Linguistics and NLP tasks

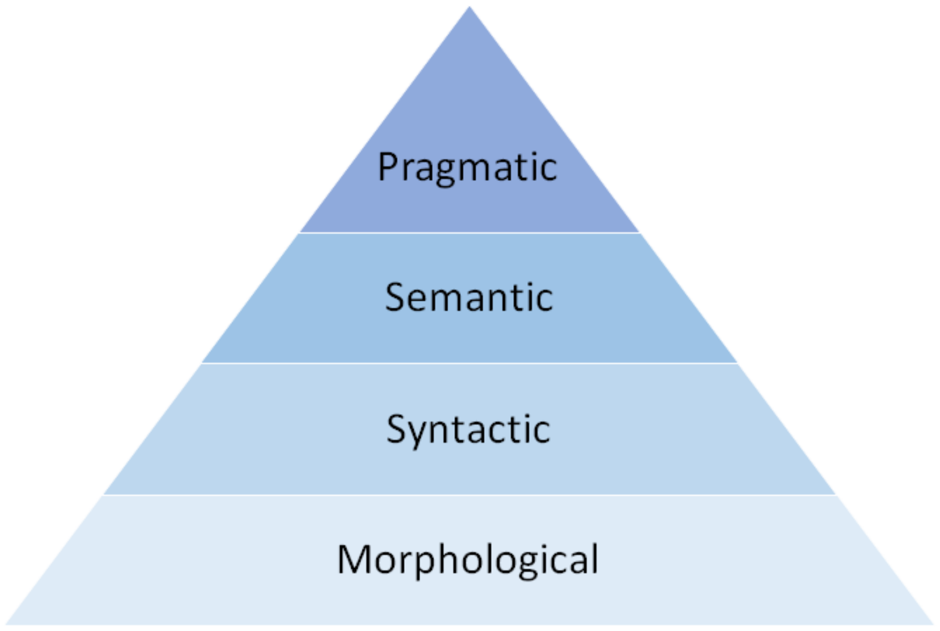
By

Mirza Rahim Baig

Agenda

- Tasks in NLP - overview
- Motivation: working with real data
- Morphological processing
- Syntactic processing
    - POS tagging
    - Constituency parsing
    - Dependency parsing
    - Information extraction

Natural Language Processing Pyramid



Pragmatic: discourse and contextual awareness

Semantic: the meaning of the words, sentences

Syntactic: structure of the language, relationship between words, and the roles they play

Morphology: words, their forms, variations

Motivation - working with real data

Hands - on

Morphological processing

Tokenization

Splitting an input sequence into ‘tokens’

- Token = unit useful for our processing
- Can be a character, word, sentence

Canonicalization

Example of word variations -

- Valid inflections
- Spelling errors
- Abbreviations

Canonicalization = mapping them back to base form

Stemming

- Rule based
- Chopps of suffix to get to ‘stem’
- driver, drive, driving >> driv
- Popular stemmers: [Porter stemmer](), [Snowball stemmer]()

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Canonicalization - Lemmatization

Dictionary based

Works best when POS tag provided

More sophisticated than stemming

Slower than stemming, but result is the actual base form

Popular lemmatizer: WordNet from Princeton University

Wordnet Online

Original	Stemmed	Lemmatized
visibilities	visibl	visibility
adhere	adher	adhere
adhesion	adhes	adhesion
appendicitis	append	appendicitis
oxen	oxen	ox
indices	indic	index
swum	swum	swim

Phonetic hashing

Spelling variations of word induced by pronunciations

E.g: Bangalore vs. Bengaluru, Delhi vs Dilli

Need to reduce variations of word to common form

Soundex algorithm:

- Reduce word to 4 letter code
- Codes represent new form
- Bengaluru, Bangalore: B254

Soundex Code	Letters
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M,N
6	R
No Code	A, E, I, O, U, H, W, Y

Canonicalization - Spell correction

Need notion of distance between words

- Levenshtein Edit distance is a popular measure
- #edits needed to convert source string to target string

LD('test', 'test') = 0

LD('acquire', 'aquire') = 1

Allowed edits -

1. Letter insertion
2. Letter deletion
3. Letter substitution

Calculation process example

Use Norwig method to spell correct

Damerau Levenshtein distance allows 'swap' operation as well

Can also use probabilistic models

Stopwords

Zipf's law

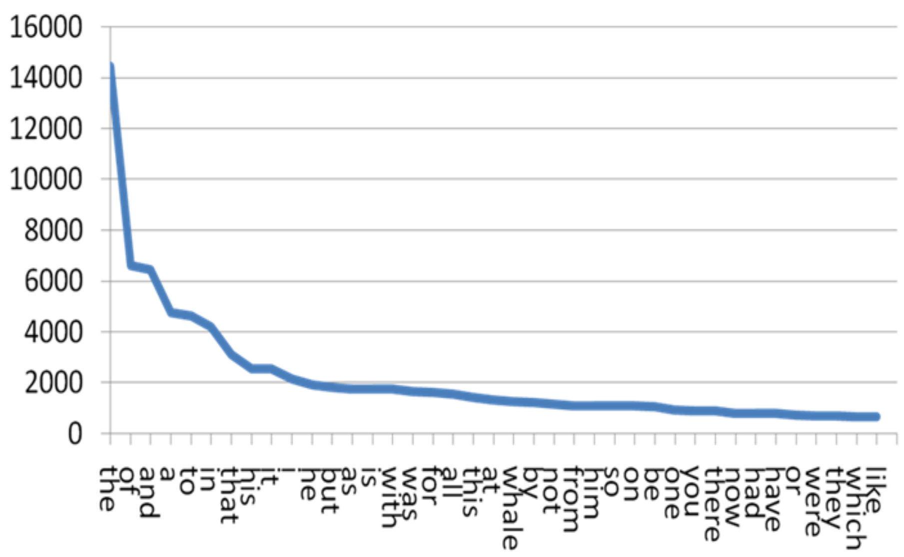
Lot of terms that don't add a lot of value*

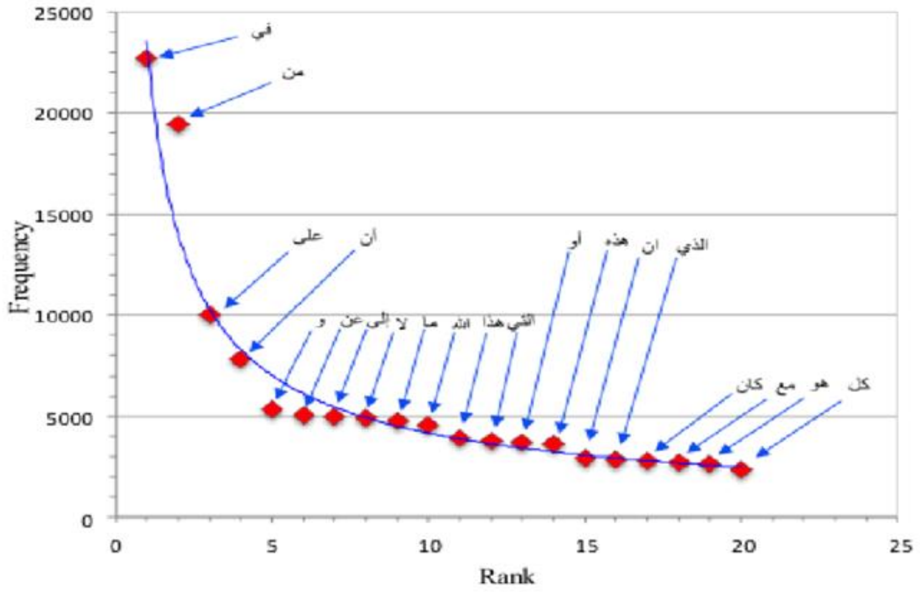
- general/functional terms
- contextual

Very high frequency, and very low frequency terms removed

Functional stopwords inbuilt in NLTK

*Careful! 'Value' definition changes with task!





Collocations

"Air India", "Ice Hockey", "Hong Kong", "King Kong"

Identifying collocations:
- Terms occur together much more than you'd expect by chance
- Need measure to capture this

[Pointwise Mutual Information]

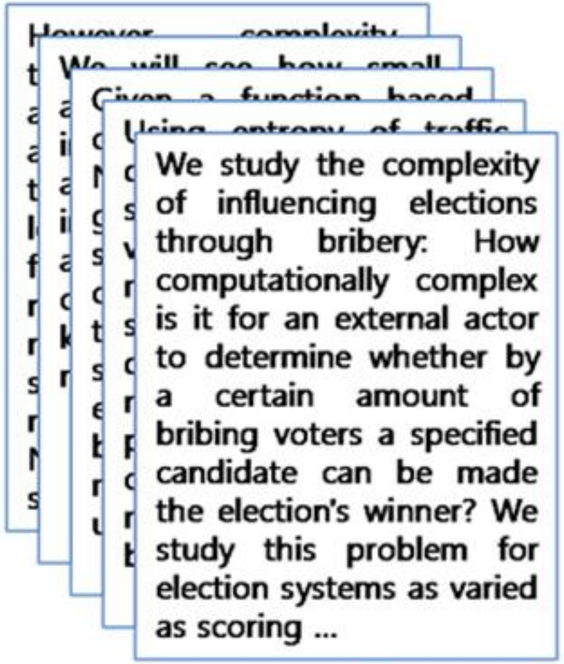
$$\operatorname{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

- Calculate PMI for all occurence contexts
- Use threshold to qualify phrases
  - Threshold depends on dataset, size
- Use chain rule for >2 terms

word 1	*word 2*	PMI
puerto	rico	10.0349081703
hong	kong	9.72831972408
los	angeles	9.56067615065
it	the	-1.72037278119
are	of	-2.09254205335
this	the	-2.38612756961

Text pre-processing complete! Let’s continue with the hands-on.

Text representation - bag of words model



Corpus of text

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Term Document Matrix

From unstructured to structured

- Vector space representation
- DTM or TDM
- Can now use regular ML models

TF model: Term frequencies in cells

Text representation - tfIdf

High occurrence =/= high importance

tf-Idf is most popular term weighing scheme

If term appears in lesser documents, then its importance for current document should be more

$$tf_{t,d} = \frac{frequency\ of\ term\ 't'\ in\ document\ 'd'}{total\ terms\ in\ document\ 'd'}$$

$$idf_t = log\frac{total\ number\ of\ documents}{total\ documents\ that\ have\ the\ term\ 't'}$$

$$tf - idf = tf_{t,d} * idf_t$$

Back to the hands-on; let’s build the model now.

Syntactic processing

What is syntactic processing?

Syntax

The set of rules, principles, and processes that govern the structure of sentences (sentence structure) in a given language.

Why is syntactic processing even required?

Love is all you need — is all need love you

Dog bites man — Man bites dog

Ken is learning driving in a driving school.

Word order, stop words, morphological forms matter!

Some applications of Syntactic processing -

- Conversational UI
- Question-answering system
- Sentiment analysis

Parsing, and levels of syntax analysis

Parsing: breaking down a sentence into its 'grammatical constituents'.

E.g. Asking Siri, “who was the Chancellor of Germany in 2018”?

Finding such relations and dependencies between words can be done through parsing

Levels of syntactic parsing -

1. Part of speech (POS) tagging
2. Constituency parsing
3. Dependency parsing

We’ll look at each of them.

Syntactic processing - POS Tagging

Parts of speech

Syntactic class of a term based the role it plays.

Terms of the same POS class can be swapped without affecting the syntax of the sentence

My **cat** eats fish
My **car** eats fish

8 main classes:

nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions and interjections

Definitions
Some examples

- Divided into further sub-classes
- Penn Treebank uses 36 granular tags

Parts of speech - Quiz time!

I bought a **beautiful** dress at the mall
a) preposition b) adjective c) noun

I want to go to a **university** in the USA
a) preposition b) adjective c) noun

After lunch, let's go for a coffee
a) pronoun b) preposition c) verb

He knocked **but** nobody answered
a) adverb b) adjective c) conjunction

If we finish it **quickly**, we can leave
a) adverb b) verbc) conjunction

Well, I don't think I'll make it on time
a) interjection b) conjunction c) preposition

I dropped the keys **under** the table
a) adjective b) preposition c) pronoun

Yesterday, I ate my lunch quickly
a) adverb b) noun c) preposition

Parts of speech Tagging

Main approaches -

1. Lexicon based
2. Rule based
3. Probabilistic
4. Deep learning

POS Tagging - Lexicon Based

- Use a training corpus
- For each word, assign highest occurring POS tag in the training set

E.g. If ‘Verb’ occurs most times for ‘drive’

- I like to drive
- I lack the drive

Both instances of ‘drive’ will be tagged with ‘VB’

This simple approach too gives about 90% accuracy

POS Tagging - Rule based

Lexicon base not good enough.

Define a set of rules -

- Discovered from corpus, or
- Tagged manually

E.g.

- Tag all words ending with 'ing' as verb (VBG; present participle)
- Tag all instances of ‘learning’ as NN

Applying -

1. Start with lexicon base tagger to assign based tags
2. Apply rules to fine tune the tagging

Can lead to significant increase in accuracy
Can also inject domain knowledge through rules

POS Tagging - Probabilistic tagging

Achieve high accuracy without manually handcrafting linguistic rules

Formulated as a stochastic process -

- Look at the sequence to assign a tag
- Tag for a word depends on some previous words/tags.

Probabilistic tagging - Bayes theorem and our formulation

Calculate the probability of the hypothesis, given some evidence.

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$

For multiple evidences -

P (Class = c1 | x1, x2) = P (x1, x2 | c1). P(c1) / P(x1, x2)

Our formulation for POS tagging:

- Markov process
- Assumed local dependencies
- POS for term depends on the term, and POS for the neighbouring terms

Finding most likely parse sequence for "the big show" from candidate sequences

E.g. P (DT, JJ, NN | 'the', 'big', 'show')

Markovian Assumption: Probability depends only on the previous event

- Greatly simplifies the process

Viterbi Heuristic

For K possible tags, n sequence length, #candidate tag sequences: K^n

- 36 tags in Penn Treebank; 36^3 candidates for sequence of length 3

Viterbi Heuristic -

- Follows greedy approach
- Assign tags one at a time, to get maximum probability until that point
- Tag dependent only on the previous tag

$$[P(W_1 | T_1) * P(W_2 | T_2) * P(W_3 | T_3) * P(T_1) * P(T_2 | T_1) * P(T_3 | T_2)]$$

$$\Downarrow$$

$$[P(W_1 | T_1) * P(T_1)] * [P(W_2 | T_2) * P(T_2 | T_1)] * [P(W_3 | T_3) * P(T_3 | T_2)]$$

P(tag|word) = P(word|tag) * P(tag|previous tag)

= Emission probability * Transition probability

Hands-on

Syntactic processing - Constituency Parsing

Constituency parsing

Shallow parsing can't -

- tell if a sentence is grammatically correct
- Understand dependencies between terms

Constituents: grammatically meaningful groups of words/phrases (e.g. noun phrase)

The man	walked	to the door
The dog	jumped	at his owner
The pigeon	flew	up the building
Noun Phrase	**Verb**	**Prepositional Phrase**

Most common constituencies:

- Noun phrase
- Verb phrase
- Prepositional Phrase

Constituency parsing is breaking sentence into constituents

Context-Free Grammars

Setup that can describe all possible strings in a language

G = (N, T, P, S)

- N: set of non-terminal symbols (POS tag)
- T: set of terminal symbols (words in the vocabulary)
- P: set of production rules
- S: start symbol

S	$\rightarrow$	NP VP
NP	$\rightarrow$	ART NOUN
NP	$\rightarrow$	NP PP
PP	$\rightarrow$	P NP
VP	$\rightarrow$	VERB NP
VP	$\rightarrow$	VERB NP PP
ART	$\rightarrow$	the
ART	$\rightarrow$	a
NOUN	$\rightarrow$	telescope
NOUN	$\rightarrow$	man
NOUN	$\rightarrow$	spider
VERB	$\rightarrow$	saw
VERB	$\rightarrow$	complimented
P	$\rightarrow$	with
P	$\rightarrow$	in

A -> B

- A is POS tag (non-terminal)
- B can be POS tag or terminal

NP -> DT N | NP PP

- LHS can produce RHS
  - The/**DT** cat/**N**
  - The/**DT** cat/**N** in/**P** the/**DT** pool/**N**

Context-Free because rule doesn't depend in the context in which it appears.

Parsing using CFGs

S: top most production symbol

Parsing:

- Generate sentence from top most symbol ← **Top down**

- Reduce sentence to top most symbol ← **Bottom up**

Results in a tree like structure

Top Down Parsing

Given a grammar and a sentence, we'll assess if we can generate the sentence from S

Sentence:
The bird sang

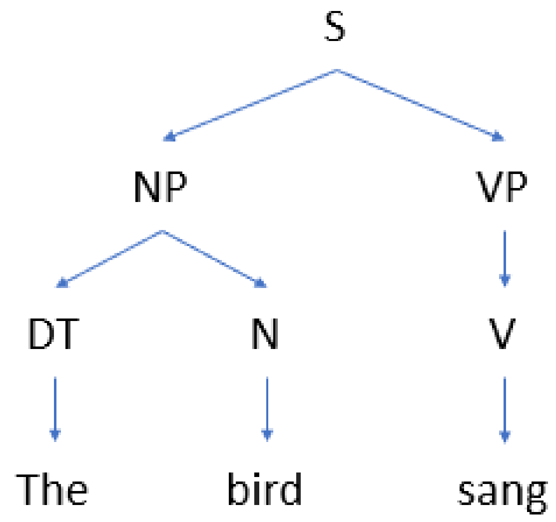
Grammar:
S -> NP VP
NP -> N | Det N
VP -> V | V NP
DT -> 'the'
N -> 'bird'
V -> 'sang'

S -> NP VP

...

S -> NP VP
S -> (Det N) VP
S -> (The) N VP
S -> (The) (bird) VP
S -> (The) (bird) (V)
S -> (The) (bird) (sang)

...



Bottom up Parsing

Keep reducing elements to non-terminal symbols until you get to S

S -> (The) (bird) (sang)
S -> (The) (bird) (V)
S -> (The) (bird) VP
S -> (The) N VP
S -> (Det N) VP
S -> NP VP
S -> S

Shift-reduce parser is popular algorithm

Probabilistic context free grammars

CFGs give multiple parse trees in case of ambiguity

- Look at the man with one eye
- Look at the man with one eye

<= We need to pick the most likely tree

PCGFs have rules along with probabilities

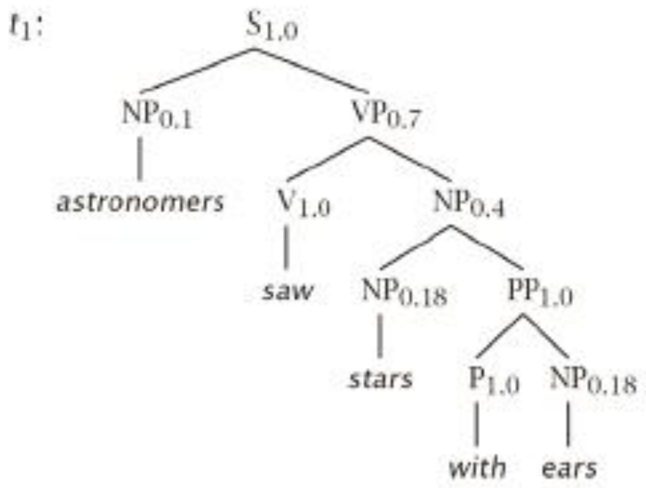
G = (M, T, R, S, P)

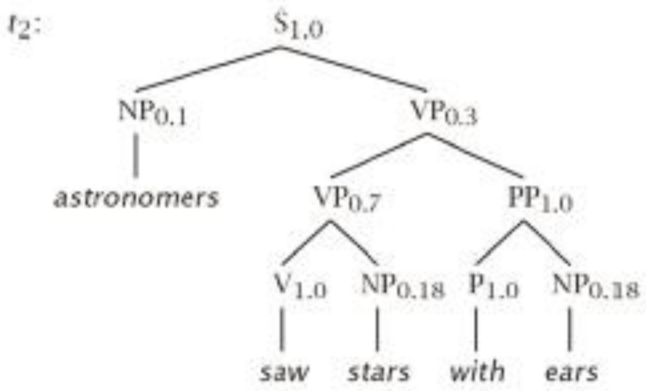
- M is the set of non-terminal symbols
- T is the set of terminal symbols
- R is the set of production rules
- S is the start symbol
- P is the set of probabilities on production rules

NP -> Det N (0.5) | N (0.3) |N PP (0.2)

Each tree has a probability of generating the sentence

- Multiple probabilities of all the nodes

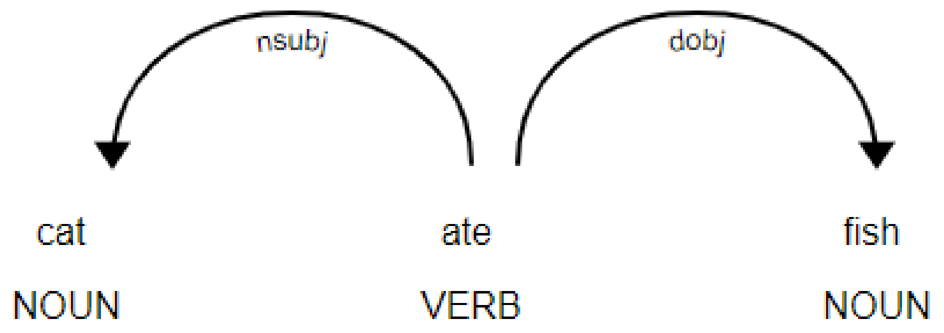




Syntactic processing - Dependency Parsing

Dependency parsing overview

- Not looking at constituencies (VP etc.) anymore
- Dependencies between words themselves



Basic idea of dependency: each sentence is **about something**

- Sentences (esp. declarative) follow SVO structure
- **Subject** (who does something), **Verb** (the action), **Object** (the object of the action)

Note: dependencies can be long range

Throwback to school days

Grammar: subject, predicate, modifiers

Subject:
The topic of the sentence; includes who or what the sentence is about

Object:
Object of the sentence on which the action happens

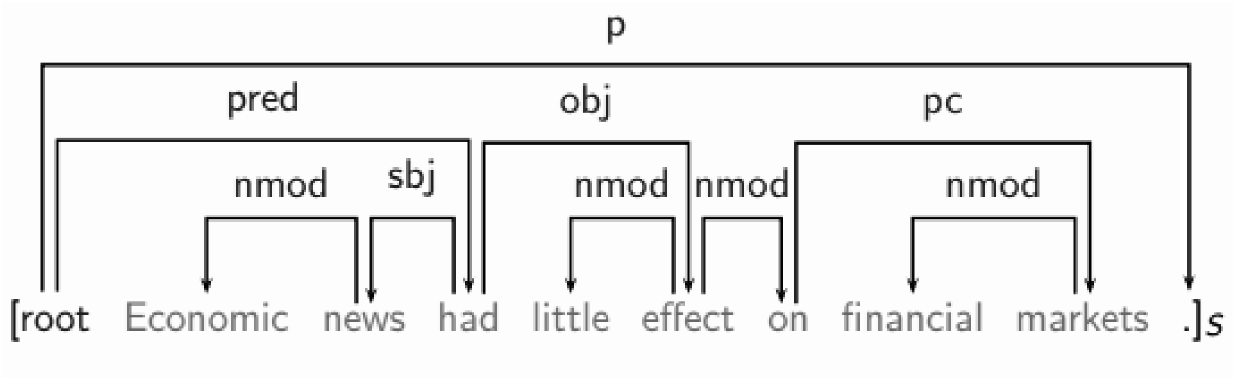
Modifier:

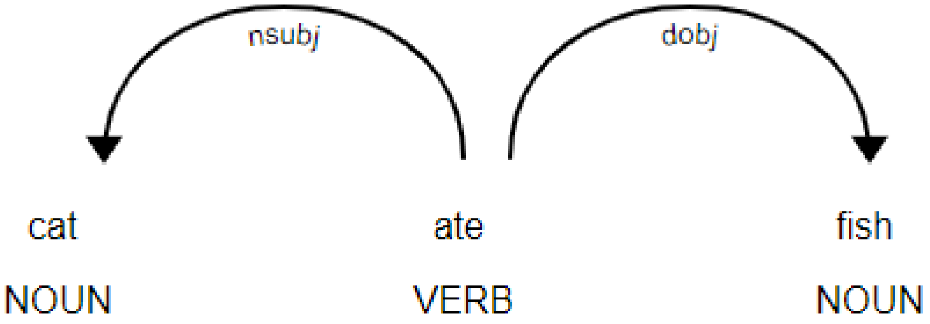
An optional element within a sentence, removal just makes the sentence less specific.

Links to learn/practice -
https://www.slideshare.net/srgeorgi/grammar-subject-predicate-modifiers
http://guidetogrammar.org/grammar/objects.htm

Dependency grammars

- Dependency/word grammars based on identifying dependencies
- Based on notion that words play different roles (subject, modifier, object)
- Not tree but graph





- Arrows from Head -> Dependent
- Rooted at main verb or predicate for sentence

Universal Dependencies

- Commonly occurring labels across languages
- Cross lingual framework for dependency annotation

Examples -

Nsubj: nominal subject
A nominal phrase which is the syntactic subject of a clause.
- 'Clinton' in 'Clinton defeated Dole'
- 'Car' in 'The car is red'

csubj: clausal subject
'said' in 'what she said made sense'

amod: adjectival modifier
'red' in 'Sam eats red meat'

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Full specification: http://universaldependencies.org/docs/u/dep/index.html

Creating the dependency tree

For detecting dependencies

- No hard set of rules
- Machine learning based approaches
    - Treebanks created for dependency grammars (Penn Treebank too)
    - Use treebank for training

Creating the tree -

- Modified Shift reduce parser

Syntactic processing - Information extraction

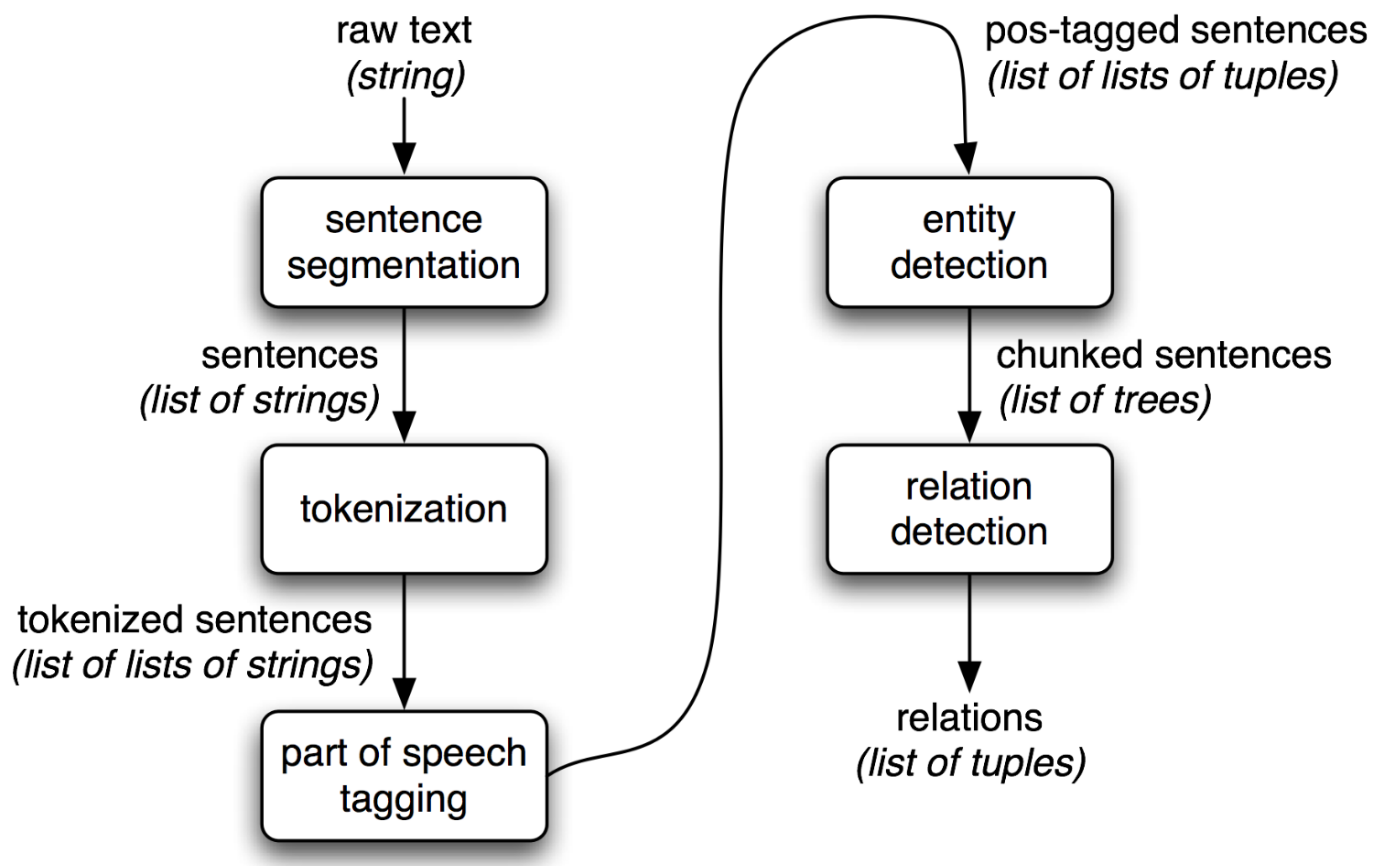
Information extraction - Named Entity Recognition

IE used in a wide variety of NLP applications -
- conversational agents (chatbots), etc.
- Virtual assistants such as Apple's Siri, Amazon's Alexa, Google Assistant etc.
- extracting structured summaries from large corpora

Named entity recognition: Subtask of Information extraction
- seeks to locate and classify named entity mentions
- pre-defined categories (names, organizations, locations, time expressions, monetary values, etc.)

Elon Musk [PERSON] apparently wasn't aware that his company SpaceX had a Facebook [ORG] page. The SpaceX and Tesla [PRODUCT] CEO has responded to a comment on Twitter [GPE] calling for him to take down the SpaceX, Tesla and Elon Musk [ORG] official pages in support of the #deletefacebook movement by first [ORDINAL] acknowledging he didn't know one existed, and then following up with promises that he would indeed take them down.

Information Extraction Pipeline



[Good overview of the steps here.]()

Reference material

Speech and Language processing, Dan Jurafsky and James H. Martin

NLTK book

Coming up next:

Topic Modeling

THANK YOU