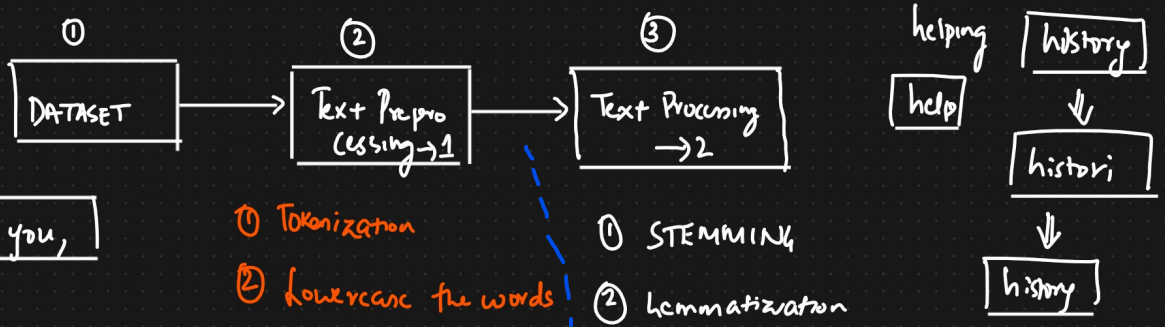
NLP In Machine Learning

Until now Data set was in the form of numerical or categorical data.
What if dataset in terms of text. Then use NLP where we will be converting text into vectors.

Below is the NLP workflow



the, he, she, Of, is, you,

↓

Sententiment Analysis

Sentiment Analysis is one of the use case of NLP using machine learning

1. Tokenization
2. Lowercase the words
3. Regular Expression

4. Text → Vectors

1. STEMMING
2. Lemmatization
3. STOPWORDS

Text Preprocessing 2 contains all these parts.

Whereas, left section represents the subparts of the Text Preprocessing 1

Below are different approaches that can be used for Text to Vector conversion

1. One hot Encoding
2. Bag of words (BOW)
3. Tf-Idf

⇒ ML

4. Word2Vec
5. Avgword2Vec

⇒ Deep Learning (Will be covered in)

- Tokenization: We will try to convert the text (sentence, paragraph etc.) into words.

- Why we need to lower case all the words? Just a standard so that for ex "trim" and "TRIM" are not treated as 2 different words.

- Regular Expression: Used for pattern matching.

- Stemming: It is the process where we will stemp or cut off the original word in such a way that we obtain the root word from it.
For Ex;
"helping" and "help" are same word so using stemming we will stemp (cutt off) the word "helping" such that we get the root stem that is "help". In this way stemming helps in reducing the redundancy of the similar words by converting them into root word.

- Both Stemming and lemmatization are the text normalization techniques. Please refer to Lemmitization_and_Stemming_diff.png present in the same folder to understand the difference b/w Stemming and Lemmatization.
- Stopwords: when performing sentimental stop words will not play important role. STOPwords has list of similar kinds of words which can be removed from the sentence for the sentimental analysis kind of use cases.
Eg; the, he, she, of, is, you etc.

Tokenization

Topics

1) Corpus ⟶ Paragraph
2) Documents ⟶ Sentences
3) Vocabulary ⟶ unique words
4) Words

Corpus

"My name is KRISH and I have a interest in teaching ML, NLP and DL. I am also a good human".

↓ Tokens

1. My name is KRISH and I have an interest in " " ' ' .
2. I am also a good human.

Vocabulary size = 18 words.

In the example the one that are encircled in orange represents unique words which are equivalent to the vocabulary.

1) One hot Encoding

In one hot encodeing: D1, D2, D3 represents documents. 1 represents +ve sentiment and 0 -ve represents sentiment

Text	O/p
D1 The food is good	1
D2 The food is bad	0
D3 Pizza is Amazing	1

O/P represents the sentiment of the respective sentence or document.
If 1 then +ve sentiment and if 0 then -ve sentiment.

Sentiment Analysis Problem

	the	food	is	good	bad	pizza	amazing
D1	1	1	1	1	0	0	0
D2	1	1	1	0	1	0	0
D3	1	0	1	0	0	1	1

vocabulary

Preparing the vectors of all the unique words(vocabulary) , This will be used for making vector representation for each of the documents.

Test [Burger is Bad] X

D1 [[1 0 0 0 0 0 0],
The [0 1 0 0 0 0 0],
food [0 0 1 0 0 0 0],
is [0 0 0 1 0 0 0]]
4x7
good

D2 [[1 0 0 0 0 0 0],
[0 1 0 0 0 0 0],
4x7 [0 0 1 0 0 0 0],
[0 0 0 0 1 0 0]]

D3 [[0 0 0 0 0 1 0],
[0 0 1 0 0 0 0],
3x7 [0 0 0 0 0 0 1]]

Above we have converted the documents into respective vector values using the vector mapping formed using vocabulary

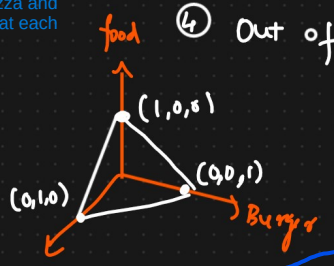
Advantages

1. Easy to Implement with python
   [Sklearn OneHotEncoder]

Disadvantages

1. Sparse Matrix → Overfitting
   We are basically creating sparse matrix(matrix with lot of 1 and 0) which will need lot of space in RAM and will lead to overfitting.
2. ML Algorithms → Fixed Size I/p.
   ML algorithm works with fixed Input size but here Input size is not fixed. For Ex; D3 have only 3 words meaning 3 i/p while D1 and D2 have 4 i/p. Hence i/p size is not fixed.
3. No semantic is getting captured
   Example
   For ex we want to capture semantic/relationship b/w 3 words: food, pizza and burger. If we plot the respective sparse matrix then we can observe that each of these are equidistant from each other representing no or constant relationship

food	pizza	burger
1	0	0
0	1	0
0	0	1



Relationship b/w 2 words is not getting captured that's why Semantics not getting captured.

4. Out of Vocabulary (OOV)
   No Relation
   During testing or performing prediction over new dataset if we are provided with document containing vocabulary that were present in the set of vocabulary used while training the model then the condition that will arise is called OOV (out of vocabulary)

2. Bag of Words (BOW)

Dataset

Text	O/P
good boy	1
She is a good girl	1
Boy And girl are good	1

O/P represents the sentiment of the respective sentence or document.
If 1 then +ve sentiment and if 0 then -ve sentiment.

Bag of words can be used to resolve the disadvantages raised in One hot encoding approach. By using Small case conversion and Stopwords we are lowering the dimensions.
In Binary BOW : have only 0 and 1.

Lowercase all the words

Stopwords

S1 → good boy boy
S2 → good girl
S3 → boy girl good

Sentences formed by lowering the case followed by removing the stop word of respective Text or Document

Vocabulary	frequency
good	3
boy	2
girl	2

	f1 [good]	f2 boy	f3 girl	O/p
S1	1	1	0	1
S2	1	0	1	1
S3	1	1	1	1

Binary Bow

In above after converting text into vector using vocabulary mapping we can clearly observe that S1, S2 and S3 have more 1s as compared to 0s. This is the reason why their respective output is also 1

Advantages

1. Simple and Intuitive (This advantage is common for all the approaches)
2. Fixed Ip Size ⇒ ML Algorithms.

Disadvantages

1. Sparse Matrix → overfitting
2. Out of vocabulary (OOV Problem)
3. Semantic meaning not there.

- The food is good → [1 1 1 0 1] → v1
- The food is not good → [1 1 1 1 1] → v2

N-grams

Eg: unigram, bigram, trigram

S1 → The food is good

S2 → The food is not good

→

food	not	good
1	0	1
1	1	1

Unigram (1,1)

↳ W
(1,1)

Bigram (1,2) Trigram (1,3)

	food	not	good
S1	1	0	1
S2	1	1	1

Unigram

+

food not	not good	food good
0	0	1
1	1	0

Combination of 2 vocabulary at a time

= Bigram

Sklearn → n-gram = (1,1) → unigrams

(1,2) → unigrams, bigrams

(1,3) → unigrams, bigrams, trigrams

(2,3) → bigram, trigram

(2,2) → Bigram.

③ TF-IDF [Term frequency – Inverse Document Frequency].

In TF-IDF, based on the frequency of word we will decide word is important or not.

S1 → good boy

S2 → good girl

S3 → boy girl good

Term frequency = (TF) = $\frac{\text{No. of rep of words in sentence}}{\text{No. of words in sentences}}$ (reptation)

IDF = $\log_e\left(\frac{\text{No. of Sentences}}{\text{No. of sentences containing the word}}\right)$ (log with base e)

Term frequency

	S1	S2	S3
good	1/2	1/2	1/3
boy	1/2	0	1/3
girl	0	1/2	1/3.

IDF

IDF for good is 0 means it is not important word since it is present in all the sentences.

Words	
good	$\log_e(3/3) = 0$
boy	$\log_e(3/2)$
girl	$\log_e(3/2)$

Final TF-IDF

Final TF-IDF obtained by multiplying respective TF and IDF values. That is first IDF value is multiplied with all the elements of first column in TF .... so on.

	f1 good	f2 boy	f3 girl	O/P
Sent 1	0	$\frac{1}{2}\log_e(3/2)$	0	1
Sent 2	0	0	$\frac{1}{2}\log_e(3/2)$	1
Sent 3	0	$\frac{1}{3}\log_e(3/2)$	$\frac{1}{3}\log_e(3/2)$	1

May be final output is representing the sentiment of the respective sentence based on OR logical gate.

Advantages

1. Intuitive (Simple and intuitive)
2. Fixed Sized I/p → Vocab Size
3. Word Importance is getting Captured

Disadvantages

1. OOV
2. Sparsity is still exists

⇓

Word2Vec

Problem of sparsity can be resolved by using Word2Vec technique for the conversion of Text into Vector. This will be discussed in the deep learning sessions.