\* Data pre-processing and data cleaning are the same thing. It is a crucial aspect and often comes up in interviews during discussions about NLP.

\* In natural language processing (NLP), we try out and evaluate different feature encoding techniques, build models on them, and choose the technique that gives the best results.

\* Analyse most occurring words that all are related to data. Check for suspicious words.

\* For NLP tasks Multinomial Naive Bayes is used and for classification tasks in ML Gaussian Naive Bayes is used.

\* When evaluating a model, we can check the classification report for each category to see which categories have low recall or precision. If a category has a recall or precision of less than 0.50, we can work on that category to improve its recall and precision. This can be done by giving the model more data for that category. Increasing the amount of data for a category will help the model learn more about that category and make better predictions for it.

\* By looking at the classification report, you can see which categories the model is having trouble with. You can then remove those categories from the model and rebuild it. This will likely improve the accuracy of the model.

\* Before feature engineering, check that the data is cleaned. Make sure that all stop words, punctuation marks, special characters, symbols, and non-ASCII characters have been removed from the data. If you are satisfied with the data, then proceed with feature encoding and model building.

\* If the accuracy is not good, then recheck the data to make sure it is perfectly cleaned. Try different feature encoding techniques and multiple machine learning algorithms.

\* POS classify that which word is noun, pronoun, verb, adverb, adjective, etc.

\* Go to Kaggle and get the idea and make projects of NLP, CNN, etc.

\* You known all the difference ways of performing Data Pre-processing along with codes.

\* Spend time on Text pre-processing steps and their multiple ways along with code. (Important).

\* Question related to TF-IDEF are asked in the interviews.

\* Learn the formulas of TF-IDF by heart because in interviews questions are asked from TF-IDF.

\* NLP is important topic so do good preparation of it.

\* Explore about NLP on google and you tube, Kaggle. Explore text preparation, feature engineering techniques.

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Questions comes like: -

Q) What are the different steps of data pre-processing?

Q) How to remove punctuation marks?

Q) If you are not removing punctuations marks then what is the reason?

Q) How to remove all punctuation marks except semicolon (;) or apostrophe (‘)?

Q) How to remove a particular word?

Q) What is TF-IDF?

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Feature Engineering

\* Feature engineering means feature extraction, converting text into numbers.

\* After feature engineering next step is model building using ML algorithms.

\* Mostly we use N-grams, TF-IDF, Word2Vec (word embedding).

\* One Hot Encoding and Bag of Words are not used anywhere.

\* Study about all methods because in interviews questions they will asks for them.

\* Word embedding means convert text into vectors. There are 2 types of word embedding techniques: - frequency based and prediction based.

\* If interviewer asks what is OHE, BOW, TF-IDF, W2V, N-grams? Tell these are the word embedding techniques.

\* For implementation of word embedding, we using ‘genism’ library.

Different feature encoding techniques:

1) One Hot Encoding

\* One Hot Encoding is the first technique of feature encoding in NLP.

\* It is implemented by get dummies method which we study in python.

E.g.: -

We have 3 documents:

**\* People watch tiger**

**\* Tiger watch tiger**

**\* Tiger write comment**

Corpus: [people watch tiger tiger watch tiger tiger write comment]

Vocabulary: [people watch tiger write comment]

\* We have vocabulary of 5 words.

\* In One Hot Encoding, it makes columns equal to amount of vocabulary.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | people | watch | tiger | write | comment |
| D1 | 1 | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 0 | 0 | 0 |
|  | 0 | 0 | 1 | 0 | 0 |
|  |  |  |  |  |  |

\* We write this in vector form, like this: [ [1,0,0,0,0] [0,1,0,0,0], [0,0,1,0,0] ]

\* In one hot encoding, there is a separate row for particular word.

\* In this way we convert text into vectors. It is called vector representation or number representation.

\* One Hot Encoding is not used in companies because it has several disadvantages.

Disadvantages: -

(i) Sparsity: We are adding unnecessarily zeros (sparsity) in vectors. Putting zeros means we are adding data which is not required. Vocabulary is small but vector becomes big and vector doesn’t contain much information. Suppose, in real time vocabulary there is 2K words and after one hot encoding number of columns made in lakhs. It will slow down the training speed of model. It will create problem when data (reviews or comments) are large.

(ii) Out of Vocabulary (OOV): Suppose any word comes in test data which is not present in vocabulary, then model fails to recognise that word and fails to predict outcome. That word is called out of vocabulary.

(iii) No capturing of semantic meaning: Model cannot capture the semantic meaning like people comes first, then watch, then tiger.

\* Due to these limitations of One Hot Encoding, it is not used in real time.

Advantages: -

(i) Easy to understand.

2) BOW (Bag of Words)

\* It is 90% same as one hot encoding but in BOW sparsity is less.

\* Working of BOW:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | people | watch | tiger | write | comment |
| D1 | 1 | 1 | 1 | 0 | 0 |
| D2 | 0 | 1 | 2 | 0 | 0 |
|  |  |  |  |  |  |

\* In BOW, there is a separate row for separate document.

\* It works almost same as one-hot encoding, but it is better than one-hot encoding because one-hot encoding creates large vectors for documents due to high sparsity, while bag-of-words (BOW) has less sparsity because the vectors are created after the words are reduced in size.

Disadvantage:

(i) Sparsity: Suppose the vocabulary is in lakhs, so the size of the vector is also made large. For example, a document of 15 words would have a vector of lakhs of dimensions, where 15 of the dimensions would be 1 and the remaining dimensions would be 0.

(ii) No capturing of semantic meaning: A model cannot find the semantic meaning of a document based on a vector. The model can only identify which words are used in the document and how many times each word is used. For example, the model can see that the word "tiger" is used twice in the document, but it cannot understand how the document is combined or what the document means semantically.

(iii) Out of Vocabulary (OOV): Suppose any word comes in test data which is not present in vocabulary, then model fails to recognise that word and fails to predict outcome. That word is called out of vocabulary.

(iv) Ordering / Sequence: When text is converted into vectors, the order of the documents is lost. Therefore, the model cannot understand the sequence of the documents.

(v) Suppose there are 2 documents: -

\* This is a good movie.

\* This is not a good movie.

\* One document is positive and the other is negative. However, during sentiment analysis, the model will classify the negative document as positive because the two documents are almost identical. The second document contains the word "not," but the model will understand that 90% of the two documents are identical, so it will classify both documents as positive sentiment.

\* To deal with this problem we have bag of n-grams.

.3) N-grams / Bag of N-grams

\* One-hot encoding and bag-of-words are both **unigram** representations of text, means separate column for each word. This means that they represent each word in a vocabulary as a single feature. For example, the word "cat" would be represented as a single feature in a one-hot encoding or bag-of-words representation.

\* In N-grams if we taking 1 word it is called uni-gram. By default, is uni-gram.

\* 2 words = Bi-gram

\* 3 words = Tri-gram

\* 4 words = Quad-gram

\* 5 words = pent-gram

\* In N-grams we make feature of vocabulary by taking combination of words such as 2 words, 3 words, 4words … combination.

\* N-grams are able to identify semantic meaning.

\* E.g.: - Bi-Grams

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | people watch | watch tiger | tiger watch | tiger write | write comment |
| D1 | 1 | 1 | 0 | 0 | 0 |
| D2 | 0 | 1 | 1 | 0 | 0 |
|  |  |  |  |  |  |

\* E.g.: - Tri-Grams

|  |  |  |  |
| --- | --- | --- | --- |
|  | people watch tiger | tiger watch tiger | tiger write comment |
| D1 | 1 | 0 | 0 |
| D2 | 0 | 1 | 0 |
| D3 | 0 | 0 | 1 |
|  |  |  |  |

\* Using Count vectorizer we have implemented N-grams.

Advantage: -

(i) Understand and able to retain the semantic meaning of documents because feature is made up of combined words and in sequence. It means, it can make features by using document.

(ii) Ordering of document is remain same in vector.

(iii) (v) disadvantage of BOW is a advantage of N-Grams.

Disadvantages: -

(i) Don’t able to understand out of Vocabulary words

(ii) Sparsity

4) TF-IDF (Term Frequency – Inverse Document Frequency)

(i) Term Frequency (TF)

\* TF is separate thing and IDF is separate thing

\* First, we find TF and then IDF. They have separate formulas.

\* Then we multiple both and then the value is assign to the feature.

\* TF-IDF assigns weights to vocabulary based on the frequency of the words in the corpus. Words that are repeated more often in the corpus will be assigned higher weights, while words that are repeated less often will be assigned lower weights.

\* One-hot encoding, bag-of-words, and n-grams all represent text as vectors of whole numbers. The numbers in these vectors represent the number of times each word appears in the document. However, TF-IDF assigns weights to the words in the vector, giving more weight to words that appear more often in the corpus and less weight to words that appear less often.

\* Formula of TF (Term Frequency): -

TF (t, d) = No. of occurrences of term t in document d / Total no. of terms in document d

\* TF is used to identify the importance of particular word /feature in document by calculating their occurrence in document.

\* People watch tiger: People = 1/3 = 0.33. It means ‘people’ word is 33.33% important in document.

\* TF identify the importance of particular word in document.

(ii) Inverse Document Frequency (IDF)

\* IDF identify the importance of particular word in vocabulary by calculating their occurrence in corpus.

Formula of IDF: -

IDF (t) = log e (Total no. of document in corpus) / No. of document with term t in them

\* After calculating the occurrence of word in corpus and in document, we multiply both values and the resulted value is the weight of that particular word / feature. It means it multiply the resulted values of TF and IDF.

Example of TF-IDF: -

**Suppose we have 4 documents.**

**\* D1 – People watch tiger**

**\* D2 – Tiger watch tiger**

**\* D3 – People write comment**

**\* D4 – Tiger write comment**

TF (people, D1) = 1/3

TF (tiger, D2) = 2/3

IDF (tiger) = log e 4/3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | people | watch | tiger | write | comment |
| D1 | 1/3 \* log e (4/3) | |  | 0 | 0 |
| D2 |  |  |  |  |  |
| D3 |  |  |  |  |  |
| D4 |  |  |  |  |  |
|  |  |  |  |  |  |

\* In this way we are convert text into vectors by using TF-IDF.

\* We can implement TF-IDF by using TFIDF Vectorizer.

Disadvantages: -

\* Sparsity: Features/words those are present in particular document have values in particular row and other remaining features contains zero.

\* Out of vocabulary

Where to use: -

\* TF-IDF is used in search engines to rank websites based on how relevant they are to a user's search query. The more times a keyword appears in a website's content, the higher the website will rank in the search results.

5) Word2Vec

\* In Word2Vec we are using DL techniques.

\* Word2vec is very powerful that it will find the relations between words of corpus.

\* We see a project in which we are passing whole 5 books of games of thrones and it will find relations in it.

\* Word embedding means convert text into vectors.

\* It will extract features in vocabulary and assign numbers to them.

\* Word embedding are of 2 types: -

(i) Frequency Based: - One Hot Encoding, BOW, N-grams, TF-IDF these 4 are frequency-based feature encoding techniques because these techniques convert word into number according to their frequency in corpus.

(ii) Prediction Based: - Word2Vec

\* Word2Vec is a prediction-based word embedding technique which uses DL. We will not able to see what’s happening inside the Word2Vec because DL is a black box.

\* Word2Vec comes around in 2014 and it is invented by engineers of google.

\* It will try to extract features from vocabulary and do predictions.

\* We only see that what values are assigning to feature but we cannot see the feature name. Feature names are like F1, F2, F3 …

\* It will assign values to features in that way: -

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | king | queen | man | women | monkey |
| gender | 1 | 0 | 1 | 0 | 1 |
| wealth | 1 | 9 | 7 | 6 | 0 |
| power | 1 | 8 | 6 | 5 | 6 |
|  |  |  |  |  |  |

\* How king is representing in vector: [1,1,1]

\* The function of Deep Learning (DL) is feature extraction from data. DL randomly extracts features based on vector representations and assigns weights to these features.

\* In games of thrones project, we see that Word2Vec assign random numbers to features.

\* There are 2 types of Word2Vec: -

\* Both are exactly opposite of each other.

(I) CBOW (Continuous Bag of Words): - In CBOW model will generate a fake problem / dummy problem in which it will do predictions.

E.g.: - Document = **watch tiger for great stunts.**

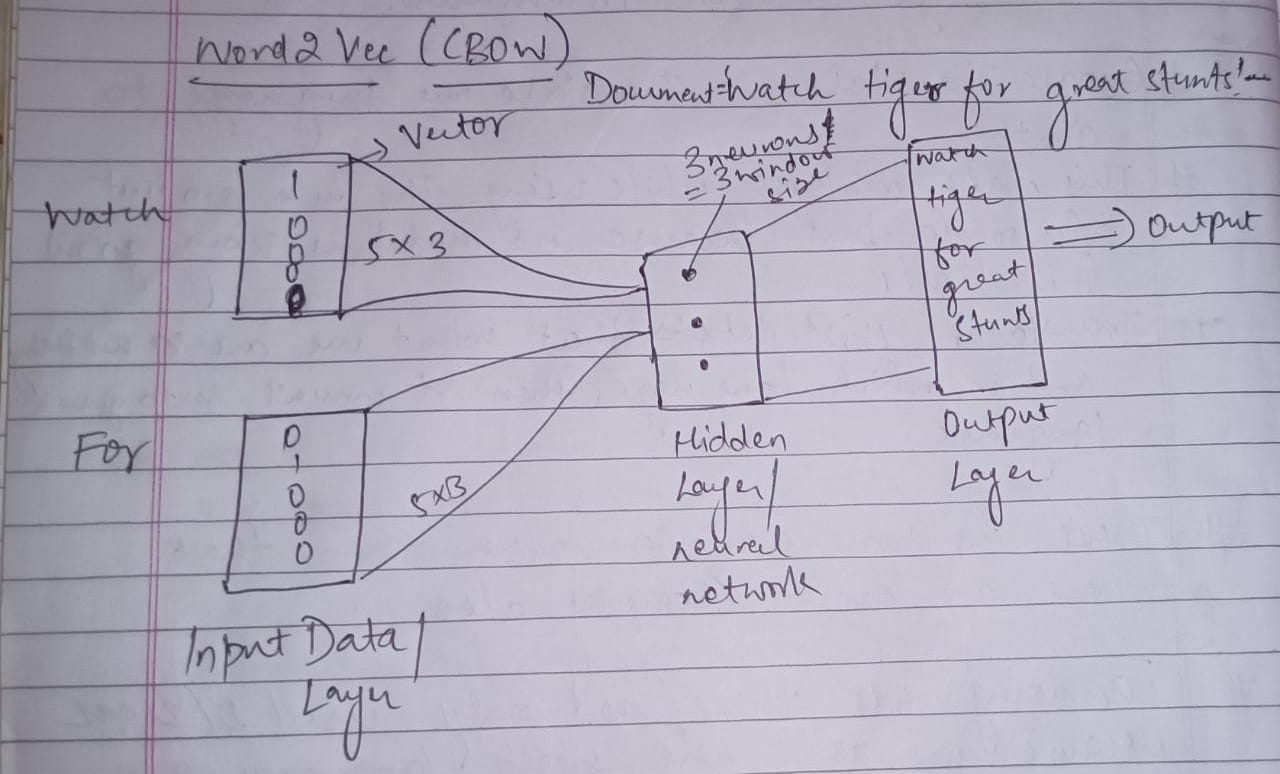
\* We have to define the window size in it. Let’s suppose we take window size = 3. So, it makes window of 3 words like watch tiger for. Watch and for becomes the independent variable (x) and tiger becomes the target variable (y). Based on independent variable it will predict target variable.

\* In Word2Vec we say x = content and y = target.

\* Like this it will creates a dummy problem.

\* Similarly next it will take window of tiger for great. Tiger and great is independent variable and for is target variable. In this way this process is happening for whole corpus.

\* In Word2Vec DL is running.



\* In output layer which word having maximum probability, it will predict that word as output.

\* Working of this process is same as neural networks, such as using of optimizer, loss function, forward & backward propagation.

\* In this way CBOW is working. Based on 2 variables it will predict the third variable.

\* Suppose we create a model using CBOW and provide the entire book 'Game of Thrones' as input. Then, we ask the model to give similar words of king then it will fetch the closest words to king from vocabulary, using cosine similarity.

Advantages: -

\* It will help to identify relation between words based on cosine similarity. It will also assist in determining which word is close to another word.

(ii) Skip-Gram

\* Skip-Gram is exactly opposite of CBOW.

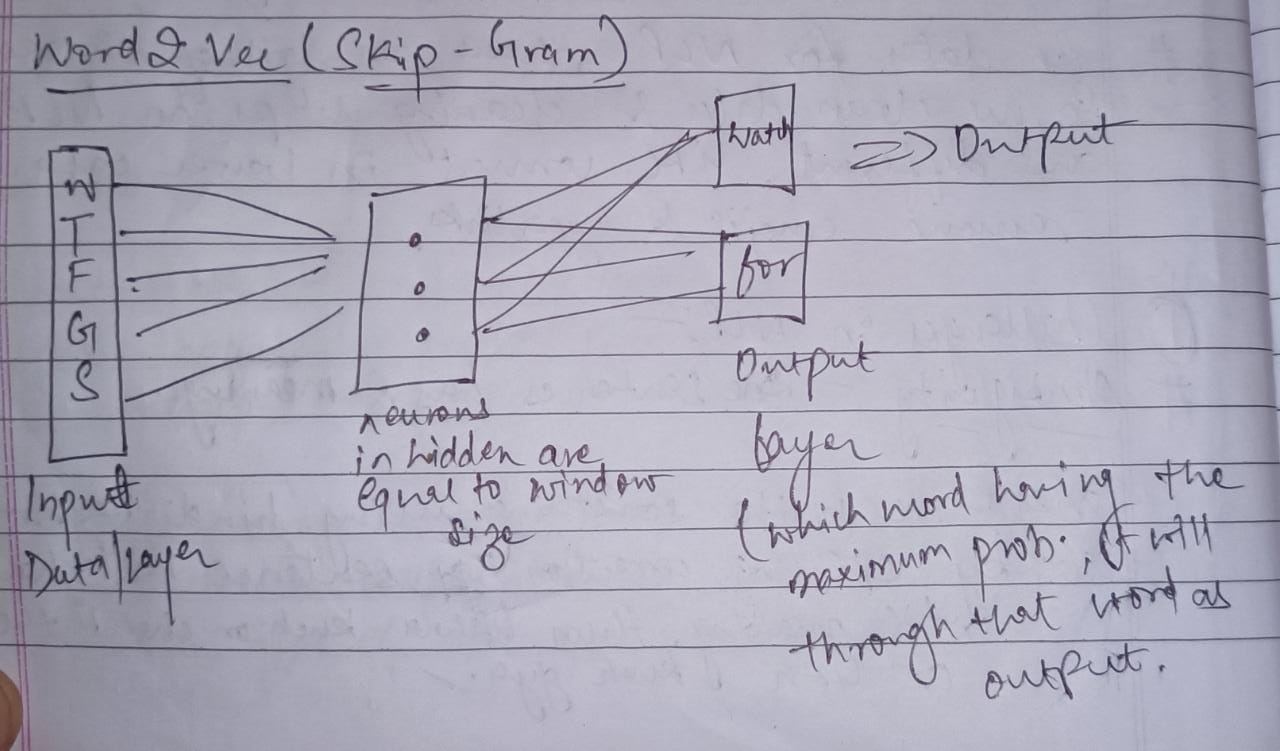
\* Functioning of Skip-Gram: -

Document = **watch tiger for great stunts.**

\* Window size = 3. It will create window of: - watch tiger for, tiger for stunts, for great stunts.

\* In first window tiger become the independent variable (x) and watch for becomes target variable (y).

\* Neural network of Skip-Gram is the mirror of CBOW.



\* Sometimes, the accuracy of Word2Vec (CBOW & Skip-grams) does not turn out to be satisfactory, and you may find yourself struggling with it. In that case increase the hidden layer, window size, increase the training (number of epochs).

**\* This are feature engineering techniques.**