(*) Compositional semantics :-

→ if we understand individual meaning, then we can get overall meaning.

→ " The meaning of an expression is function of and only of the meaning of its parts and the way in which the parts are combined".

challenges: (1) syntactic structure.
e.g. house rent & rent house
Both have diff meanings.
(2) Semantic Preference
(3) Polysemy of constituent words.
(4) Idiomatic & metaphoric usages.

(*) Distributional semantics :-

→ " words that occur in similar contexts tends to have similar meaning".

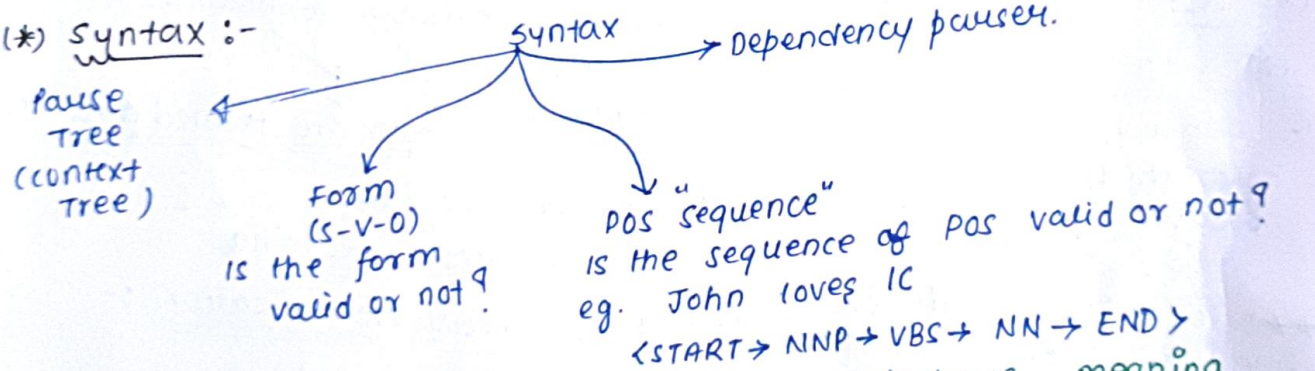
e.g. house and flat frequently occurs with content words rent, bedroom, sale, etc.

- conditions for two entity to be similar acc. distrib^n sem:-

(1) Performing same activities / actions.
(2) Performing same actions under certain condition.
(3) Frequency of performing activities also same.

(***) "Even though two entity / words are semantically similar and if they does not satisfy above (3) conditions then they are not similar according to distributional semantics".

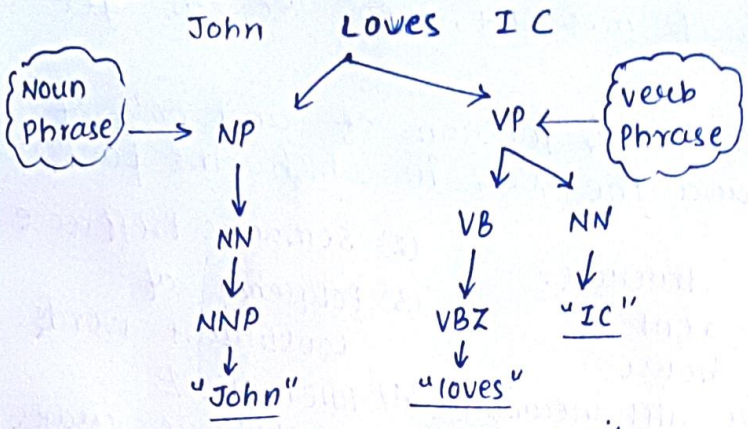
(*) Syntax :-



(***) A sentence without grammar can still have meaning and a sentence without meaning can still have grammar.

e.g. { Colorless Dreams sleeps furiously }
grammar ✓ but no meaning.

(*) Parse Tree :- (context-Tree)



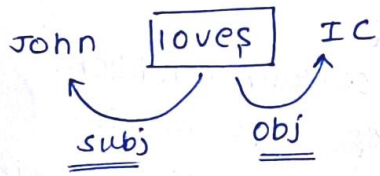
(***) we can't extract subject, obj from parse tree.

(***) parse tree doesn't tells us the relationship.

→ Parse tree is not necessarily a binary tree.

→ According to chomsky, any parse tree can be converted to a Binary tree called Chomsky Normal Form (CNF).

(*) Dependency Parser :-



(***) DP tells direct relationship between john and loves & loves and IC.

(***) DP takes longer time than parse tree.

(**) If we want to get subject and object then only we will use dependency parsing else we will use parse tree.

(*) NLP Preprocessing :-

(1) stop words Removal :
- depends on situation whether to remove stopwords or keep as it is.

(**) In neural models, we don't remove stopwords.
↳ In neural model we doesn't take into account the count of words. Thus, it doesn't affect much. whereas in statistical model it can affect.

(2) Punctuation removal :
Machine-learning & machine learning
(*) "de-hyphening required".
Both are treated differntly

(3) De-abbreviation :
ML → machine learning.

(4) Numerical Normalization :-
50 - fifty

(5) Lemmatization / stemming :-
(**) lemmatization tooks more time than stemming.
saying → say plays → play
playing → play

Neural → compositional + Distributional.

(=) Domain specific Pre-processing :-

(6) Named Entity Recognition (NER):
- considering New York, machine Learning collectively. rather than considering two different words...

(7) Tense Normalization: [2]
- converting everything to present or past tense.

(8) Plural Normalization:
- convert all words to plural or singular.

(9) Active passive Normalization :-
- convert all sentences to passive.

(*) Vector Space Modelling :-

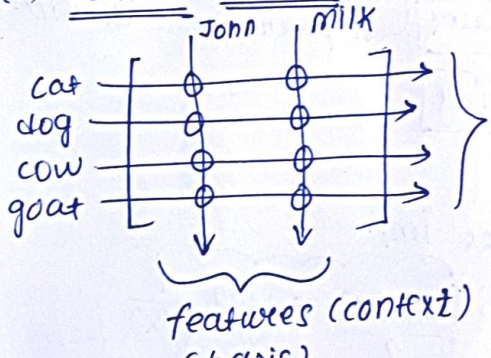
→ Distributional semantics claims that they understand meaning because they understand context.

[w_{j-2} w_{j-1} house w_{j+1} w_{j+2}]
[w_{i-2} w_{i-1} flat w_{i+1} w_{i+2}]
} if these two words satisfies ③ conditions of similarity of distribution semantics then house and flat are similar.

• Diff. Distribution: if there is diff distribution then acc. to distribn semantics they will be not consider similar even though they are similar in reality. ← (:|) Drawback of Distributional sem.

- " Represent a 'linguistic unit' as a vector."

(=) context Matrix :-

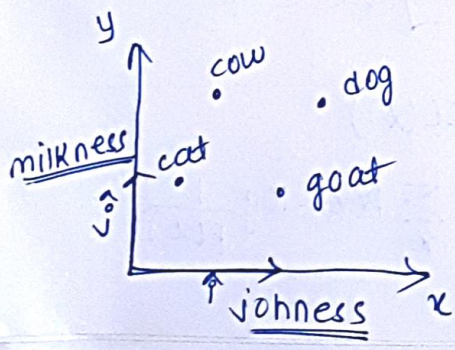


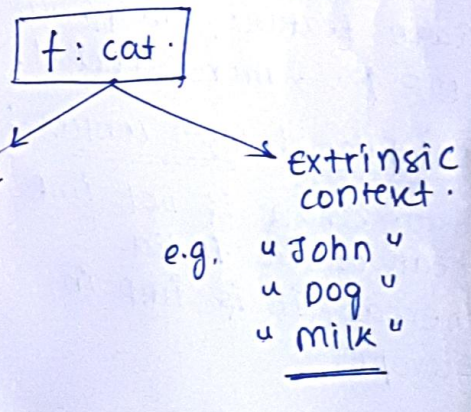
- Given a set of features and set of datapoints
↳ $(f_1, f_2, \ldots f_m)$
↳ $(w_1, w_2, \ldots, w_{|V|})$

f_1 : John
f_2 : Milk
} words associated with each datapoints.

John ≠ John-ness.

(*) John-ness is unique characteristic of John. initially, John ≡ John-ness





(=) context Engineering :

For word : Ext:- word
Int.:- characters.

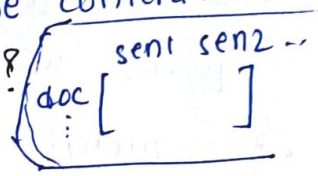
For phrase : Ext: phrase
Int : words.

For sentence : Ext: sentence
Int: words / phrases.

Q. when should we use extrinsic and when should we use intrinsic features ?

→ on document level, we use "intrinsic".
on document level, extrinsic context makes less sense.
e.g., document created on same date and time are in same collection but not similar.
→ if two doc are in same folder we cannot claim that they are similar. As we don't know how the folder was created.
→ Folders are not always organized based on content.
→ The moment we go outside doc. we go outside content.

Q. using sentences as features is preferrable ?



- we are computing similarity bern two docs and asking that 2 doc should contain same sentences, chances of which are very low.
- ∴ usually words / phrases are used as features...

(*) How to Initialize M_c matrix ?

- Linear Transformation is performed by M_c, hence it needs to be learned well.
- Initialize using → Random values (maybe belonging to a particular distribn)
  → TF-IDF modelling

• TF-IDF Modelling :-
→ Rare features are more important.
→ Features occuring in all docs are less imp.
→ Rare features helps in clustering.
→ IDF ↑ ⇒ more special.

$$IC(x) = \log \frac{1}{P(x)}$$

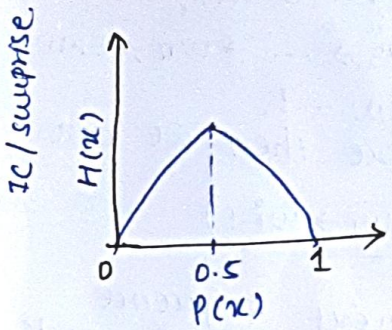
Information content.

• Conditional Infon content :-
- Autorikshaw is not imp. feature in India. whereas it is imp in Europe.

$$H(x) = E(IC(x))$$

Entropy

$$= \sum_x P(x) \cdot \log \frac{1}{P(x)}$$



we are not interested in $p=0$ & $p=1$.

↑ we are certain that something will not happen.

↑ we are certain that something will happen.

- **Inverse Document Frequency :-**

$|D|$ ← Total no. of documents.

$\#d_i$ ← no. of docs in which $f_i \in d_i$

$$IDF = \log \frac{|D|}{\#d_i}$$

↑ information content.

$$TF(cat) = \sum_{d_i} \frac{|W_{cat} \in d_i|}{|W \in d_i|}$$

f_i = tail w_j = cat.

$$P(f_i | w_j) = \frac{count(f_i, w_j)}{count(w_j)}$$

for doc, w_j becomes d_j

- while finding $P(f_i|w_j)$, we don't consider all other w's and f's.
- Each calculation is independent in terms of rows and columns.

(~~**~~) **Assumption**: All features are independent of each other.

- for this assumption, If we have seq. "John loves IC". If we want to predict "IC" after "John loves", then if we assume that they are all completely independent of each other then model will not be able to predict "IC".
- All of them must have some dependence. Hence unigram models are worst language models.

(***) But it is a good way to initialize, rather to initialize with random values. It is a kind of inductive bias.

$$H(f_i | w_j) = P(f_i | w_j) * \log \frac{1}{P(f_i)}$$

Doing across all w_j's

$$H(f_i) = \sum_j H(f_i | w_j)$$

This assumption is also known as "Bag of words" assumption.

(*) Language Modelling :- $\langle w_1 \to w_2 \to w_3 \to \dots \to w_n \rangle$

$\langle START \to w_1 \to w_2 \to w_3 \to \dots \to w_n \to END \rangle$

- Language model can find $\hat{P}(w_1 \to \dots \to w_n) = ?$

If we can compute probability of a sentence then we can

$\hat{P}(w_n \mid w_1 \to w_{n-1})$ ↑ Predicting next word of the sequence.

and $\hat{P}(w_i \to w_n \mid w_1 \to w_{i-1})$ ↑ Predicting next sequence given a sequence of words.

Gram Modelling

$P(w_1 \to w_2 \to w_3) \neq P(w_1 w_2 w_3)$ → sequence does not matter in this case.

$= P(w_1) * P(w_2 \mid \leftarrow w_1) * P(w_3 \mid \leftarrow w_2 \leftarrow w_1)$

$= P(w_1) * P(w_2 \mid w_1) * P(w_3 \mid w_2 w_1)$ → start from begin

or

$= P(w_3) * P(w_2 \mid w_3) * P(w_1 \mid w_2 w_3)$ → start from end.

$P(w_3 \mid w_2 w_1)$ ↑ search in a doc where w_1 and w_2 occurs together.

$P(w_3 \mid \leftarrow w_2 \leftarrow w_1)$ ↑ search in a doc where w_1 and w_2 occurs together and w_2 occurs after w_1.

Trigram modelling.

$P(w_3 \mid \leftarrow w_2 \leftarrow w_1) \Rightarrow$ gram modelling.

$P(w_3 \mid w_2) \Rightarrow$ Bigram modelling.

$P(w_3) \Rightarrow$ Unigram modelling.

$\hat{P}(w_n \mid \leftarrow w_{n-1} \leftarrow \dots \leftarrow w_1)$ → n-gram modelling. (computationally expensive)

(*) Markov Assumption :-

$\hat{P}(w_n \mid \leftarrow w_{n-1} \leftarrow \dots \leftarrow w_1) \approx \hat{P}(w_n \mid w_{n-1})$

(***) Default degree of markov assumption = 1.

$P(A \to B \to C) = P(A) * P(B \mid \leftarrow A) * P(C \mid \leftarrow B \leftarrow A)$

$P(A \to B \to C) = P(A) * P(B \mid \leftarrow A) * P(C \mid \leftarrow B)$ [∵ Bigram modelling]

$P(C \mid \leftarrow B \leftarrow A)$ approxes $P(C \mid \leftarrow B)$

Problem with Bigram:

e.g. John who lives in NY and works at Google loves IC.

'IC' predicted only based on loves...

we can use Maximum likelihood Estimate (MLE),

$$P(IC \mid loves) = \frac{count(IC, loves)}{count(loves)}$$

we need,

$$\hat{P}(IC \mid \leftarrow loves) = \frac{count(IC \leftarrow loves)}{count(loves)}$$

(***) what if 'IC' and 'loves' have not come together yet.
count(IC, loves) = 0.
model will predict that IC can never occur with loves.
soln:- use various smoothing techniques...

Q. what if there is out of vocabulary situation ?
- solution: use smoothing techniques...

(**) 'BOW' contradicts with language model.
↳ can be used were no seq. required.

(*) Skip-Gram Modelling :- $\hat{P}(w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_i^{\circ} \rightarrow \cdots \rightarrow w_n)$

depends on previous as well as future words.

In trigram modelling,
e.g. John → loves → IC

$$P(IC \mid \leftarrow loves \leftarrow John) = \frac{count(John \rightarrow loves \rightarrow IC)}{count(John \rightarrow loves)}$$

No significant no. of John → loves sequence.
No significant no. of John → loves → IC.
It may happen that 'John' has more influence on 'IC' than 'loves'.

strong influence → (Boost) whenever John comes IC comes.
strong influence → (supress) whenever John comes IC won't come.

(*) Skip-Gram (word2vec) :-

$P(w_{t+j}^{c} \mid w^{t})$ w^{t} :- target word w^{c} : context word.

e.g.: John → loves → IC

$\{ w_{John}^{c}, \underbrace{w_{loves}^{t}}_{\text{center word.}}, w_{IC}^{c} \}$ $m = (1)$ ← context window.

$$\sum_{j=-m}^{m} \left(P(w_{t+j}^{c} \mid w^{t}) \right)$$

$$P(John \rightarrow loves \rightarrow IC) = \sum_{-m}^{m_1} P(w_{t+j}^{c} \mid loves)$$

$$= P(John \mid loves) \cdot P(IC \mid loves)$$

(**) skip-gram assumption: conditional independence of w_s^{c}.

$$P(\text{seq}) = \prod_{w^t=1}^{T} \left(\prod_{j=-m}^{m} P\left(w^c_{t+j} \mid w^t \right) \right)$$

→ In word2vec we initially starts with world of vector space and moves to probability space.

Initially, we don't have the co-ordinates of "John", "loves" and "IC".

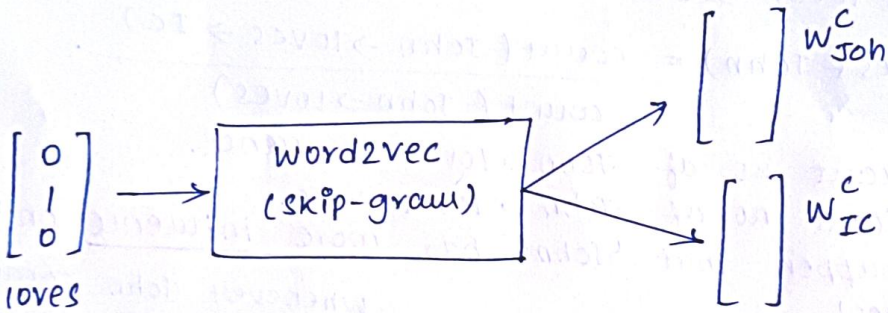
Vocab = {"John", "loves", "IC"}.

Initially, we assume that

Initial
- John ≡ Johnness → $\begin{bmatrix}1\\0\\0\end{bmatrix}$
- loves ≡ love ness → $\begin{bmatrix}0\\1\\0\end{bmatrix}$
- IC ≡ IC-ness → $\begin{bmatrix}0\\0\\1\end{bmatrix}$

} one hot vectors.

length of one-hot vector = |V|



- word2vec gathers all of its friends.
- Take i/p as target word and gather all the context words.
- While training model learns the ideal position of target words.

⇒ Represent context matrix as one-Hot vector matrix.

	$w_1$ness	$w_2$ness	$w_3$ness	$w_{|V|}$ness
w_1	1	0	0 ...	0
w_2	0	1	0 ...	0
w_3	0	0	1 ...	0
⋮	⋮	⋮	⋮ ...	
$w_{|V|}$	0	0	0 ...	1

feature

Data points.

- same words acts as data point as well as a feature.