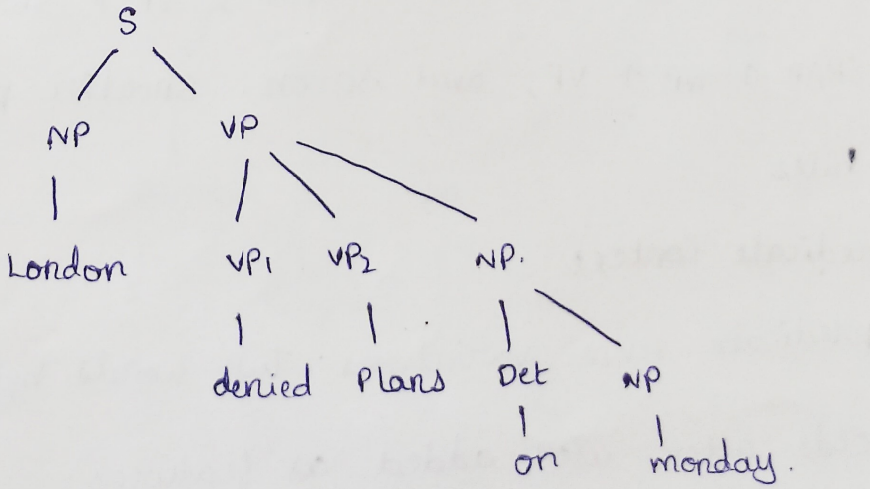
The path feature is very important from the ~~path feature is~~ argument identification task. It is one of the most sparse features and may be difficult to train or generative.

Phrase type:

This is category of the maximal projection between the two words, the medicate and the independent word.

London denied plans on monday

denied	London	(S \ NP1) NP
denied	plans	(S \ NP1 \ VP1)
on	denied	(S \ N \ VP1)
on	monday	(S \ N \ NP1)



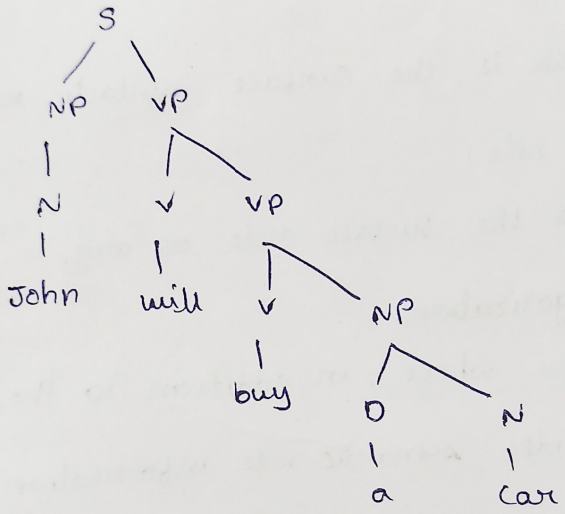
Tree adjoining grammar (TAG):

Tree generating systems

→ What are the elementary objects of CFG

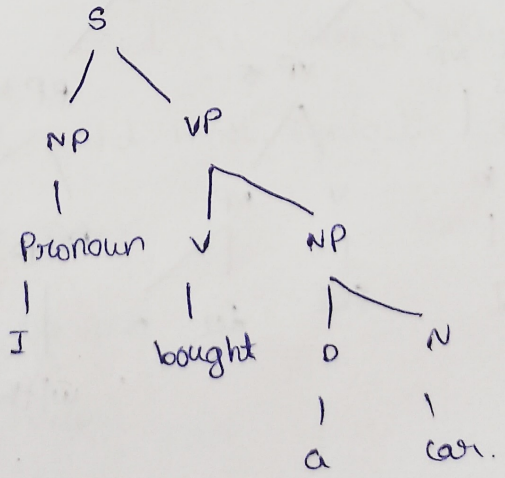
→ What are the elementary objects of TAG.

" John will buy a car"



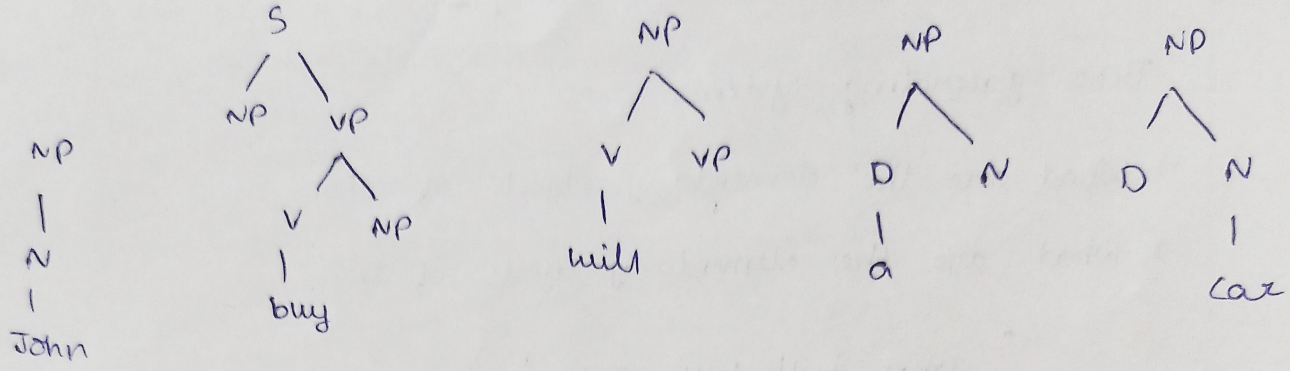
strings: S, NP, N, VP, V, VP, NP, D, N

I bought a car



S, NP, Pro, VP, V, NP, D, N.

Trees:



Super tag path:

This feature is the surface syntactic role of arg

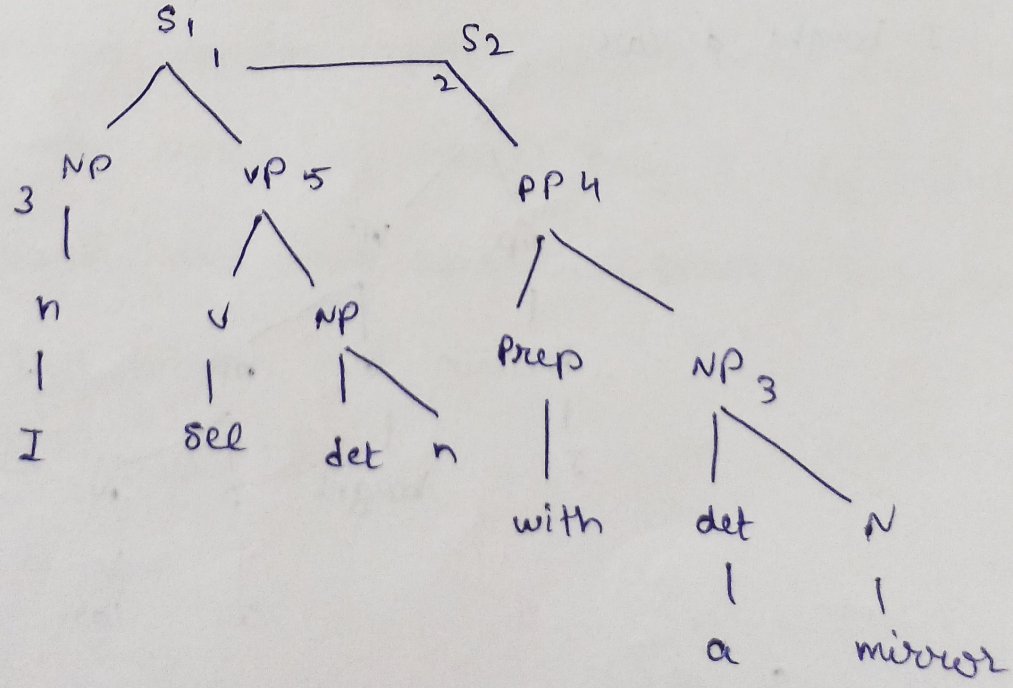
Surface Syntactic role:

This feature is the surface role of arg.

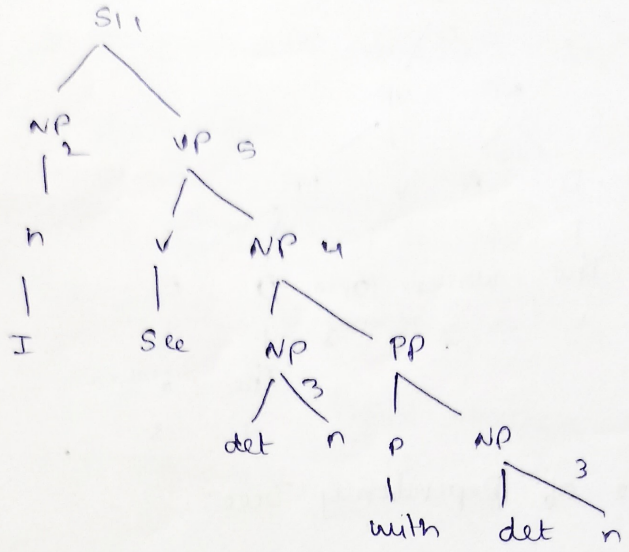
Semantic Subcategorization:

This is a frame where, in addition to the syntactic categories, the feature includes semantic role information.

"I see a man with a mirror"



(ii)



S → NP

NP → Det N2,3

VP → NP PP4

PP → P NP 5

VP → V NP

Dependency Trees:

performance of a system depends on the exact span of arg. Annoted according to the constituents in the penn tree bank. Labels are scored as correct only if they match. Because propBank and most syntactic-parsers are developed on the Penn Tree Bank.

Ex: S → NP VP

VP → V NP

V → loves

D → The

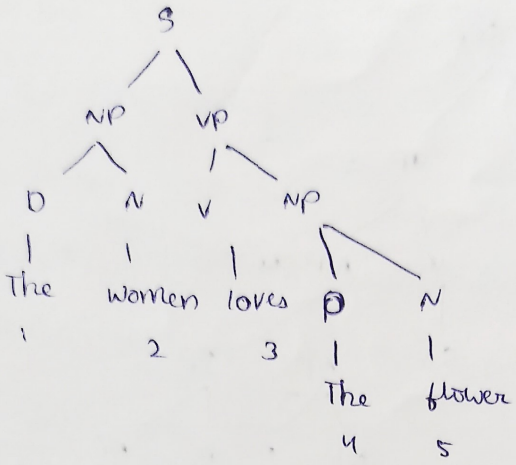
N → women | flower

S: {1,2,3,4,5}

NP: {1,2}

D: {1}

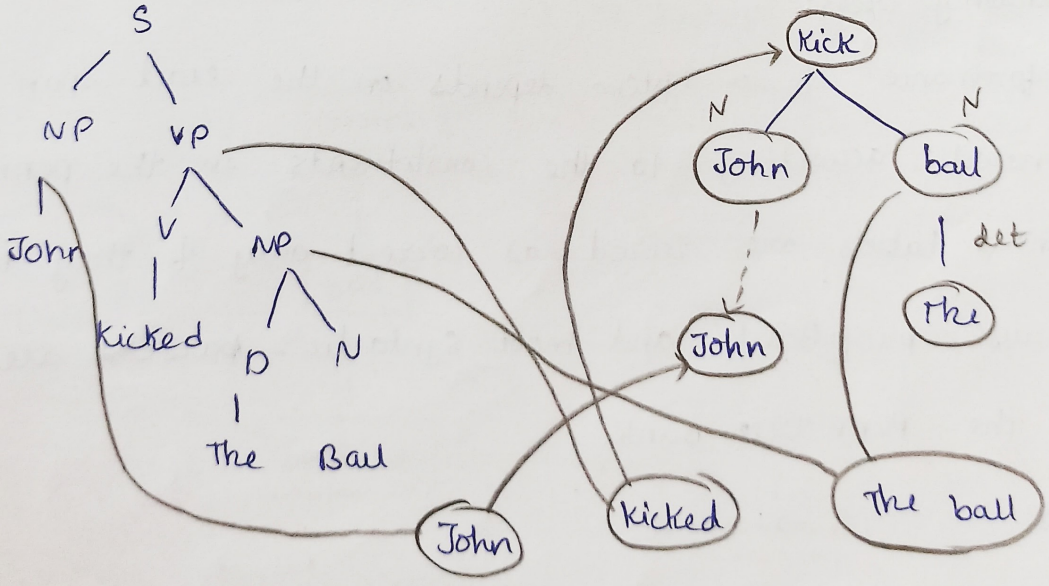
VP: {3,4,5}



V: {3}
NP {4,5}
D: {4}
N {5}

Architecture of Dependency Tree:

It is converted to Penn Tree to a dependency representation using a ~~spr~~ script and creating a dependency structure labelled with propBank arguments. It outputs dependencies between a word called head and another called modifier.



Head Word: The word representing the node in the dependency tree

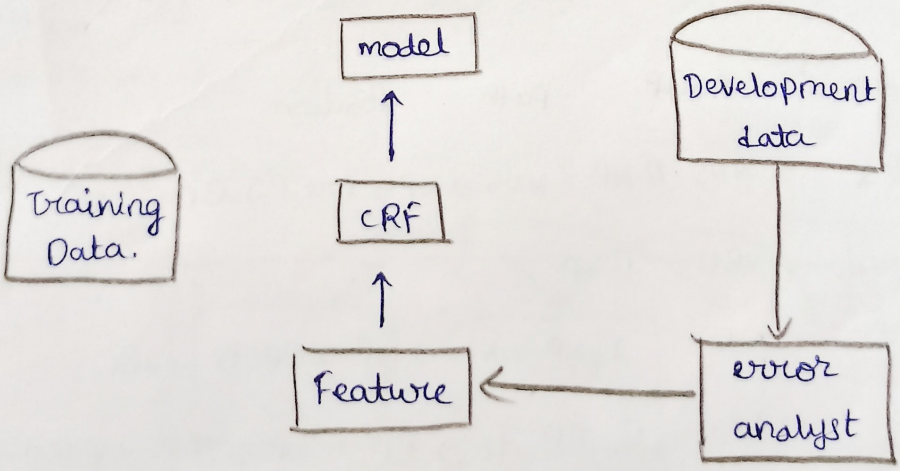
Head Word nos: pos of head word

POS path: the path from the predicate to the head word through the dependency tree.

Dependency Path: each word that is connected to the head word has a dependency relationship to the word.

Voice position: voice of the predicate whether the node is before or after the predicate.

Base phase chunks: A common question is how much does the full syntactic representation help the task of semantic role labelling. In other words, how important is it to create a full syntactic tree before classifying the arguments of a predicate? A chunk representation can be faster & more robust to phrase reordering as happens in speech-data



defines the chunks using IOB tags:

→ It specifies where the chunk begins and ends along with its types.

→ A part of speech tagger can be trained on these IOB tags to further power a chunker sub clans.

→ First using chunked_sents() method of corpus, a tree obtained and is then transformed to a format usable by a POS tagger

→ conll-tag-chunks() uses tree conll tags() to convert a sentence tree into a list of 3 tuples of the form (word, POS, iob).

Ex: B-NP, I-NP, O-NP

Begining Inside outside

* Sales declined 3% to $ 524.5 mill from $ 539.4 mill'.

⇓ chunk into syntactic basephrases

[Sales$_{NP}$] [declined$_{VP}$] [to$_{VP}$] [$_{NP}$ $524.5 m] [from] [539 mu$_{NP}$]

Phrase	Headword	POSBP	Path	Position
NP	Sales	NNS B-NP	NNS → NP → Pred → VBD	
Pred	declined	VBD B-VP	-	
NP	%	NN I-NP	NN → NP → PRED → VBD	
PP	to	TO B-PP	to → PP → NP → NP → Pred → VBD	
NP	mill	CD I-NP	CD → NP → PP → NP - Pred → VBD	
PP	from	IN B-PP	RN - PP → NP → PD → NP → Pred → VBD	
NP	mill	CD I-NP	CD → NP → DD → ND → PD → NP → Pred → VBD	

Overcoming the Independence Assumption

As mentioned earlier, various post processing stages have been proposed to overcome the limitations of treating Semantic role labelling as a series of independent argument classification.

Ex: But [nobody] [knows] [at what level] [the futures]
Arg0 P Arg1 Arg2

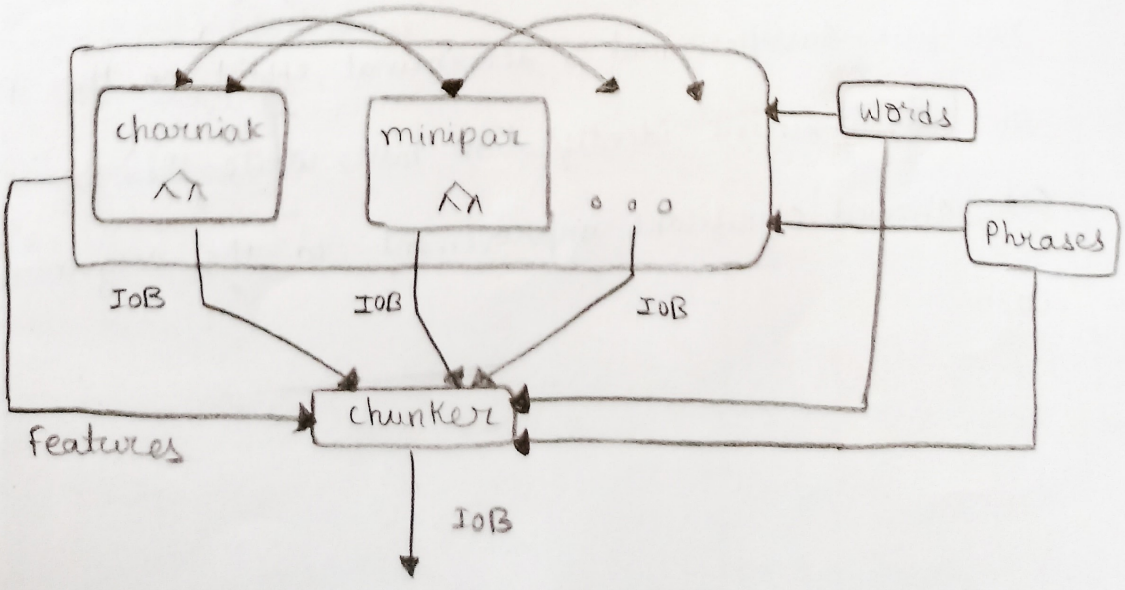
Future Performance:

Not all feature are eagerly useful in each task. Some features add more noise than information in one context than it another.

Feature Sailance

In analyzing the performance of the system, it is useful to estimate the relative constribution of the various feature sets used.

New architecture:



Overcoming Parsing Errors:

After performing a detailed error analysis, Pradhan et al. found that the identification problem poses a significant bottleneck to improving overall system performance. The base line system's accuracy on the task of labelling nodes known to represent semantic arguments is 90.1. On the other hand, the system's performance on the identification task is quite a bit lower, achieving only 80% recall with 86% precision. The two sources of these identification errors are failures by the system to identify all and only those constituents that correspond to semantic roles, when those constituents are present in the syntactic analysis and failures by the syntactic analyzer to provide the constituents that align with correct arguments.

Feature Selection:

The fact that adding the named entity features to the null/non-null classifier had a determinal effect on the performance of the argument identification task, while the same feature set showed significant improvement to the argument classification task

Table: Effect of each feature on the argument classification task & argument identification task when added to the base line system. An asterisk indicates that the improvement is statistically significant.

Features	Argument classification	Argument Identification		
	A	P	R	F1
Baseline [120]	87.9	93.7	88.9	91.3
+ Named entities	88.1	93.3	88.9	91.0
+ Head POS	88.6	94.4	90.1	92.2
+ Verb cluster	88.1	94.1	89.0	91.5
+ Partial Path	88.2	93.3	88.9	91.1
+ Verb Sense	88.1	93.7	89.5	91.5
+ Noun head PP (only POS)	88.6	94.4	90.0	92.2
+ Noun head PP (only head)	89.8	94.0	89.4	91.7
+ Noun head PP (both)	89.9	94.7	90.5	92.6
+ First POS in constituent	89.0	94.4	91.1	92.7
+ Last POS in constituent	89.4	93.8	89.4	91.6
+ Ordinal const. pos. concat	88.4	94.4	90.6	92.5
+ Const. tree distance	88.3	93.6	89.1	91.3
+ Parent constituent	87.7	93.7	89.2	91.4
+ Parent head	88.0	93.7	89.5	91.5
+ Parent head POS	87.9	94.2	90.2	92.2
+ Right sibling constituent	85.8	94.2	905	923
+ Right sibling head	88.5	94.3	90.3	91.9
+ Right sibling head POS	87.9	94.0	89.9	92.1
+ Left sibling Constituent	87.9	94.4	89.9	92.0

+ left sibling head	88.1	94.1	86.1	89.9
+ left sibling head POS	88.6	93.6	89.3	91.4
+ Temporal cue words	86.9	93.9	–	–
+ Dynamic class context	88.8	93.5	–	–

Table: Performance of various feature combinations on the task of argument classification.

Features	Accuracy
All features [120]	91.0
All except path	90.8
All except phrase type	90.8
All except HW & HW-POS	90.7
All except all phrases	83.6
All except predicate	82.4
All except HW & FW & LW info	75.1
only path & predicate	78.4
Only path & phrase type	47.2
Only Head word	37.7
Only Path	28.0

BorderSearch: Another approach is to broaden the search by selecting constituents in n-best parses or using a packed forest representation which more efficiently represents variations over much larger n. Using a parse forest shows an absolute improvement of 1.2 points over single best parses and 0.5 points over n-best parses

Example of ~~normalization~~ nominalization:

She complained about the attack
She made an official complain about the attack

John walked around the university
John took a walk around the university

identify and label the arguments of nouns. The only works that come close are the rule-based system described by Hull & Gomez and the work of Lapata on interpreting the relationship b/w the head of a nominalized compound & its modifier noun.

Robustness across Genre

One possible shortcoming of this & other approaches is that all the parsers are trained on the same Penn Treebank, when evaluated on sources other than WSJ, seems to degrade in performance. Carreras & Màrquez show that the drop in performance on test data from the Brown corpus was 10 F-score points lower than the test data from WSJ. When trained & tested on WSJ propositions the syntactic parser's performance is the main source of error, and the classification performance is quite good.

Meaning Representation:

We now turn to the third, deeper level of semantic interpre-tation whose objective is to take natural language input & transform it into an unambiguous representation that a

machine can act on. This is the form that would be more likely to be as incomprehensible to humans as it would be more likely comprehensible to machines. We can think of a parallel b/w programming languages that are much closer to the way humans manipulate information and the low-level machine code that the computer executes. Although compilers and interpreters impose various specific syntactic & semantic restrictions on a program written in a high level programming language, no such restrictions are imposed on the form that natural language can take

Resources:

A number of projects have created representations & resources that have promoted experimentation in this area.

ATIS

The Air Travel Information System project is considered one of the first concerted efforts to build systems to transform natural language into a representation that could be used by an end application to make decisions. The task involved a machine to transform a user query in spontaneous speech, using a restricted vocabulary, about flight information. It then formed a representation that was compiled into a SQL query, which was used to extract answers from a flight database.

Communicator

The communicator program was the follow- on to ATIS. while ATIS was more focused on user- initiated dialog, communicator involved a mixed- initiative dialog, whereby the human and machine had a dialog with each other with the computer ~~help~~ presenting users with real- time travel information & helping them negotiate a preferred itinerary.

Rule Based:

Semantic parsing systems that performed very well for both the ATIS & communicator projects were rule based systems in the sense that they used an interpreter whose semantic gra -mmar was handcrafted to be robust to speech recognition errors. The underlying philosophy was that the traditional syntactic explanation of a sentence is much more complex than the underlying semantic information, so parsing the meaning units in the sentence into semantic ~~p~~roved to be a better approach.

Supervised:

Although rule based techniques are relatively easy to craft in the beginning & served a good purpose to formulate solutions to various tasks, they have several downsides:

(i) they need some effor upfront to create rules

(ii) the time & specificity required to write rules usually

UNIT -V

Language model:

A language model is a probability distribution over sequence of words, Given such a sequence of length 'm', a' language model assigns a probability P to the whole sequence. Language models generate probabilities by training of text corpora in one or many languages.

Language model from the backbone of the NLP. They are a way of transforming qualitative information about text into quantitative information that machines can understand.

N-gram model:

N-gram are continous sequences of words or symbols or tokens in a document. In technical terms, they can be defined as the neighbouring sequences of item They come into play when we deal with text data in NLP.

What are n-grams:

N-grams are continous sequence of words or symbols or tokens in a document. In technical terms, they can be defined as the neighbouring sequence of items.

n-grams classifier:

'n' is just a variable that can have +ve integer values including 1, 2, 3 and so on.

→ 'n' basically refers to multiple.

n	Term
1	Unigram
2	Bigram
3	Trigram
n	n-gram

when $n=1$, it is said to be unigram

$n=2$, it is said to be a bigram

$n=3$, it is said to be a trigram

Ex: Language model:

$$P(x,y) = P(x/y)P(y)$$

$$P(x,y,z) = P(x)P(y/x)P(z/x,y)$$

$$P(\text{she}, \text{is}, \text{dead}) = P(\text{she})\, P(\text{is}/\text{she})\, P(\text{dead}/\text{she is})$$

Markov Assumption:

I wish was a unicorn

$\rightarrow P(\text{unicorn}/\text{I wish was a}) \approx P(\text{unicorn}/\text{a})$

or $P(\text{unicorn}/\text{was a})$

Ex: N-gram such as unigram

("This", "article", "is", "on", "NLP") or

bigram: ("This article', 'article is', 'is on', 'on NLP')

Now, we will establish a relation on how to find the next word in the sentence.

$P(\text{NLP} / \text{This article is on})$

equation: $P(w_5 / w_1 w_2 w_3 w_4)$ or $P(w)$

$= P(w_n / w_1 w_2 \ldots w_n)$

Calculation: $P(A/B) = \frac{P(A,B)}{P(B)}$

$P(A,B) = P(A/B)P(B)$

Corpus:

<s> Iam a human </s>

<s> Iam not a stone </s>

<s> I I live in mumbai </s>

→ check the probability of

<s> Iam not </s> using

→ (I Iam not)

$= P(I/<s>)\, P(I/I)\, P(am/I)\, P(not/am)\, P(</s>/not)$

$\doteq \frac{C(<s>/I)}{C(<s>)} \cdot \frac{C(I/I)}{C(I)} \cdot \frac{C(I/am)}{C(I)} \cdot \frac{C(am/not)}{C(am)} \cdot \frac{C(not/</s>)}{C(not)}$

$= 3/3 \cdot 1/4 \cdot 2/4 \cdot 1/2 \cdot 0/1$

$= 0$ //

→ Consider the following training data

<s> I am Jack </s>

<s> Jack I am </s>

<s> Jack I like </s>

<s> Jack I do like </s>

<s> do I like Jack </s>

Assume that we use a bigram language model based on above data and is most probable next word predicted by the model

1. <s> Jack 2. <s> Jack I do ...

3. <s> Jack I am Jack ...

4. <s> do I like ...

$$p(I/<s>) = \frac{c(<s>/I)}{c(<s>)} = 1/5$$

$$p(Jack/<s>) = c(<s>/Jack) / c(<s>) = 3/5$$

$$p(do/<s>) = c(<s>/do) / c(<s>) = 1/5$$

$$p(am/I) = c(I/am) / c(I) = 2/5$$

$$p(like/I) = c(I/like) / c(I) = 1/5$$

$$p(do/I) = c(I/do) / c(I) = 2/5$$

$$p(</s>/Jack) = \frac{c(Jack/</s>)}{c(Jack)} = 2/5$$

$$p(</s>/like) = c(like/</s>) / c(like) = 2/3$$

$$p(</s>/I am) = c(am/</s>) / c(am) = 1/2$$

$P(I/Jack) = C(Jack/I)/C(Jack) = 3/5$

$P(like/do) = C(do/like)/C(do) = 1/2$

$P(Jack/like) = C(like/Jack)/C(like) = 1/3$

$P(Jack/am) = C(am/Jack)/C(am) = 1/2$

1) Jack I
2) Jack I like
3) Jack Iam Jack--
4) <s> do I like </s>

Trigram model:

A Trigram model approximates the probability of a word given all the previous words by using only the conditional probability of the preceeding words while trigram model looks two words into the past.

Ex: <s> sun is a </s>
<s> is a kind </s>
<s> a kind soul </s>
<s> kind soul she </s>
<s> soul she will </s>

for: $P(x,y,z) = P(x)\,P(y)\,[P(z/x,y)] = P(x,y/z)$

→ $P(<s>, sun, is) = P(<s>, sun/is = 1/2 = 0.5$

$P(sun, is, a) = P(sun, is/a) = 1/3 = 0.333$

$P(is, a, </s>) = P(is, a/</s>) = 2/5 = 0.4$

$P(<s>, is, a) = P(<s>, is/a) = 1/3 = 0.333$

$P(<s>, a, kind) = P(<s>, a / kind) = 2/3 = 0.6666$

$P(a, kind, </s>) = P(a, kind / </s>) = 2/5 = 0.4$

$P(<s>, a, kind) = P(<s>, a / kind) = 1/3 = 0.3333$

$P(a, kind, soul) = P(a, kind / soul) = 2/3 = 0.666$

$P(kind, soul, </s>) = P(kind, soul / </s>) = 2/5 = 0.4$

$P(<s>, kind, soul) = P(<s>, kind / soul) = 1/3 = 0.333$

$P(kind, soul, she) = P(kind, soul / she) = 2/2 = 1$

$P(soul, she, </s>) = P(soul, she / </s>) = 2/5 = 0.4$

$P(<s>, soul, she) = P(<s>, soul / she) = 1/2 = 0.5$

$P(soul, she, will) = P(soul, she / will) = 2/1 = 2$

$P(she, will, </s>) = P(she, will / <s>) = 1/5 = 0.2$

$P = 0.5 \times 0.3333 \times 0.4 \times 0.333 \times 0.6666 \times 0.4 \times 0.333 \times 0.666 \times 0.4 \times 0.333 \times 1 \times 0.4 \times 0.5 \times 2 \times 0.5$

$P = 0.00001403821$ //

∴ Total probability $P = 0.00001403821$

Language Model Evaluation:

Before descriptive parameter estimation method further refinements of the basic n-gram modeling approach.

Let us consider the problem of gudging the performance of a language model.

Unigram

This	is	Big	Data	AI	Book

Bi-gram

This is	is Big	Big Data	Data AI	AI Book

Tri-gram

This is Big	is Big Data	Big Data AI	Data AI Book

Text	N-gram
Data	1-gram
great info	2-gram
Nice meeting you	3-gram
Nice to meet you	4-gram

.Evaluating the string:

<s> I want to eat chinese food </s>

No. of tokens = 8

No. of sentences = 1

No. of paragraphs = 0

No. of types = 1

→ Counting:

N-gram	unigram	Bigram	Trigram
Text probability	962	170	109

$$P = P(w_1, w_2, \ldots w_n)$$

$$\text{Bigram}(P) = \frac{C(w_n)}{C(w_1)}$$

$$\text{Trigram}(T) = C\left(\frac{(w_1 w_2)}{w_n}\right)$$

Parameter Estimation:

1. Maximum-Likelihood Estimation and smoothing:

So far, we have discussed estimating the mean and varrance of a distribution. our methods have been some what and have more specifically. If it is not clear how we can estimate other parameters.

now we would like to talk about a systematic way of parameter estimation.

Defination: The process of redistributing probability means search to some small non zero value is called smoothing. The most common smoothing technique is back-off. It includes splitting n-grams into those counts exceed the threshold.

Ex: <s> I am a human </s>
<s> I am not a stone </s>
<s> I I live in Lahore </s>

→ $P(\text{I am a human}) = P(I/<s>)\, P(am/I)\, P(a/am)\, P(\frac{human}{a})$

$= 3/3 \cdot 2/4 \cdot 1/2 \cdot 1/2$

$= 0.125$

→ Split vocabulary:

$P(\text{I am human}) = P(I/<s>)\, P(am/I)\, P(human/am)$

$= 3/3 \cdot 2/4 \cdot 1/2 = 0.25$

General Bigram probability: $P(x/y) = C(xy)/C(y)$

→ Bigram probability with "Laplace smoothing":

$$P(x/y) = \frac{C(xy)+1}{C(y)+V}$$

where V = distinct words

$$P(I/am) = \frac{2+1}{2+11} = 0.230$$

Trigram Training set:

denied the allegations
denied the reports
denied the claims
denied the request

→ probability using simple trigram model

Denied the _offer_ ?

→ $P(\text{offer}/\text{Denied the}) = \frac{\text{count(denied the offer)}}{\text{count(denied the)}} = 0/4 = 0$

→ probability using "Laplace smoothing or add one smoothing

$$P(\text{offer} \mid \text{denied the}) = \frac{\text{count(denied the offer)} + 1}{\text{count(denied the)} + V}$$

$$= \frac{0+1}{4+6} = 1/10 = 0.1$$

⇒ steal probability mass to generalize better

denied the allegation = 3

denied the reports = 2

denied the claim = 1

denied the request = 1

total : 7

⇒ Bigram example given table:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

→ Bigram counts for 8 of words (V=1446)

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

⇒ what is the probability of the sentence "I want to"

$$P(\text{I want to}) = P(\text{want}/\text{I}) * P(\text{want}/\text{want}) * P(\text{to}/\text{want})$$

$$= \frac{C(\text{I want})}{C(\text{I})} * \frac{C(\text{want want})}{C(\text{want})} * \frac{C(\text{want to})}{C(\text{to})}$$

$$= \frac{827}{2533} * \frac{0}{927} * \frac{608}{2417} = 0 \text{ //}$$

→ using Laplace or add 1 smoothed bigram model

$$= \frac{827+1}{2533+1446} * \frac{0+1}{927+1446} * \frac{608+1}{2417+1446} = 1$$

⇒ Table after adding 1 in all values:

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Bayesian Parameter Estimation:

Bayesian probability estimation is an alternative parameter estimation method where by the set of parameters of a model is itself viewed as a random variable

governed by a prior statistical distribution. given training samples rs:

training set:

"he is he is going abroad is going to study in the field"

→ P(is going abroad to study) = ?

→ using bi-gram model:

$$= P(\text{going}/\text{is}) * P(\text{abroad}/\text{going}) * P(\text{to}/\text{abroad}) * P(\text{study}/\text{to})$$

$$= \frac{\text{count(is going)}}{\text{count(is)}} * \frac{\text{count(going abroad)}}{\text{count(going)}} * \frac{\text{count(abroad to)}}{\text{count(abroad)}} * \frac{\text{count(to study)}}{\text{count(to)}}$$

$$= 2/4 * 1/2 * 0/1 * 1/1 = 0 \quad //$$

→ using Laplace smoothing

Distinct vocabulary = V = 9

→ P(is going abroad to study)

$$= \left(\frac{2+1}{4+9}\right) * \left(\frac{1+1}{2+9}\right) * \left(\frac{0+1}{1+9}\right) * \left(\frac{1+1}{1+9}\right)$$

$$= 3/13 * 2/11 * 1/10 * 2/10 = 0.000839$$

→ using Bayesian:

"he is he is he is going abroad is going to study in the field"

→ P(is going abroad to study) = ?

total distinct words = V = 9

(is going, goingabroad, abroad to, to study)

pairs of distinct values = 4//

pairs of distinct value = 9

→ using Laplace smoothing add v=4

$= \left(\frac{2+1}{4+4}\right) * \left(\frac{1+1}{2+4}\right) * \left(\frac{0+1}{1+4}\right) * \left(\frac{1+1}{1+4}\right)$

$= 3/8 * 2/6 * 1/5 * 2/5 = \frac{12}{1200} = 0.01$ //

=

Large-scale language model:

These language models (LSM) estimates the relative likelihood of different phrases and are useful in many different nlp applications.

Ex: They have been used in twitter Bots for robot accounts to for own sentences.

• Defining Language models: The goal of probabilistic language modeling is to calculate the prob of a sentence of sequence of words.

$$P(w) = p(w_1 w_2 w_3 \ldots w_n)$$

→ And can be to find the probability of the next word in the sequence.

$$P(w_5 / w_1 w_2 w_3 w_4)$$

Conditional probability:

Let A, B be two events with $P(B) \neq 0$

conditional probability of A given B is:

$$P(A/B) = \frac{P(A,B)}{P(B)}$$

Chain Rule:

In general cases, the formula is as follows:

$$P(x_1, x_2, \ldots x_n) = P(x_1)\,P(x_2/x_1)\ldots P\left(x_n / {}^{x_{n-1}}_{x_{n-1}}\right)$$

Ex: P("this water is so transparent")

→ $P(\text{This}) * P\left(\frac{\text{water}}{\text{this}}\right) * P\left(\text{is}/\text{this water}\right) *$

$$P\left(\frac{\text{so}}{\text{this water is}}\right) * P\left(\frac{\text{transparent}}{\text{this water is so}}\right)$$

Markov Property:

A stochastic process has a Markov Property if the conditional probability distribution of future states of the process depends only on the present state, not on sequence of events that proceeded it. In other words probability of next word can be estimated given only the previous k no. of words.

→ if k=1;

$P(\text{transparent}/\text{this water is so}) \approx P(\text{transparent}/\text{so})$

or if k=2;

$P(\text{transparent}/\text{this water is so}) \approx P(\text{transparent}/\text{is so})$

General eqn for all the Markov Assumption

if k=1; then $P(w_1/w_1 w_2 \ldots w_{1-1})$

Types of language Models:

1) Class-based language model:

Class-based N-gram model represent a common meaning for improving the robustness of language modeling by providing adequate representation of words or sequences using Markov assumption not existing or under represented in training or corpus.

$$P(w_i/w_{i-1}) = P(w_i/c_i)P(w_i/c_{i-1})$$

Ex: P(I saw a cat on a mat)

⇒ P(saw/I)

P(a/I saw)

P(cat/I saw a)

P(on/I saw a cat)

P(a/I saw a cat on)

P(mat/I saw a cat on a)

After class-based model eliminate or split before word in the denominator in the bigram

P(I saw a cat on a mat)

P(I) → P(I)

P(saw/I) → P(saw/I)

P(a/I saw) → P(a/I saw)

P(cat/I saw a) → P(cat/saw a)

P(on/I saw a cat) → P(on/a cat)

P(a/I saw a cat on) → P(a/cat on)

P(mat / # sdw of cat on a) → P(mat / on a)

Estimating probability using sentence simplify bigram

using class-based model:

<s> I am sam </s>

<s> sam I am </s>

<s> I fly </s>

→ distinct words : I, am, sam, I, fly

Bigram : <s> and </s> are also tokens.

There are 6 (v+2) tokens and 6×6 = 36 bigrams.

→ $P(I/<s>) = 2/3$ $P(\frac{sam}{<s>}) = 1/3$ $P(\frac{am}{<s>}) =$ $P(fly/<s>) = 0$ $P(\frac{<s>}{<s>}) = 0$

$P(</s>/<s>) = 0$

$P(I/I) = 0$ $P(\frac{sam}{I}) =$ $P(\frac{am}{I}) =$ $P(\frac{fly}{I}) =$ $P(\frac{<s>}{<s> I}) =$ $P(\frac{</s>}{I}) =$

$P(I/am) = 0$ $P(\frac{sam}{am}) =$ $P(\frac{am}{am}) =$ $P(\frac{fly}{am}) =$ $P(\frac{<s>}{am}) =$ $P(\frac{</s>}{am}) =$

$P(I/sam) = 1/2$ $P(\frac{sam}{sam}) = 0$ $P(\frac{am}{sam}) = 0$ $P(\frac{fly}{sam}) =$ $P(\frac{<s>}{sam}) =$ $P(\frac{</s>}{sam})$

$P(I/fly) = 0$ $P(\frac{sam}{fly}) =$ $P(\frac{am}{fly}) =$ $P(\frac{fly}{fly}) = 0$ $P(\frac{<s>}{fly}) =$ $P(\frac{</s>}{fly}) =$

$P(I/</s>) =$ $P(\frac{sam}{</s>}) =$ $P(\frac{am}{</s>}) =$ $P(\frac{fly}{</s>}) =$ $P(\frac{<s>}{</s>}) =$ $P(\frac{</s>}{</s>}) =$

Trigram are 6*6*6 = 216

→ Compute the probability of sentence.

I want English food

P(I want English food)

② Variable-length language model:

Since probabilities are less than or equal to 1, the more probabilities we multiply together, the smaller the product becomes. multiply enough n-grams together result in numerical underflow.

By using log probabilities instead of raw probabilities, we get numbers that are not as small.

Adding in log space is equivalent to multiplying in linear space, so we combine log.

→ adding is faster than multiplying the result of doing all computing and storage in log space is that, we only need to convert back into prob if we need to report them at the end.

$$\log(P_1 P_2 P_3 P_4) = \log P_1 + \log P_2 + \log P_3 + \log P_4$$

$$P_1 * P_2 * P_3 * P_4 = \exp(\log P_1 + \log P_2 + \log P_3 + \log P_4)$$

③ Discriminative language model:

Standard n-gram models is a general model for assigning a probability to a given word sequence 'w'. However, in practical applications like machine translation or speech recognition, the task of a language model is often to separate good sentence hypotheses from bad sentence hypotheses. such that word strings of wildly differing quality receive maximally.

Distinct probability estimates Recent attempts at such discriminative language model.

Ex:

Given data:

Jane saw will

Mary saw Jane

→ Collect data with POS tagging

Jane saw will Mary saw Jane
N V N N V N

Create lookup table:

	N	V
Mary	1	0
saw	0	2
Jane	2	0
will	1	0

→ three training data set:

Mary will see Jane
N M V N

will will see mary
N M V N

Jane will see will
N M V N

Create lookup table:

	N	V	M
Mary	2	0	0
will	2	0	3
see	0	3	0
Jane	2	0	0

Bigram: N-followed by M

M followed by V

V followed by N

• Bigram lookup table:

	N-M	M-V	V-N
Mary - will	1	0	0
will - see	0	3	0
see - Jane	0	0	1
will - will	1	0	0
see - marry	0	0	1
Jane - will	1	0	0
see - will	0	0	1

→ let us an example proposed and appropriate tag sequence for a sentence.

"John can see will
↓ ↓ ↓ ↓
N model verb Noun

first noun word after comes modal and a modal after a verb, verb after noun.

→ "Tied will spot will"
↓ ↓ ↓ ↓
N m v N

tagged noun, modal, verb, noun. These probabilities are emission prob.

Ex②: mary Jane can see will
N N M V N

spot will see mary
N M V N

will Jane spot mary
M N V N

mary will pat sat
N M V N

• Create lookup table

	N	V	M
Mary	4	0	0
Jane	2	0	0
can	0	0	1
see	0	2	0
will	1	0	3
spot	1	1	0
pat	0	1	0
sat	1	0	0

→ Identify Noun, Model (M), verb (V). Lets divide each word appeared by the total no. of every pos in the set of sentence.

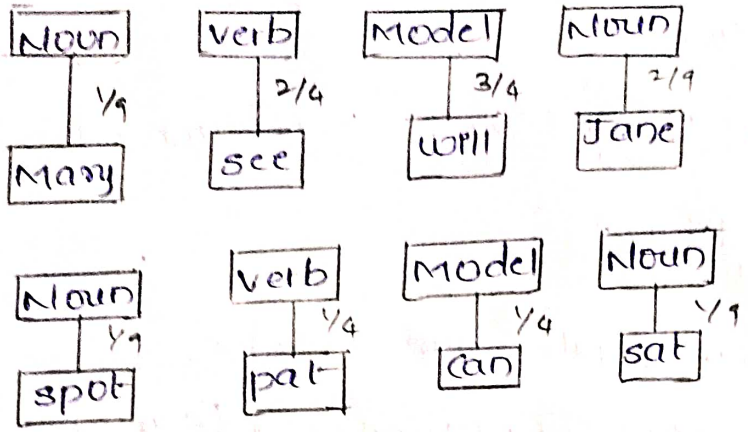
→ No. of N=9, No. of V = 4, No. of M=4

	N	V	M
Mary	4/9	0/4	0
Jane	2/9	0	0
can	0	0	1/4
see	0	2/4	0
will	1/9	0	3/4
spot	1/9	1/4	0
pat	0	1/4	0
sat	1/9	0	0

→ Mary: N = 4/9, V=0, M=0

Jane: N = 2/9, V=0, M=0

!



→ total probability = $4/9 \cdot 2/4 \cdot 3/4 \cdot 2/9 \cdot 1/9 \cdot 1/4 \cdot 1/4 \cdot 1/9$

$= 0.00694$

④ Syntax-Based Language Model:

A well known drawback of n-gram language model is that they take account of similar words in the history. That fall outside the limited window of the directly preceeding (n-1) words. However Natural language exhibits many types of long distance dependency the choice of the current word is dependent on the words that are relatively removed in terms of sequence position.

Our goal is to compute the probability of sentence or sequence of words $w = (w_1, w_2, \ldots, w_n)$

$$P(w) = P(w_1, w_2 \ldots w_n)$$

$$= P(w_n / w_1 w_2 w_3 \ldots w_n)$$

→ And compute probability of a sequence and computing the Conditional probability of a word given previous words

Conditional probability: $P(B/A) = \frac{P(A,B)}{P(A)}$

→ $P(A,B) = P(A)P(B/A)$

→ $P(A,B,C,D) = P(A)P(B/A)P(C/A,B)P(D/A,B,C)$

Unigram: $P(w_1^n) \simeq \prod_{k=1}^{n} P(w_k)$

Ex: $P(\langle s \rangle$ the man from Jupiter came $\langle /s \rangle) \simeq$

$P(\text{the})\,P(\text{man})\,P(\text{from})\,P(\text{Jupiter})\,P(\text{came})$

Bigram: $P(w_1^n) \simeq \prod_{k=1}^{n} P(w_k / w_{k-1})$

Ex: $P(\langle s \rangle$ the man from jupiter came $\langle /s \rangle) \simeq$

$P(\text{the}/\langle s \rangle)\,P(\text{man}/\text{the})\,P(\text{from}/\text{man})\,P\left(\frac{\text{Jupiter}}{\text{from}}\right)P\left(\frac{\text{came}}{\text{jupiter}}\right)$

$\cdot\, P(\langle /s \rangle / \text{jupi came})$

Trigram: $P(w_1^n) \simeq \prod_{k=1}^{n} P(w_k / w_{k-1}\, w_{k-2})$

Ex: $P(\langle s \rangle$ the man from jupiter came $\langle /s \rangle) \simeq$

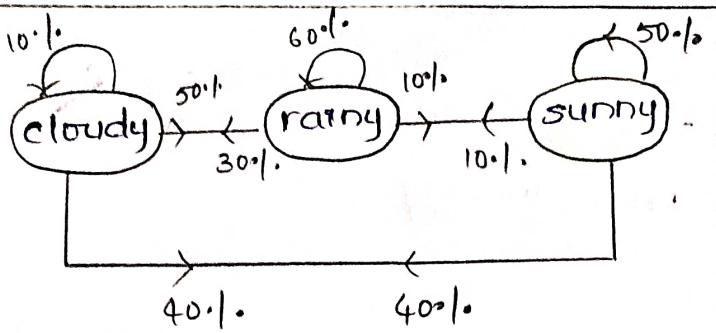
$= P\left(\frac{\text{man}}{\langle s \rangle\ \text{the}}\right) P\left(\frac{\text{from}}{\text{the man}}\right) P\left(\frac{\text{jupiter}}{\text{man from}}\right) P\left(\frac{\text{came}}{\text{from jupiter}}\right)$

$P\left(\frac{\langle /s \rangle}{\text{jupiter came}}\right)$

⑤ Max Ent language model:

This model combined with Max-likelihood-based prob estimation for language model is that the constraints imposed by estimates solely derived from the training data are too strong.

Ex: The diagram below represents three states representing weather of the day. (cloudy, rainy, and sunny).



→ There are different states such as cloudy, rainy, and sunny. following represents the transition probabilities based on above diagram.

- If sunny today, then tomorrow:

  p(sunny) = 50·1·
  p(rainy) = 10·1·
  p(cloudy) = 40·1·

- If rainy today, then tomorrow:

  p(rainy) = 60·1·
  p(sunny) = 10·1·
  p(cloudy) = 30·1·

- If cloudy today, then tomorrow:

  p(cloudy) = 10·1·
  p(rainy) = 50·1·
  p(sunny) = 40·1·

→ cloudy(wednesday) sunny(tuesday)

probability to cloudy(wednesday)

→ sunny - sunny(Tuesday) - cloudy(wed)

the prob to a cloudy on wednesday

$= 0.5 * 0.4 = 0.2$

→ sunny - rainy (tuesday) - cloudy (wed)

the prob to a cloudy on wednesday

$= 0.1 * 0.3 = 0.03$

→ sunny - cloudy (tuesday) - cloudy (wed)

the prob to a cloudy on wednesday

$\Rightarrow 0.1 * 0.4 = 0.04$

→ total prob to a cloudy on wednesday $= 0.2 + 0.03 + 0.04 = 0.27$

⑥ Factored language models:

The factored language model (FLM) approach [62, 63] builds on the observations that the prediction of words is dependent on the surface from form of the preceeding words and that better generalizations can be made when additional information, such as the word's pos or morphological class, is taken into account. In particular, word n-grain counts might be insufficient to robustly estimate the probability of word w_i given word w_{i-1}, but if we know that word w_{i-1} is of a particular class, say a determiner, we might be able to obtain a good probability estimate for $p(w_i / \text{determiner})$.

This is reminiscent of the class-based models described previously; however, in an FLM, many such class-based estimates are combined and structured hierarchically through the use of a generalized backoff strategy.

→ FLMs assume a factored word representation, where words are considered feature vectors rather than individual surface forms; that is, $w \equiv f_{1:k}$.

→ It is an extention of conventional language model.

→ The probabilistic model provides where the prediction of a factor 'f' is based on 'n' parents ($f_1 \rightarrow f_n$).

→ If 'w' represents a word and 't' represents POS tag expression is $P(w_i / w_{i-1}\ w_{i-2} \ldots w_{i-n})$.

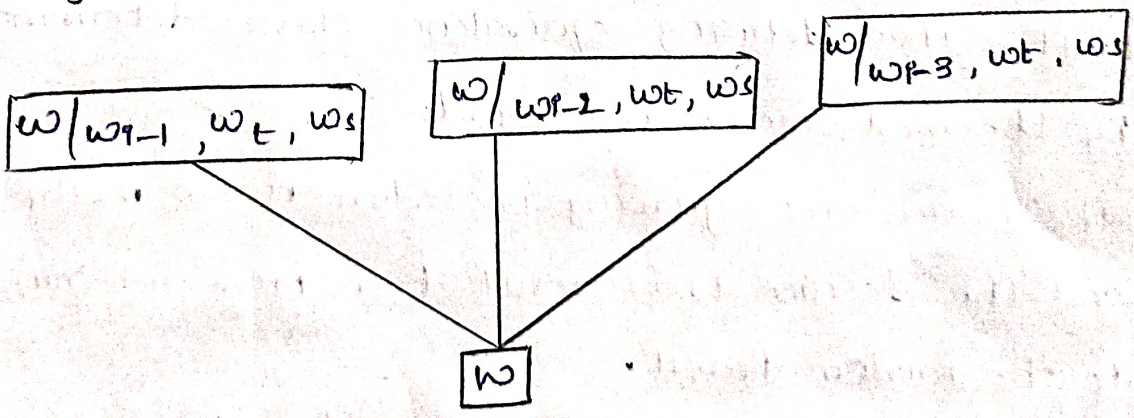
→ FLM used to allow stemming the word and provides pos tagging like morphological information.

Ex: word : Stock prices are rising

stem : stock price is rise

pos tag: N N.pluial v.plural v part.

Bigram:



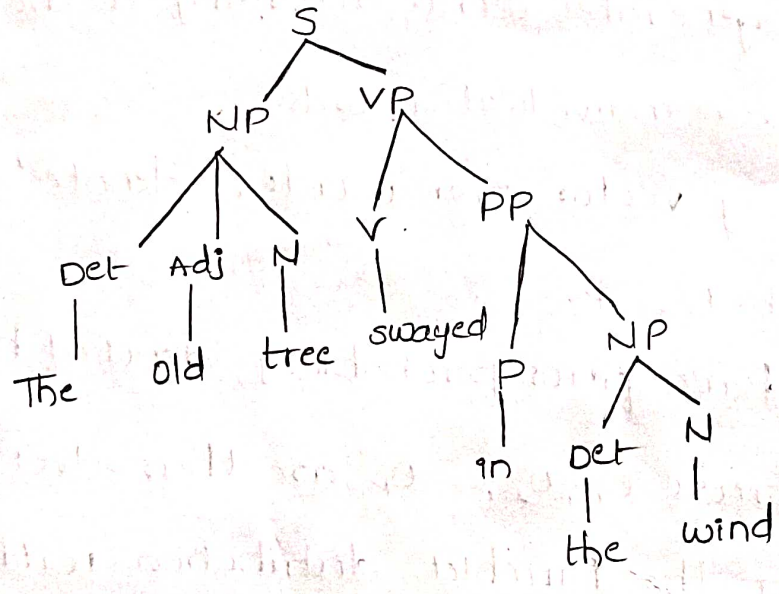
⊕ Other Tree-Based Language Models:

These model can be used for regular expression using CFG rules to generate prediction applying pos tagging to classify predicting categorization values.

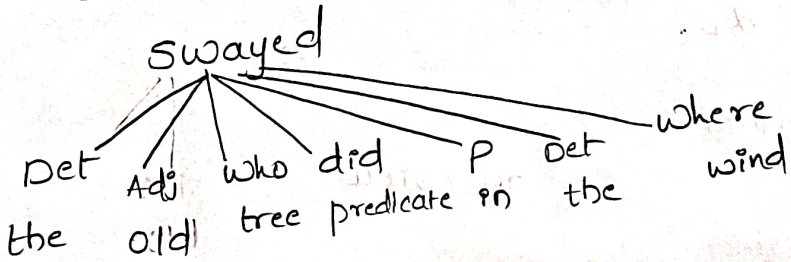
- Random forest language (RFLMs) Models, (produced) proposed by Xu and Jelinek, model all word histories seen in the training data as a collection of randomly grown decision trees (a random forest).
- Nodes in the decision trees are associated with sets of histories, with the root node containing all histories.
- Trees are grown by dividing the set of histories into two subsets according to the word identity at a particular position in the history.
- Out of a number of possible splits chosen to undergo the log-likelihood test is selected randomly as well.
- After the growing process has stopped, each leaf node in a decision tree can be viewed as a cluster of similar word histories forming an equivalence class.
- Rather than defining equivalence classes deterministically by the most recent n-1 words.
- The decision tree-growing procedure is run multiple times, and the decision trees result been from each run are added to the random forest.
- Supposing that M decision trees have been obtained, the RFLM probabilities are computed as averages of the individual decision tree probabilities.

Ex: The old tree swayed in the wind

who — predicate + did — where.



dependency tree:



(8) Bayesian Topic-Based Language Models:

- A significant recent trend in statistical language modeling is Bayesian modeling of latent topic structure in documents.
- It is a probabilistic algorithm based on applying bayes theorem with Naives assumption of conditional independent between every pair of failure. Given example training data
- One of the first models of this type was the latent Dirichlet allocation model proposed by Blei, Ng, & Jordan.

- The LDA model assumes that a document is composed of k topics, denoted as $z_1, \ldots, z_k$.
- Each topic generates words according to the topic-specific distribution over individual words.
- The probability vector over words is denoted by ϕ_k for each topic $k=1, \ldots, k$.
- Each topic has a prior probability, denoted by θ_k.
- the topic priors $\theta_1, \theta_2, \ldots \theta_k$ are themselves distributed according to the Dirichlet distribution with hyper-parameters $\alpha_1, \ldots \alpha_k$.

Ex: Given training data

Text	Review
I liked the movie	positive
It's a good movies nice story	positive
Nice songs but sadly boring ending	Negative
Hero acting is bad overall nice movie heroine acting is good	positive
Sad boring movie	Negative

→ Classify the text and calculate the probability of p("overall liked the movie") is.

$P(+ve) = 3/5$, $P(-ve) = 2/5$

→ P(overall liked the movie) = P(overall) P(liked) P(the) P(movie)

→ Calculate P(overall liked the move / +ve) =

$$P\left(\frac{\text{overall}}{+ve}\right) P\left(\frac{\text{liked}}{+ve}\right) P(\text{the}/+ve)\, P(\text{movie}/+ve)$$

$$= 1/3 \cdot 1/3 \cdot 1/3 \cdot 4/3 = 0.04938 \text{ //}$$

→ Calculate P(overall liked the move / -ve) =

$$P\left(\frac{\text{overall}}{-ve}\right) P\left(\frac{\text{liked}}{-ve}\right) P(\text{the}/-ve)\, P\left(\frac{\text{movie}}{-ve}\right)$$

$$= 1/2 \cdot 1/2 \cdot 1/2 \cdot 4/2 = 1/4 = 0.25 \text{ //}$$

⑨ Neural Network language Models:

Neural Network language models (NNLMs) take a different approach: discrete word sequences are first mapped into a continous representation, and n-gram probabilities are then estimated within this continous space.

→ The assumption is that words having similar distributional properties will receive similar continous representations, which will in turn result in smoother probability estimates.

→ The Neural Network is typically a multilayer perceptron with an input layer, projection layer, a hidden layer, and an output layer of nodes.

→ A graphical representation of the architecture of an NNLM is shown below:

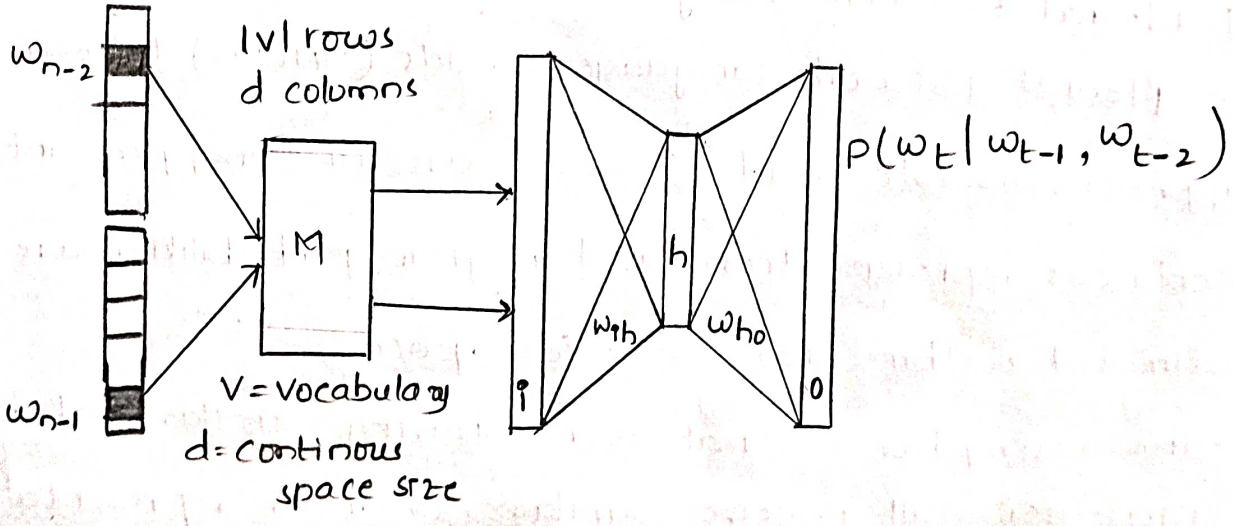
→ Adjacent layers are fully interconnected by weights

→ For a vocabulary of V words, the input consists of a concatenation of n-1 V-dimentional binary feature vectors representing the history of n-1 words (eg: The first two words in a trigram).

→ The projection layer 'i' of a given fixed dimensionality d encodes the shared continous representation of the words, which is learned during training.

→ The hidden layer 'h' also has a fixed no. of J nodes.

$$h_j = \tanh\left(\sum_{k=1}^{d} w_{jk}^{h} i_k + b_j^{h}\right) \quad \forall j, 1 = 1, \ldots, J$$



Eg: NNLM

Language - Specific Modeling problems:

The majority of language modeling research has focused on the English language.

→ However speech and language processing technology has been ported to a range of other language.

→ some of which have highlighted problems with the standard n-gram modeling approach and have necessitated modifications to the traditional language modeling framework. In this section, we look at three types of language-specific problems: morphological complexity, lack of word segmentation and spoken versus written languages.

(1) Language Modeling for Morphologically Rich languages:

→ A morphologically rich language is characterized by a larger number of different—unique word forms in relation to the no. of word tokens in a text that is due to productive morphological processes in the language.

→ A morpheme is the smallest meaning-bearing unit in a language.

→ Morphemes can be either free (i.e, they can occur on their own) or they are bound.

→ Morphological processes include compounding (forming a new word out of 2-independently existing free morphemes), derivation (combination of a free and a bound morpheme to signal a particular grammatical feature).

→ Many languages have rich inflectional paradigms.

→ Morphological complexity causes problems for language modeling due to the high ratio of types of tokens, which

results in a lack of training data: many n-grams in the test data are not seen at all in the training data or are not observed frequently enough, leading to unreliable probability estimates.

Table: MSA inflectional paradigms for present tense verb forms and possessive pronouns (affixes are separated from the word stem by hyphens)

Word	Meaning	Word	Meaning
'a-skun(u)	I live	kitaab-iy	my book
ta-skun(u)	you(MASC) live	kitaabu-ka	your(MASC) book
ta-skun-iyna	you(FEM) live	kitaabu-ki	your (FEM) book
ya-skun(u)	he lives	kitaabu-hu	hrs book
ta-skun(u)	she lives	kitaabu-haa	her book
na-skun(u)	we live	kitaabu-na	our book
ta-skun-uwna	you(MASC-PL) live	kitaabu-kum	your book
ya-skun-uwna	they live	kitaabu-hum	their book

(ii) Selection of subword units:

→ The identification of subword units can be performed in a data-driven, unsupervised way

→ It can be based on linguistic information (e.g. morphological analyzer), or it can be based on combination of both.

→ Linguistically based methods typically involve a handcrafted morphological analyzer tool, such as the Buckwalter Morpholo-gical Analyzer, which analyzes each word form into its

morphological components.

→ Data-driven approaches may incorporate varying degrees of information about the specific language.

→ Some approaches are intended to discover units corresponding to linguistically defined morphemes.

→ Automatic algorithms for identifying linguistic morphemes are as old as zelling Harris's approach of estimating the perplexity of different letters.

→ Morfessor package is one example. It attempts to derive a morpheme inventory 'M' from a corpus 'c' by maximizing its posterior probability:

$$M = \arg\max_{M} P(M/c) = P(c/M)P(M)$$

(iii) Modeling with Morphological Categories:

→ Most of the work on language modeling with subword units has focused on linear decomposition of words, primarily for agglutinative languages.

→ As a result subword units are most frequently used in a standard n-gram model.

→ One problem is that the n-gram context needs to be increased in order to be able to model interword dependencies in addition to dependencies blw subword units.

→ This in turn places demands on the amount of training data needed.

→ The language model assigns probabilities to sequences of entire words, but does so by taking into account the constraints proved by morpheme-based feature functions.

→ This model was shown to provide slightly better results (0.3% absolute word-error rate reduction on Broadcast News recognition task) than a discriminative language model using word-based feature.

→ The language model predicts the probability of the word string and the associated tree jointly, using MaxEnt probability estimation.

→ The model was evaluated on a English-to-Arabic translation task improved the points over a word trigram model.

(iv) Languages without Word Segmentation:

→ Although agglutination produces a large number of long and complex word forms in many languages, other languages do not possess any explicit segmentation of character strings into words at all.

→ In languages such as chinese and Japanese, sentences are written as sequence of characters delimited by punctuation signs but without any intervening whitespaces.

→ Rather than trying to match linguistically defined words, it is also possible to optimize segmentation directly for language model performance.

→ Sproat et al. showed that the dictionary used for Chinese word segmentation significantly influences the perplexity of a bigram language model trained on the segmented text and that is possible to iteratively optimize the dictionary by merging frequent word co-occurences:
such that the perplexity is reduced at each iteration.

Note: such approaches are similar to the Data-driven algorithms for deriving morpheme-like subword units.

→ Another example of a data-driven approach, in this case for Japanese, uses chunks of characters for language modeling.

→ chunks are derived from training data by selecting the highest-frequency n-grams and patterns.

(v) Spoken versus Written Languages:

→ Statistical language modeling crucially relies on large amounts of written text data, and significant trend within language modeling research is the development of methods for scaling current language modeling techniques to ever-larger databases. However, many of the world's 6,900 languages are spoken languages, that is, languages without a writing system.

They are either indigenous languages without a literary

tradition.

→ For example, with the many dialects of Arabic which are used in everyday conversation but are almost never found in written form.

→ other languages may be spoken as well as written but may not have a standardized orthography.

→ Only way of obtaining language model training data is to manually transcribe the language or dialect.

→ This is a costly and time-consuming process because it involves (i) the development of a writing standard (ii) training native speakers to use in writing system accurately (iii) the actual transcription effort.

Multilingual and Crosslingual Language Modeling

(1) Multilingual Language Modeling

up to this point we have discussed problems that arise when tailoring statistical language models to a particular language or language type, such as agglutinative languages or languages without word segmentation.

→ The tacit assumption has been that the resulting language model is used in an application where only the language of interest is encountered.

→ Here speakers may use several languages or dialects

side by side, often within the same utterance.

→ The phenomenon of code switching exists in a variety of bilingual and multilingual communities or where diglossia exists, such as where a formal std language is used in addition to colloquial or dialectal varieties.

② Crosslingual Language Modeling:

Another question that might be asked is whether data in one language can be leveraged to improve a language model for a different language.

→ The most straightforward way of applying this idea is to automatically translate the foreign-language text into the desired language and to use the translated text as additional language training data.

→ A unigram extracted from the translated text was interpolated with a trigram baseline language model.

→ The use of machine translation technology for processing out-of-language data is likely to fail when the desired language does not have sufficient data to train a machine translation system.

→ A drawback of the previous approaches is that the quality of the resulting model is heavily dependent on the translation accuracy

for: $P_{En}(w) = \sum_k \phi_k^{En}(w)\theta_k^{ch}$

- Assuming that the source language is chinese ($^{ch}_{(w)}$) and target language is English (En).
  - $P_{En}(w)$ is the marginal word probability probab distribution of english.
  - $\phi_k(w)$ is the probability assigned to word 'w'.
    θ_k is the prior for the k^{th} topic.

→ target-language marginals are then incorporated into the target-language model as follows:

$$P_{target}(w/h) \propto \left(\frac{P_{bLSA}(w)}{P_{base}(w)}\right)^{\beta} P_{base}(w/h)$$

- where P_{bLSA} is the adapted probability
  P_{base} is the baseline probability

→ this approach enforces a one-to-one topic correspondence across languages.