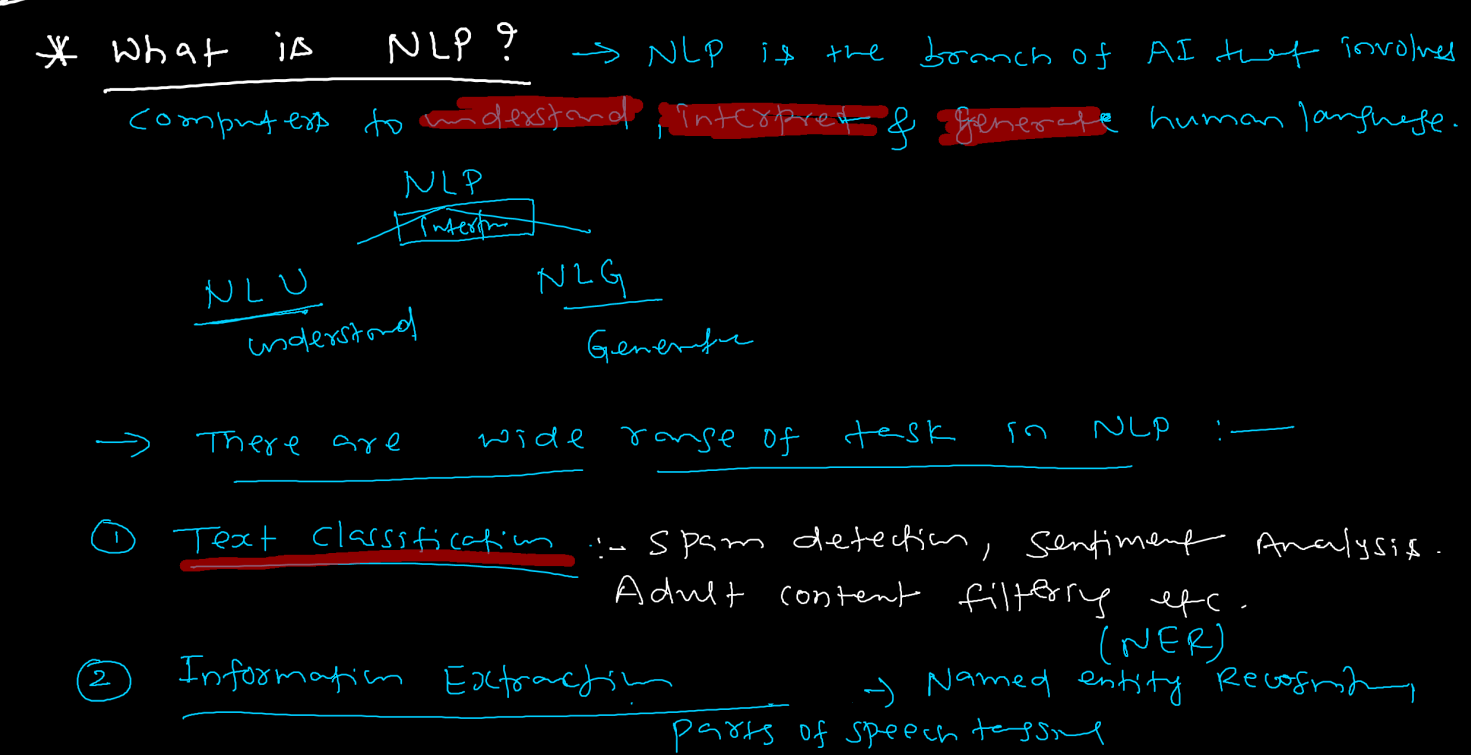
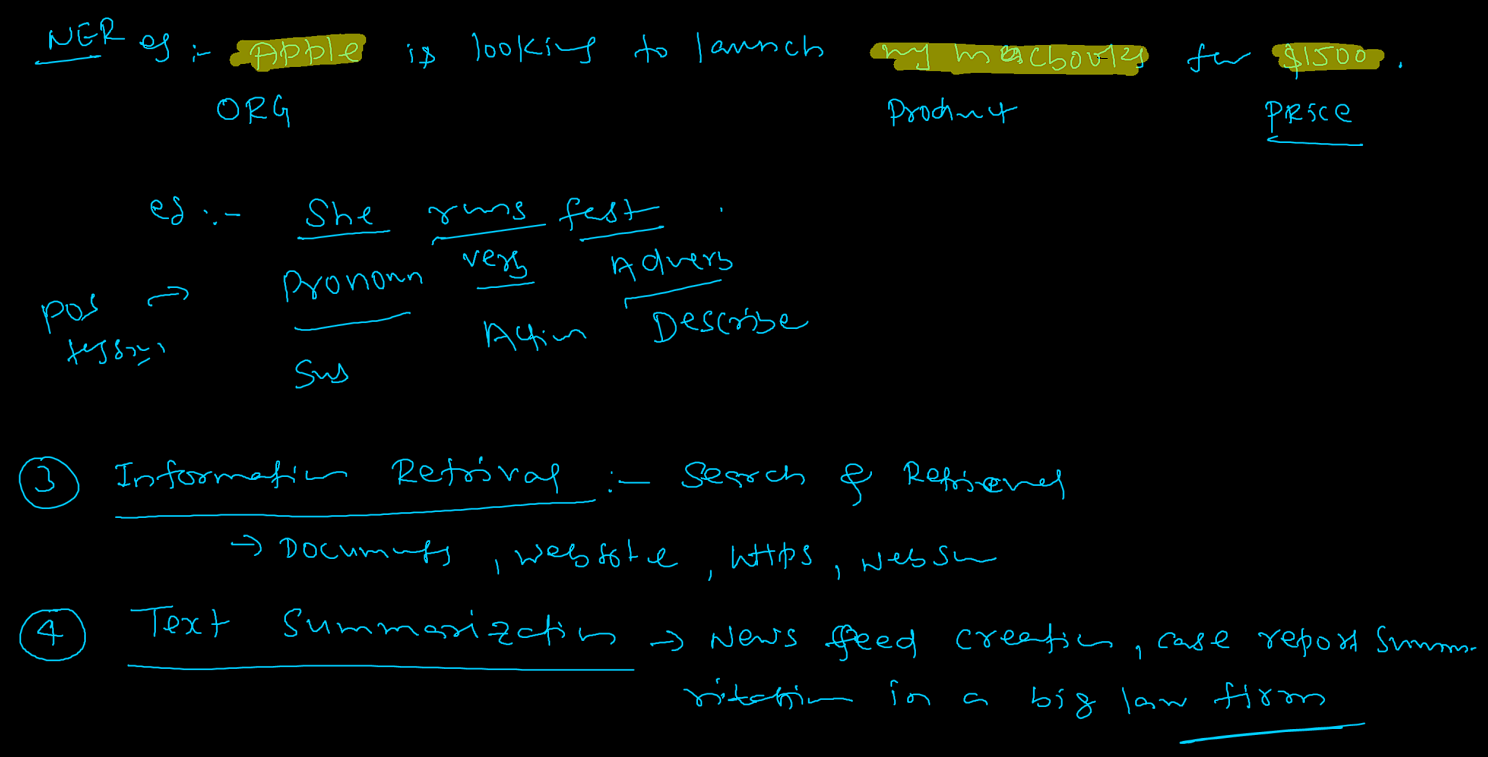
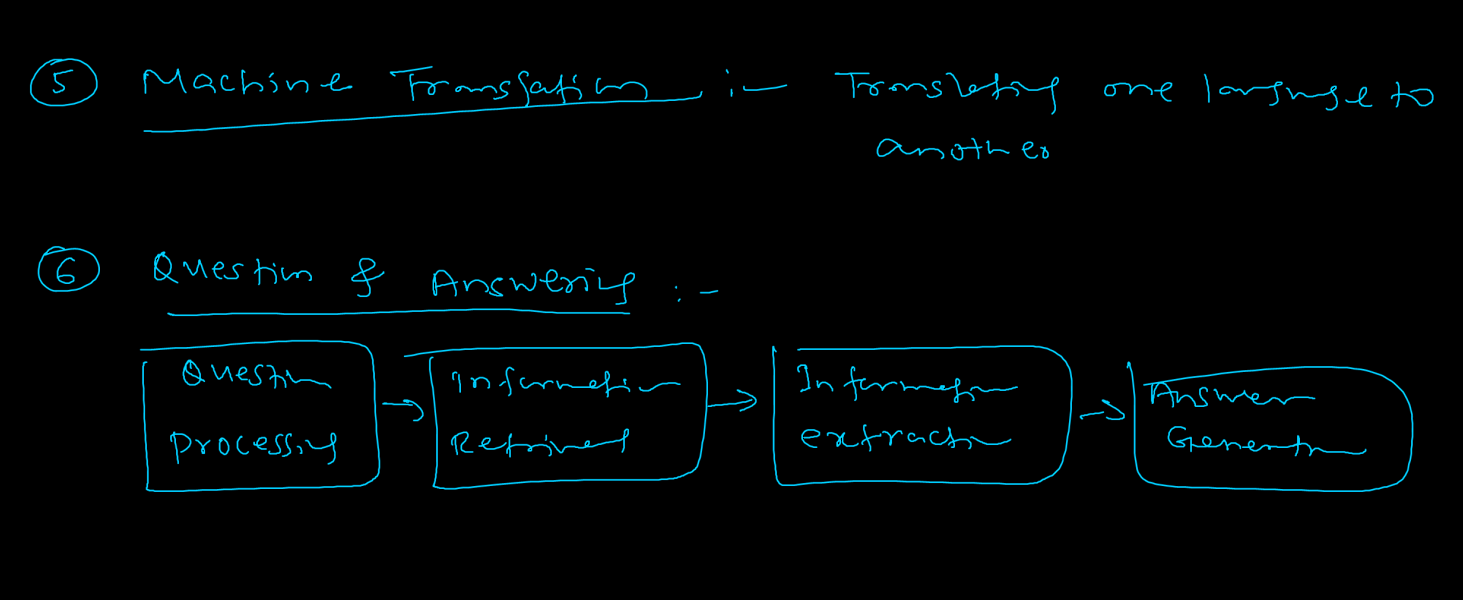
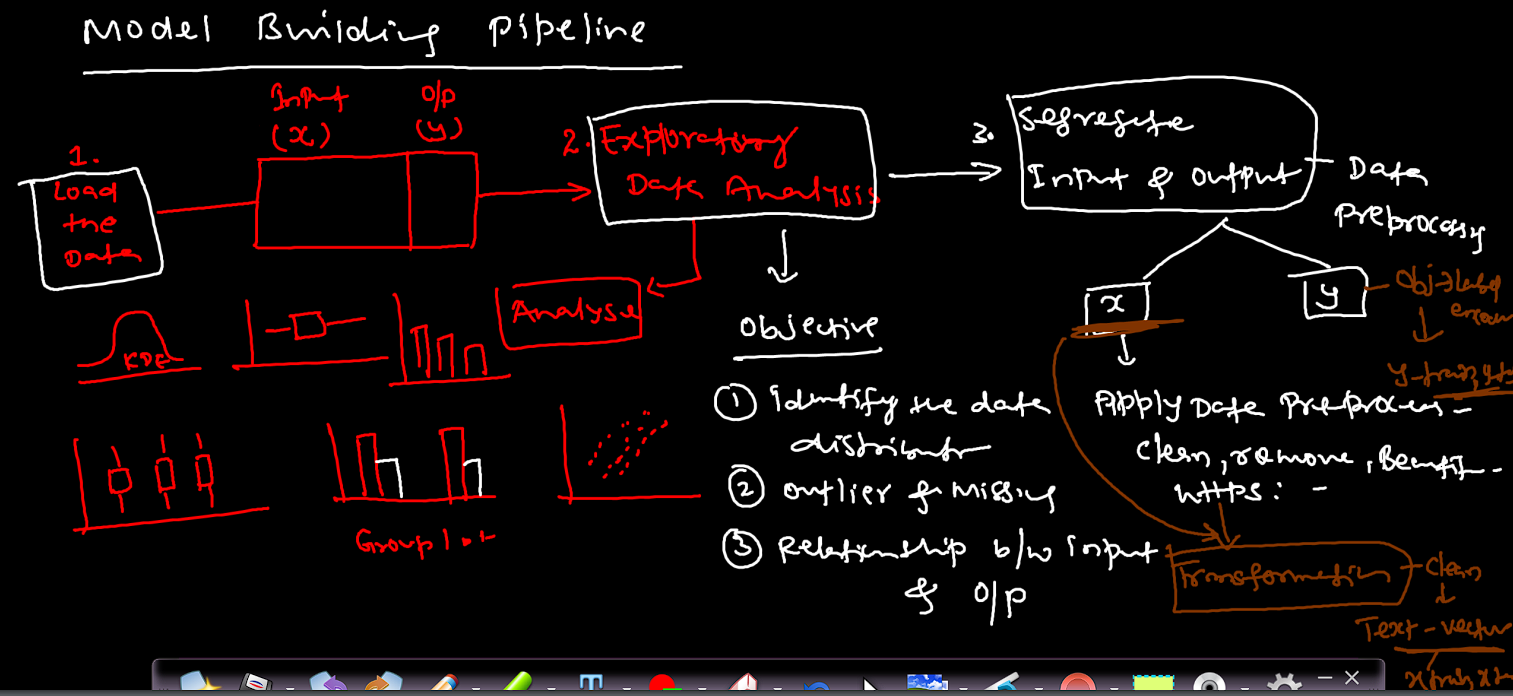
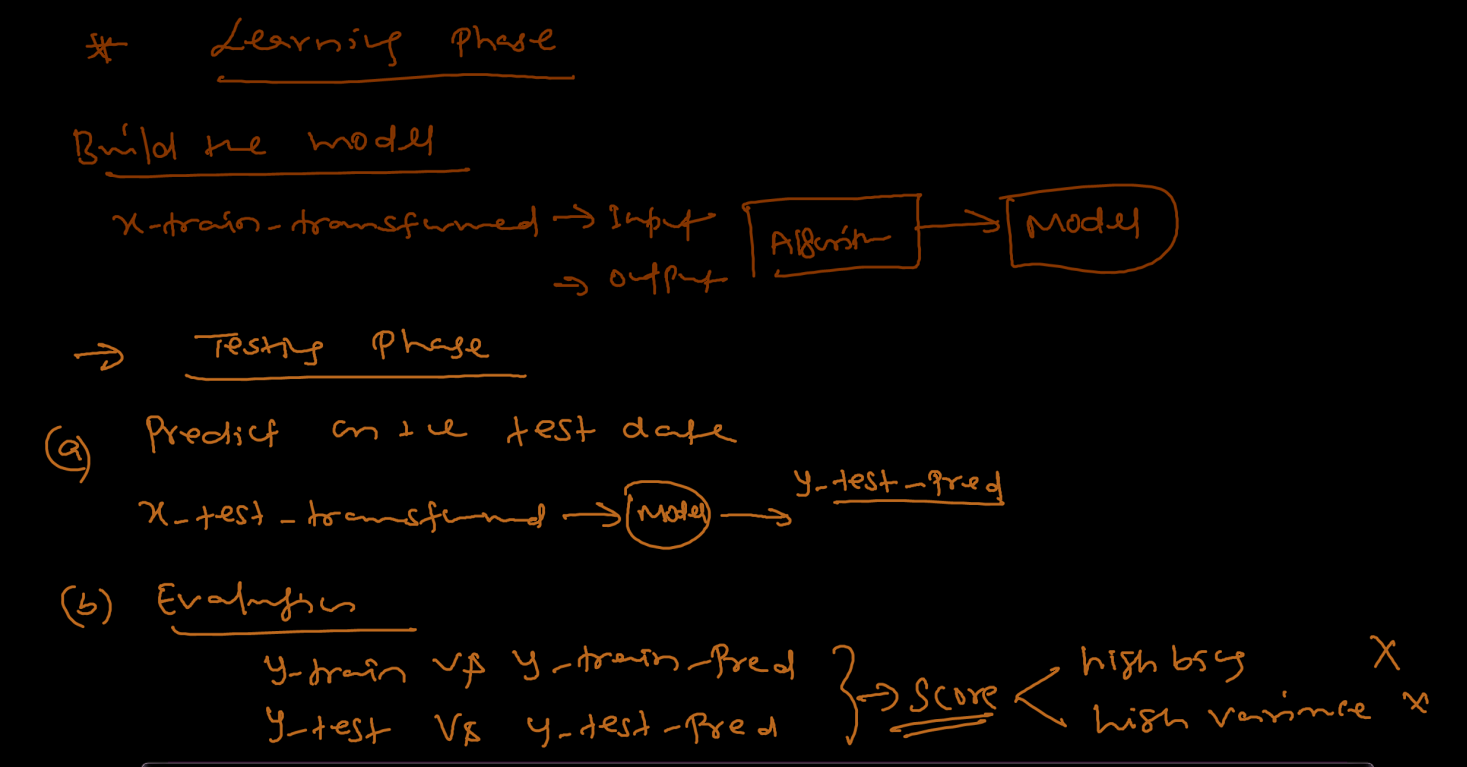
**NLP (DL Approach)**



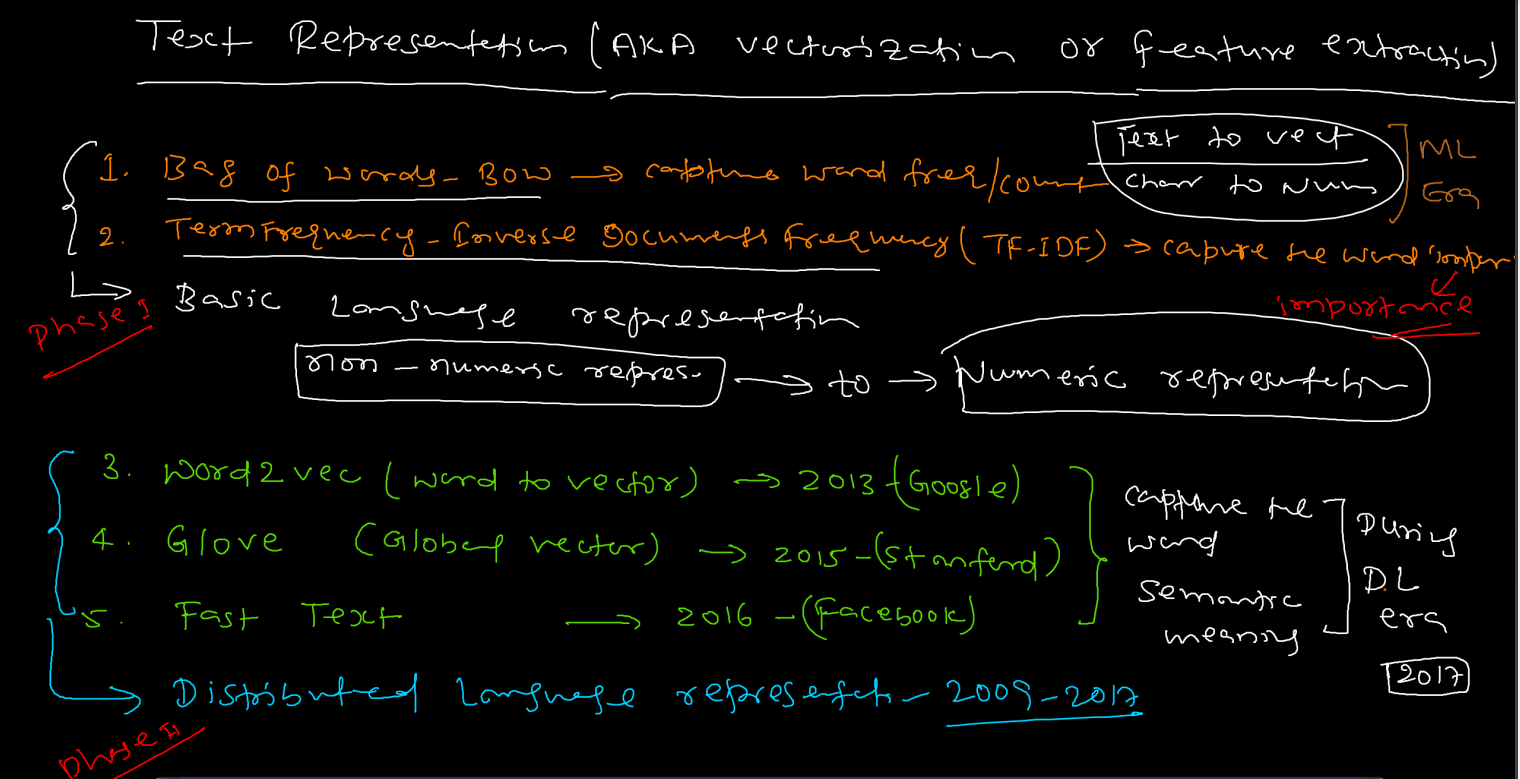


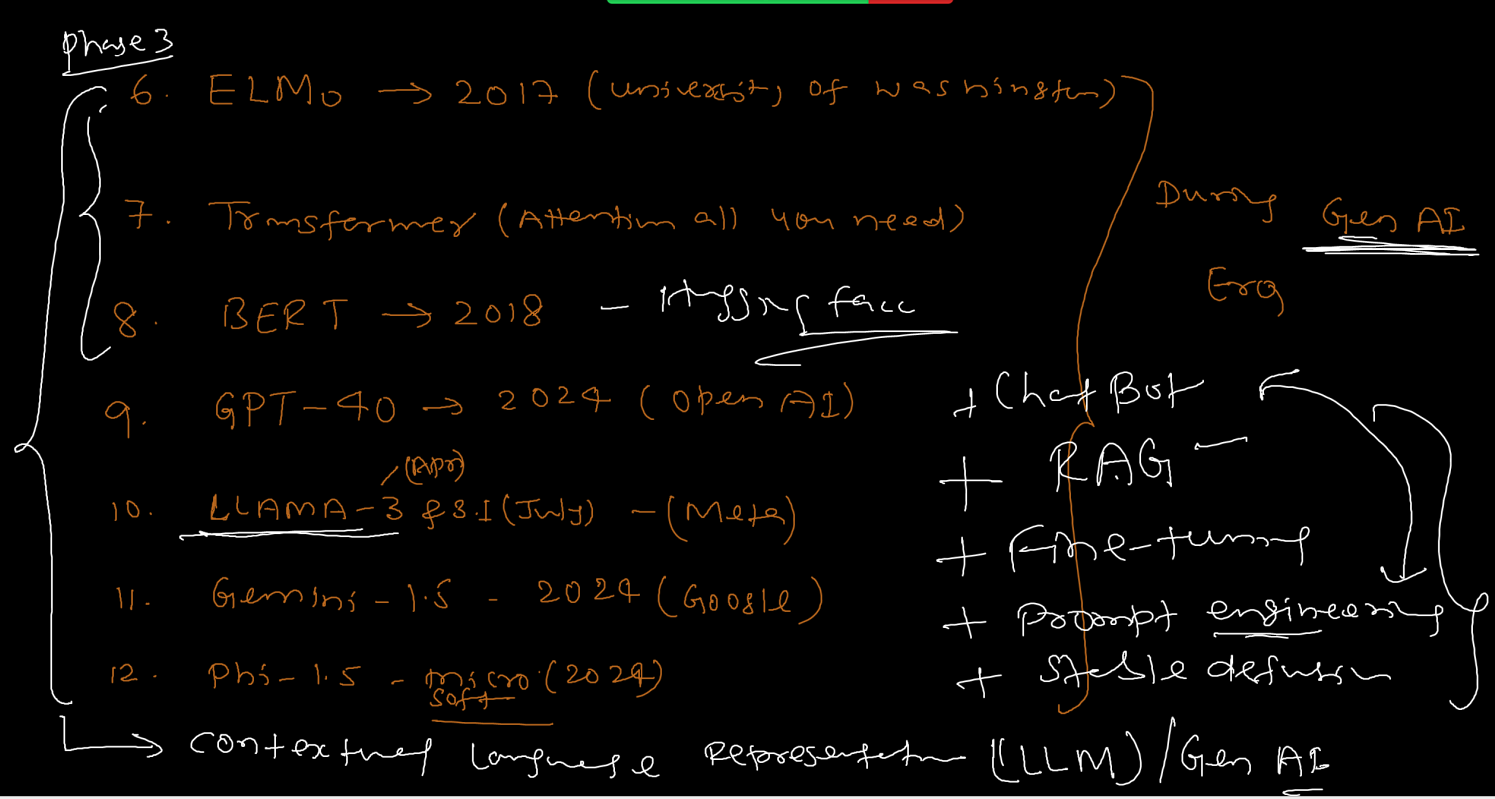


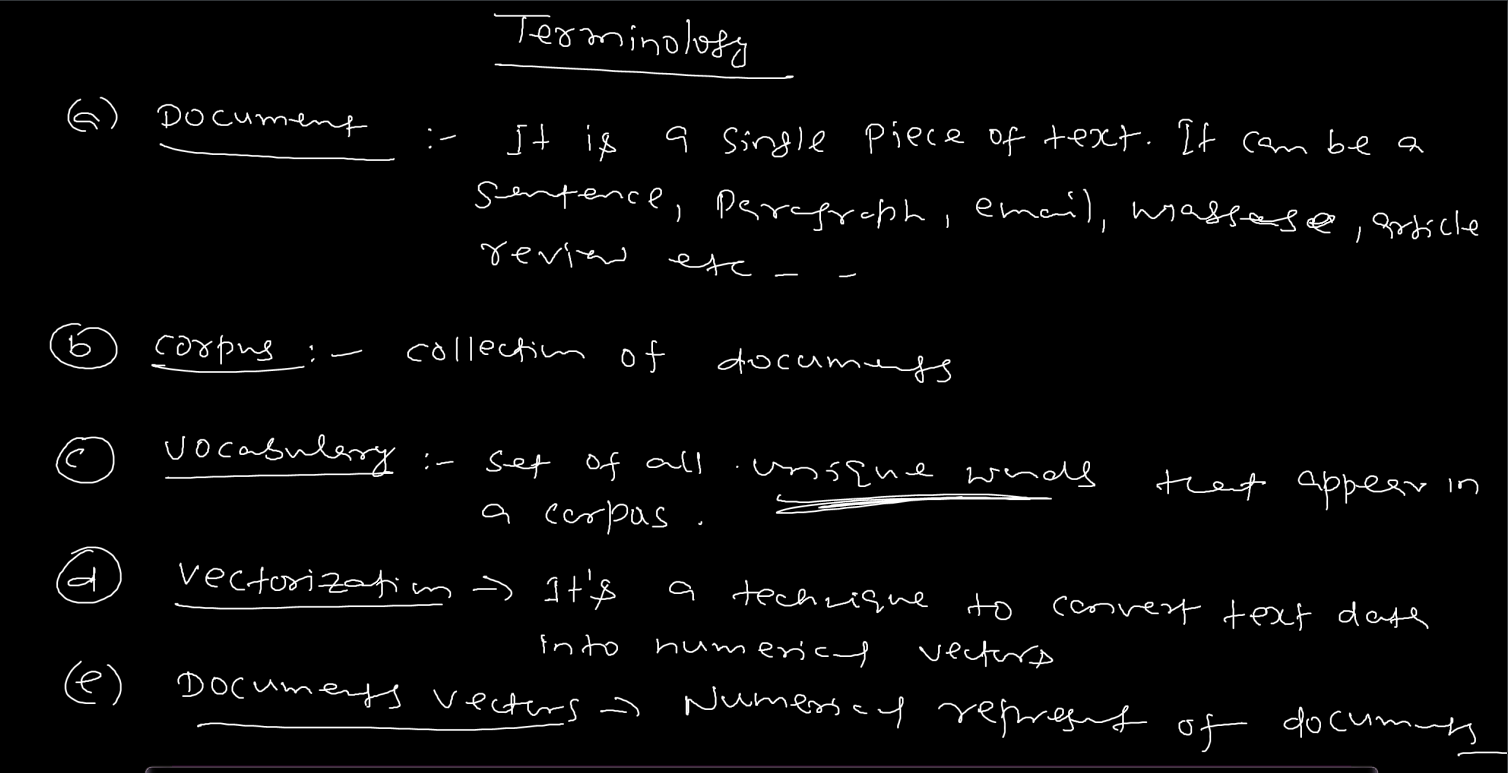


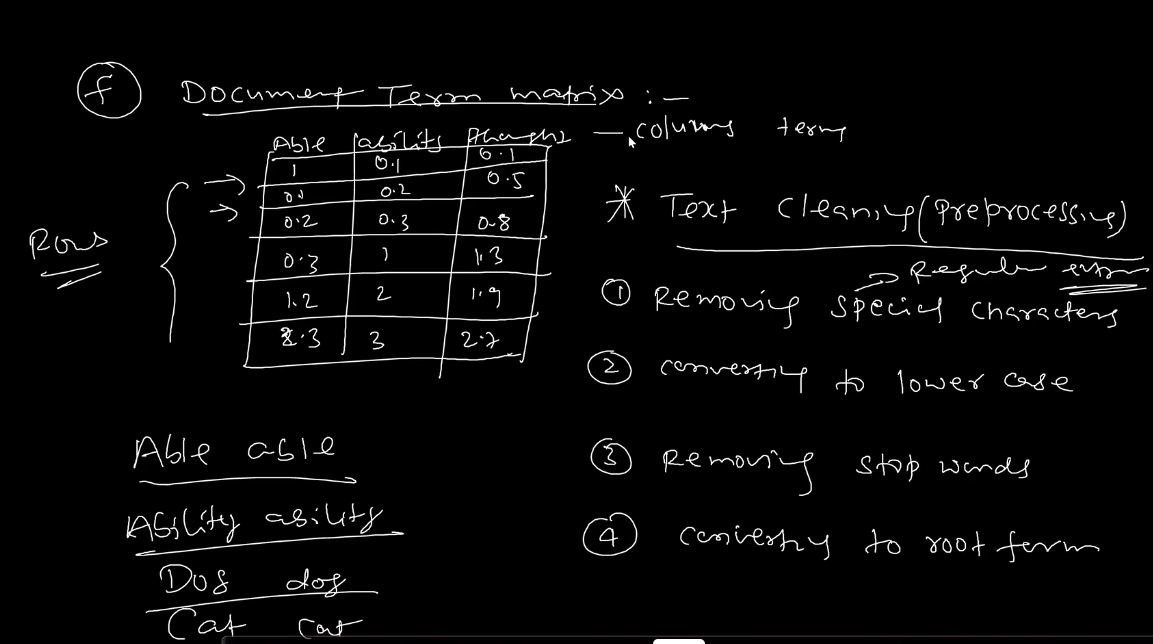


**Vectorization technizues:-**

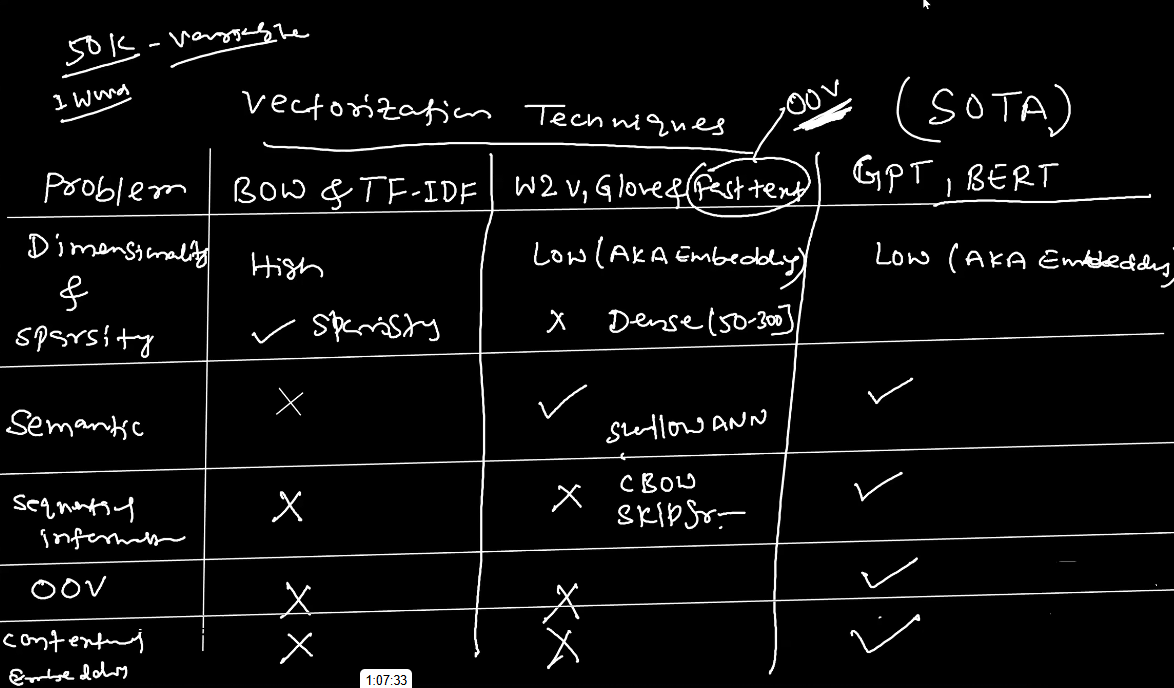








**Steps or Approach to apply different vectorization techniques to solve an NLP task (When to use which):- (interview topic)**

****

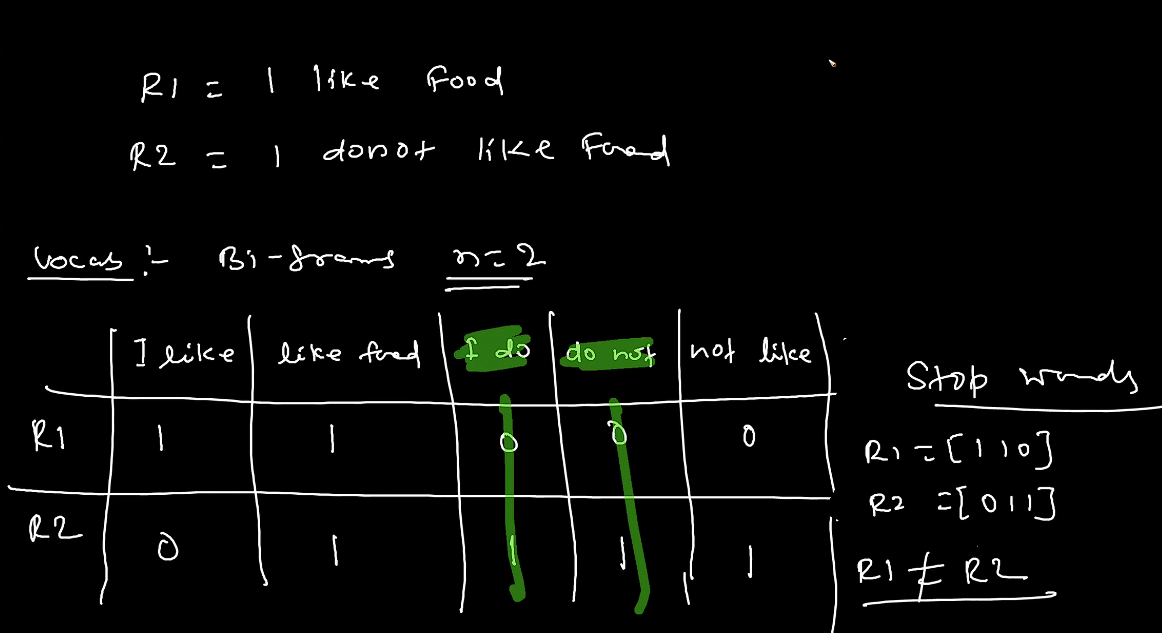
Phase I:- Basic Language Representation:-

Challenges Faced:- High memory consumption, computational complexity, curse of Dimensionality and lack of interpretability.

Solution that can be applied:- Text Preprocessing to check if we can mitigate these challenges.

* Removal of Stop Words
* Converting to lower case
* Removal of punctuations and numbers.
* Converting to root form.

Prob 2:- One thing to note here is that after removal of stop words, sequence information is lost. This can be taken care by applying n-grams (Bi-gram, Tri-gram, etc.,) to BOW method. (interview question)



Prob 3:- if we encounter a different set of words or vocab on which our model is not trained on, i.e., OOV out of vocab, (ex,- I like meal). Our basic language representation is not able to handle this problem.

Similarly, semantic meaning is also cannot be handled by this method.

Hence, our basic language representation cannot handle:-

OOV

Semantic meaning

Creates sparcity

#Note:- n\_gram is also a Data Augmentation technique.

n-grams can handle:-

Positive and negative words

Cannot handle:-

OOV

Since we were encountering major drawbacks in the Basic Language Representation, industry then started to use Distributed Representation Language Model.

Here, the main advantage is low dimensionality. We convert the text into vectors using technique called Word Embedding techniques. Here, we use methods called: -

* Word2Vec (Google)
* Glove (Stanford university)
* Fast Text (Facebook)
* Document Embedding (Doc2Vec)

All these methods are semantic in nature (can easily understand synonyms too.)

These produces dense matrix with low dimension. Typically, token count ranging from (50 – 300).

It uses Matrix Factorization method.

It is a black box method. (since it is a deep learning method).

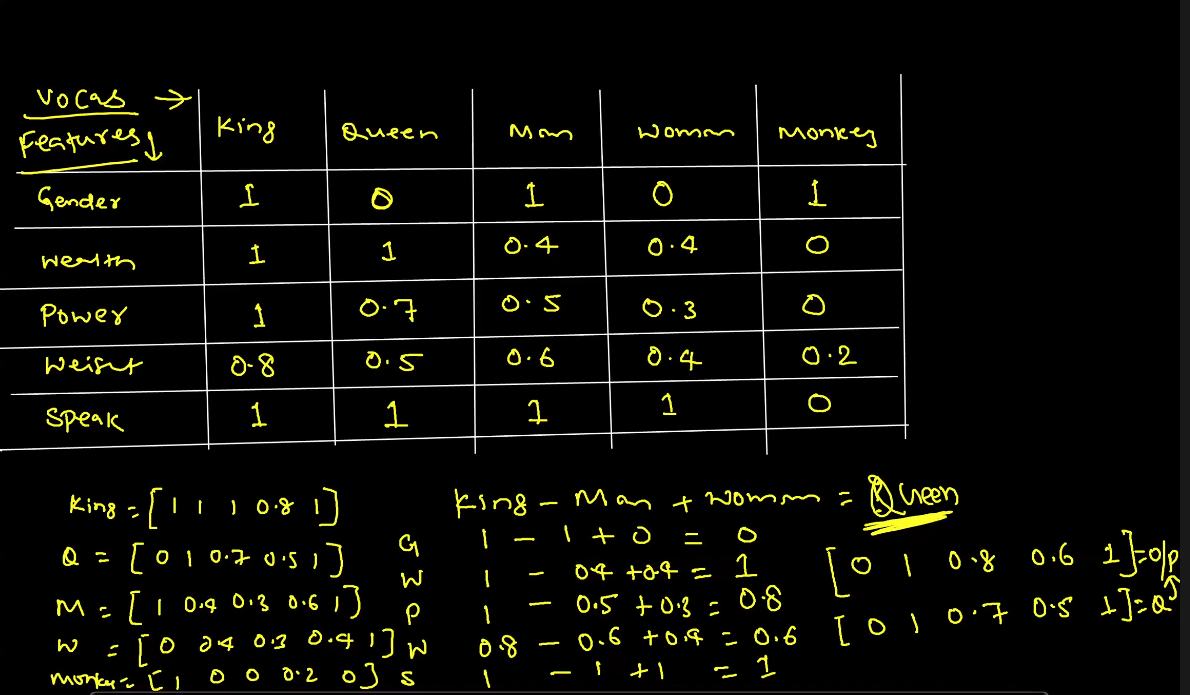
**Properties:-**

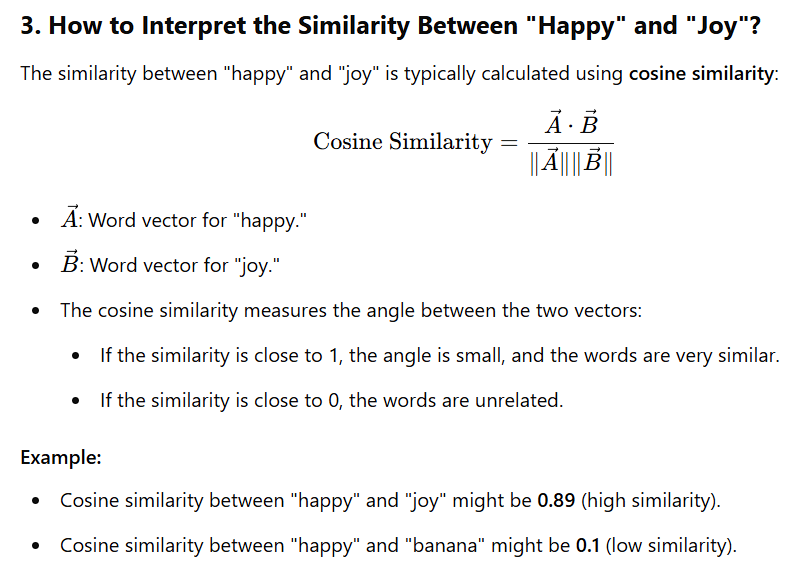
1. Suppose we have 3 words – tasty, delicious, football.

When these are converted into vectors, the vectors of tasty and delicious comes out to be close. Hence, in Word 2 vec if 2 words are semantically similar, then vectors of V1 and V2 are closer.

1. Relationships:- the relationship between two words (man, woman) comes out to be similar to words (king, queen) also the relationship between (india, delhi) comes out to be similar to (usa, Washington d.c)

This is derived from the feature vectors of these words (exact process we have no clue as this is a black box method but the feature vector table gives some idea). The similarity is calculated by methods like cosine similarity and probability values.





Word2Vec model building has two parts:- pre-trained model, self-trained model.

The pre-trained model is trained on 3 million vocab and creates vectors of dimension 300.

Word2Vec has 2 types:-

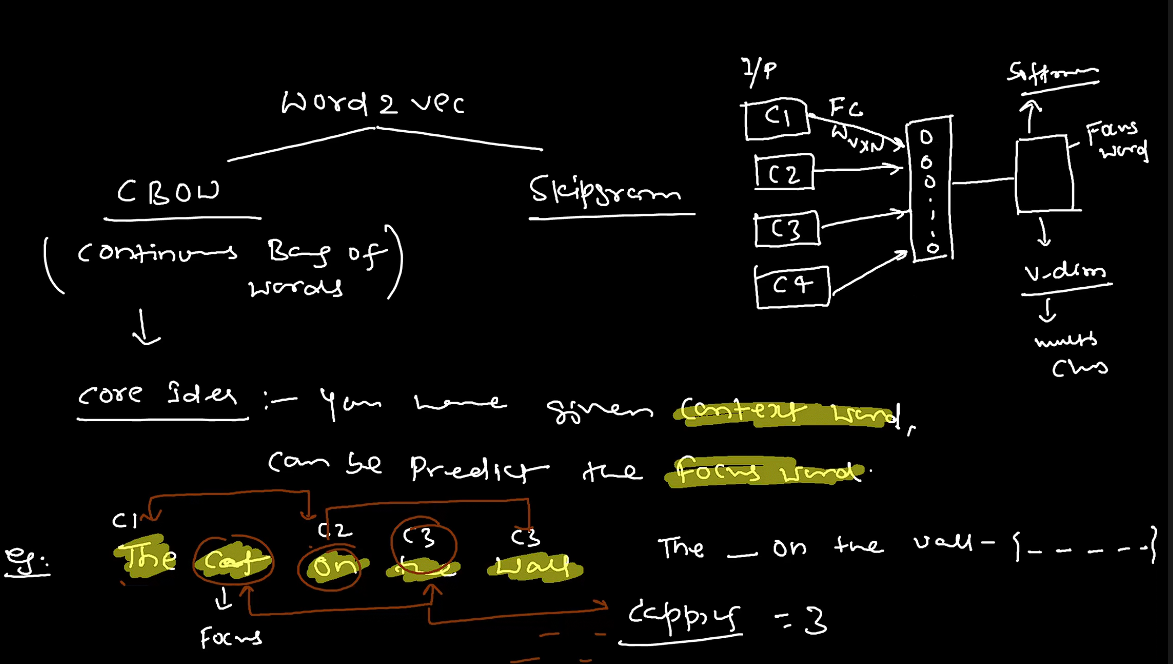
* CBOW (Continuous bag of words)
* Skipgram

CBOW Core idea – you have given context word and based on that can we predict the focus word.

Example – the cat on the wall. Suppose we have to predict the word ‘cat’ which is now our focus word. And the remaining words will become our context words like c1, c2, etc.,

The context words will serve as an input which will then gets connected to a fully connected layer and then on the output layer the softmax activation is applied to calculate the probability. This output layer is also called the vector dimension.

We can also apply the ‘capping’ here which represents the number of words to be taken at a time to predict the focus word.



Dermerits:-

* It is not used in the real time.
* Bad representation for rare words.
* Overfits on frequent words.
* We will get D-dim classification.

Skipgram:-

The idea here is to predict the context words given the focus word.

Here, the one focus word is connected to the dense hidden layers and the outputs are the context words. The ‘softmax’ is applied to each context word to predict the sequence.

We can see that Skipgram is more challenging because now with a single word we now have to predict the whole context. Also the weight calculation is very high here.

Skipgram is computationally more expensive than CBOW. Since cbow uses only 1 softmax and skipgram uses multiple softmax.

Skipgram is well for infrequent words.

