Natural language processing (NLP)

↳ To make machine learn natural language.

→ Real World Applications of NLP

↓

→ Contextual Advertisement
→ Email Clients - spam filtering, smart reply.
→ Social Media - removing adult content, opinion mining.
→ Search Engine
→ Chatbots.

→ Common NLP Tasks

↳ Text or Document classification
→ Sentiment Analysis.
→ Information Retrieval.
→ Parts of speech Tagging.
→ Language Detection and machine translation.
→ Conversational Agents.
→ Knowledge graph and QA Systems.
→ Text Summarization.
→ Topic Modelling.
→ Text Generation.
→ Spell checking and grammer Correction.
→ Text Parsing and speech to text

→ Approaches to NLP:
  ↳ Heuristic Methods.
  → ML based Models or Methods.
  → DL based Methods.

→ Heuristic Approach
  ↳ Examples:
    → Regular Expression
      ↙
Can help remove html tags from scrapped text.
or it can also help to search some specific words.

→ Wordnet
  ↳ Stores the relationship of word with other words.
    ↳ i.e: Run ↔ jogg
      ↕
      Shoe ⇒ (you need shoes while running).

→ Open mind Common Sense
  ↳ you can store Common Sense facts from all over the world here.

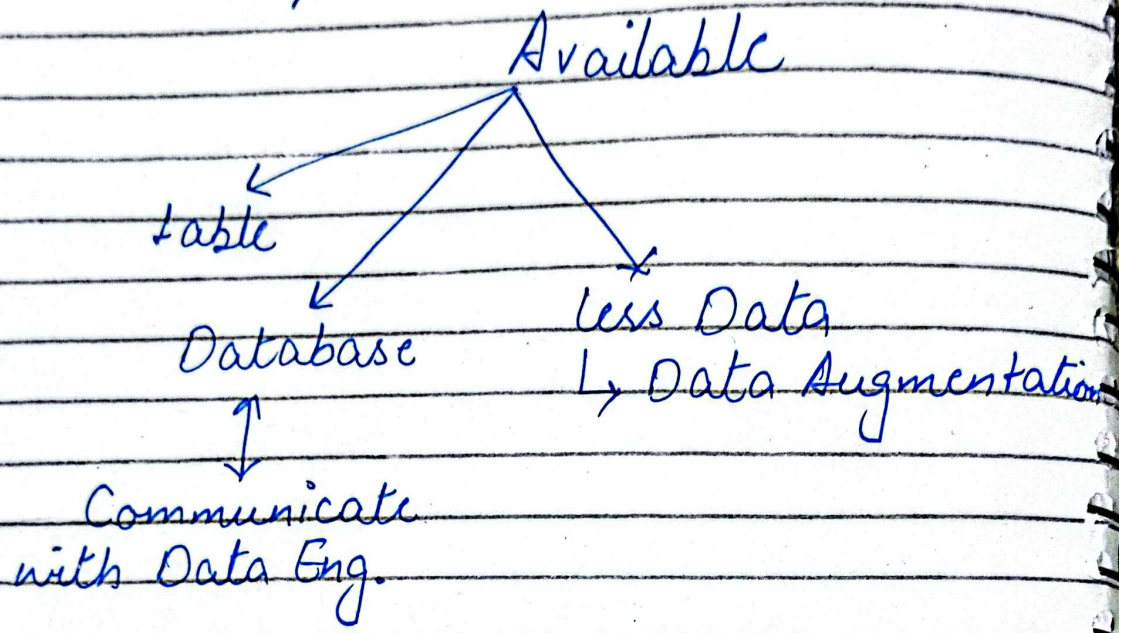
→ Advantages:
  ↳ you get quick response
  → Less error

→ NLP (Pipeline)
  L, Set of steps followed to build an end to end NLP software.

NLP software consists of following steps:
  L, Data Acquisition.
  → Text preparation.
    L, Text cleanup
    → Basic Preprocessing
    → Advance Preprocessing
  → Feature Engineering
    L, Converting text into nums.
  → Modelling
    L, Applying actual Algorithm
      L, Model Building.
      → Evaluation.
  → Deployment
    L, Deploy
    → Monitoring
    → Model update

Data Acquisition:

→ Available:
  ↳ on the table
→ Others (External Data).
→ Nobody.



Others
  ↳ Public Dataset (Kaggle)
→ Web Scrapping (Beautiful Soup).
→ API → Rapid API → request → Data in JSON
→ Pdf → use Python libraries.
→ image → use Python libraries
→ Audio → audio to text using Python libraries.

Text Preparation:
- Cleaning
  - Basic preprocessing
  - Advance preprocessing

→ Cleaning Text:
- Cleaning Emojis is called unicodenormalization
  - Converting emojis into machine language

→ Checking spelling:

```
incorrect = "----"

from textblob import TextBlob
txt_blb = TextBlob(incorrect)
txt_blb.correct()
```

—.

→ Basic preprocessing
- Tokenization => Basic
  - Sentence
  - Words

=> Optional
- Stopword removal
- Stemming
- Removing punctuations, digits
- Lowercasing or uppercasing
- Language detection.

→ Tokenization:
↳ Converting text into tokens it could be a sentence or word.

```
from nltk.tokenize import
                    sent_tokenize,
                    word_tokenize
```

→ Optional:
↳ Stop word removal:
We use some words in english whose task is to form sentence but they don't have any contribution in meaning i.e.: a, an, for, the etc. So if we are making NLP application from where we want to derive a context or meaning so there these stop words will not help and we remove them.

→ Stemming
If we have different words for the same meaning i.e.: dance, dancing and danced, so we can bring these words in root form.
their

Diff b.w Stemming and lemitization?

→ Lowercasing or uppercasing the txt:
  ↳ So that our text remains consistent.

→ Advance Preprocessing
  ↳ POS tagging (Parts of speech)
  → Parsing
  → Co-refrence resolution

=> POS tagging can only be done if we have not use Stop "word removal"

Note: There is a technique in Deep learning that is mostly used now days called Transfer learning.
  ↳ you take advance DL model that is trained on very Big data. i.e: BERT transformer.
  ↓
  it is trained with 40GB of data already.

→ Text Pre-processing
(with Code implementation)

① Lowercasing:
Suppose if we have data then
↳ data.iloc[3,0].lower()

② Remove HTML Tags:
↳ Do it by using regex

```
def remove_html(text):
    Pattern = re.compile('<.*?>')
    return Pattern.sub(r'', text)

clean_text = lower_case.apply(remove_
                              html)
Print(clean_text).
```

③ Remove URL's:

```
def remove_url(text):
    Pattern = re.compile(r'hHPs?://\S+
                         |www\.\S+')
    return Pattern.sub(r'', text)
```

(4) Chat word treatment:

```
chat_words = {
    'ASAP' : 'As Soon as Possible',
    'FYI' : 'For Your Information',
    ____ ____
    ___ ___
    ____ ___
}

def chat_conver(text):
    new_text = []
    for w in text.split():
        if w.upper in chat_words:
            new_text.append(chatwords
                            [w.upper()])
        else:
            new_text.append(w)
    return " ".join(new_text)
```

(5) Removing Stop words
↳ Do it if you are doing
↳ Document Analysis
↳ Sentiment Analysis
→ Use NLTK Library

a the of are

But you don't remove them if you want to do POS (Parts of speech tagging).

```
def remove_stp(text):
    new_text = []
    for w in text.split():
        if w in stopwords.words('english'):
            new_text.append('')
        else:
            new_text.append(w)

    x = new_text[:]
    new_text.clear()

    return " ".join(x)
```

(6) Handling Emojis
→ 2 options
↳ remove emoji
→ replace emoji with its meaning

(7) Tokenization:

We can do it by using:
→ Split function
→ regular expression

But they might have some problems.
→ Best way is to use libraries
use NLTK:

```
from nltk.tokenize import
                word_tokenize,
                sent_tokenize
```

These 2 functions have internal algorithms that are ready to handle situations like:
→ Prefix
→ Suffix
→ Infix
→ Exception

→ Another Best tokenization technique is given by Spacy library it is better than nltk.

```
import spacy
nlp = spacy.load('en_core_web_sm')

sent = "We've here to help! mail us
        at nks@gmail.com".

d = nlp(sent)

for i in d:
    Print(i)
```

(8) Stemming:

- walk → Stem
- walking
- walks
- walked

Majorly used when are are making IR (information retrieval) system.
↓ ↳ use nltk library
i.e: Google ↳ we'll use porter stemmer in this library. ↓ Algorithm

Snow ball Stemmer
↳ for other Languge

(9) Lemmatization → It searches (slow)

↳ Same as Stemming But it will take only specific Language word as root word.

Lemmatization is slower than stemming as it tries to find the actual word of a Language. (Use it if you want to show the output to user.

We use Wordnet lematizer

↳ it has stored the english language words that are related to eachother.

Feature Extraction (In NLP)

↳ we extract features from text after converting text into numbers.

→ Core Idea? (Goal)

↳ Whenever you are converting from text to numbers then this conversion or set of numbers should tell the meaning (Semantic) of something.

↓
hidden meaning should be convey through numbers.

→ Techniques (for converting txt to num) (with semantic meaning)

↳ → One hot Encoding
→ Bag of words
→ n-grams
→ Tf Idf
→ Custom features
→ Word 2 Vec {Embedding}

→ Common Terms

↳ → Corpus
→ Vocabulary
→ Document
→ Word

Corpus => Combination every word in whole dataset. (C)
Vocabulary => combination of every unique word in corpus. (V)

Document => Individual block in a feature or column. (D)

Word => Individual word in a document.

→ One-hot Encoding

D1 People watch movie
D2 rafay watch movie
D3 People write letter
D4 rafay write letter

Corpus => People watch movie rafay watch movie people write letter rafay write letter

Vocabulary => Corpus کا (کتنے لفظوں کو ملا کر بنا ہے)

People watch movee rafay letter
V = 5
Now OHE will make 5 features of every unique word.

write

	People	watch	movie	rafay	letter
D_1	1	1	0	1	0
D_2	0	1	0	1	0
D_3	1	0	1	0	1
D_4	0	0	1	0	1

$D_1 = [[1\ 0\ 0\ 0\ 0\ 0],$
$[0\ 1\ 0\ 0\ 0\ 0],$
$[0\ 0\ 0\ 1\ 0\ 0]]$

$D_2 = [[0\ 0\ 0\ 0\ 1\ 0],$
$[0\ 1\ 0\ 0\ 0\ 0],$
$[0\ 0\ 0\ 1\ 0\ 0]]$

$D_3 = [[1\ 0\ 0\ 0\ 0\ 0],$
$[0\ 0\ 1\ 0\ 0\ 0],$
$[0\ 0\ 0\ 0\ 0\ 1]]$

$D_4 = [[0\ 0\ 0\ 0\ 1\ 0],$
$[0\ 0\ 1\ 0\ 0\ 0],$
$[0\ 0\ 0\ 0\ 0\ 1]]$

One hot encoding

Pros => Intuitive
→ Easy to implement

Flaws => Sparsity => Sparse array.
causes overfitting.

→ No fixed Size.
→ OOV (out of vocabulary)
→ No Capturing of Semantic meaning.

→ Bag of Words
↳ Specially used when we are working on text classification problem. i.e→ News Document
↳ Dividing it in some categories like: Sport news, Political news etc.

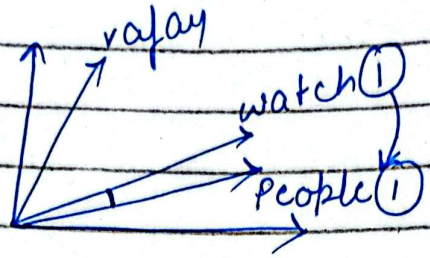
D1 People watch rafay
D2 rafay watch rafay
D3 People write comment
D4 rafay write comment

BOW:

	People	watch	rafay	write	comment
D1	1	1	1	0	0
D2	0	1	2	0	0
D3	1	0	0	1	1
D4	0	0	1	1	1

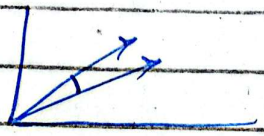
Suppose if we have only 3 features then:

People watch rafay



Giving Sentence a vector form then checking the angle with the closest one.

→ Pros
  ↳ Simple and intuitive
→ Disadvantages
  ↳ Sparsity
  → OOV
  → un Order
→ This is a good movie
  this is not a good movie

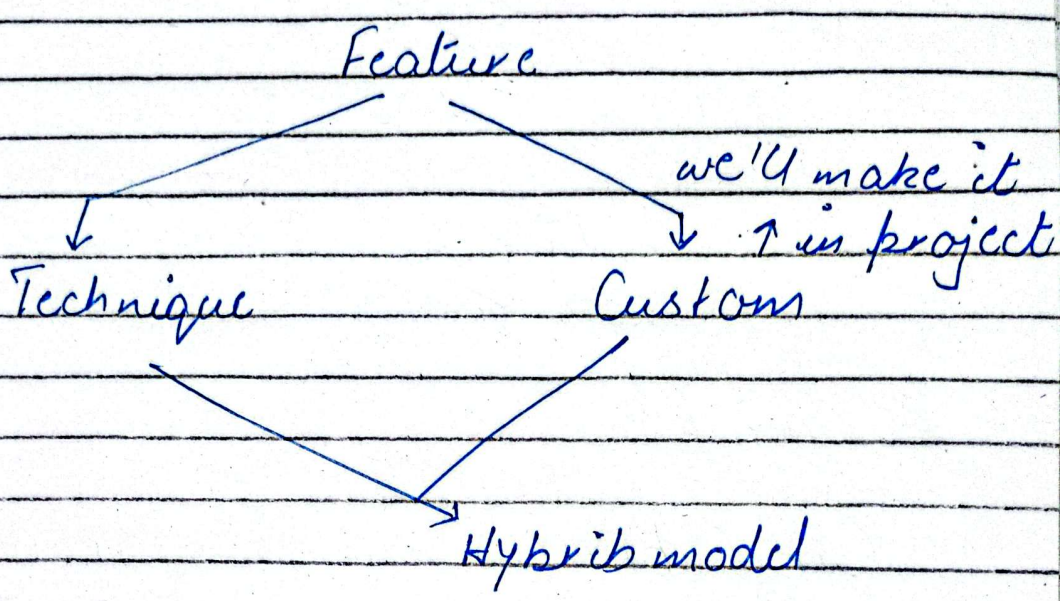


→ Bag of n-grams

Pros
  ↳ Able to capture Semantic meaning.
  → Easy implementation → OOV

Cons
  ↳ As we increase n-grams the dimension of vector also increases due to which machine takes time to learn.

→ Custom Feature :



→ Word Embeddings
- ↳ BOW
  - ↳ grams
  - → Tf Idf

> Frequency Base

In NLP, word embedding is a term used for the representation of words for text analysis, typically in the form of a real valued vector that encodes the meaning of words that are closer in vector space are expected to be similar in meaning.

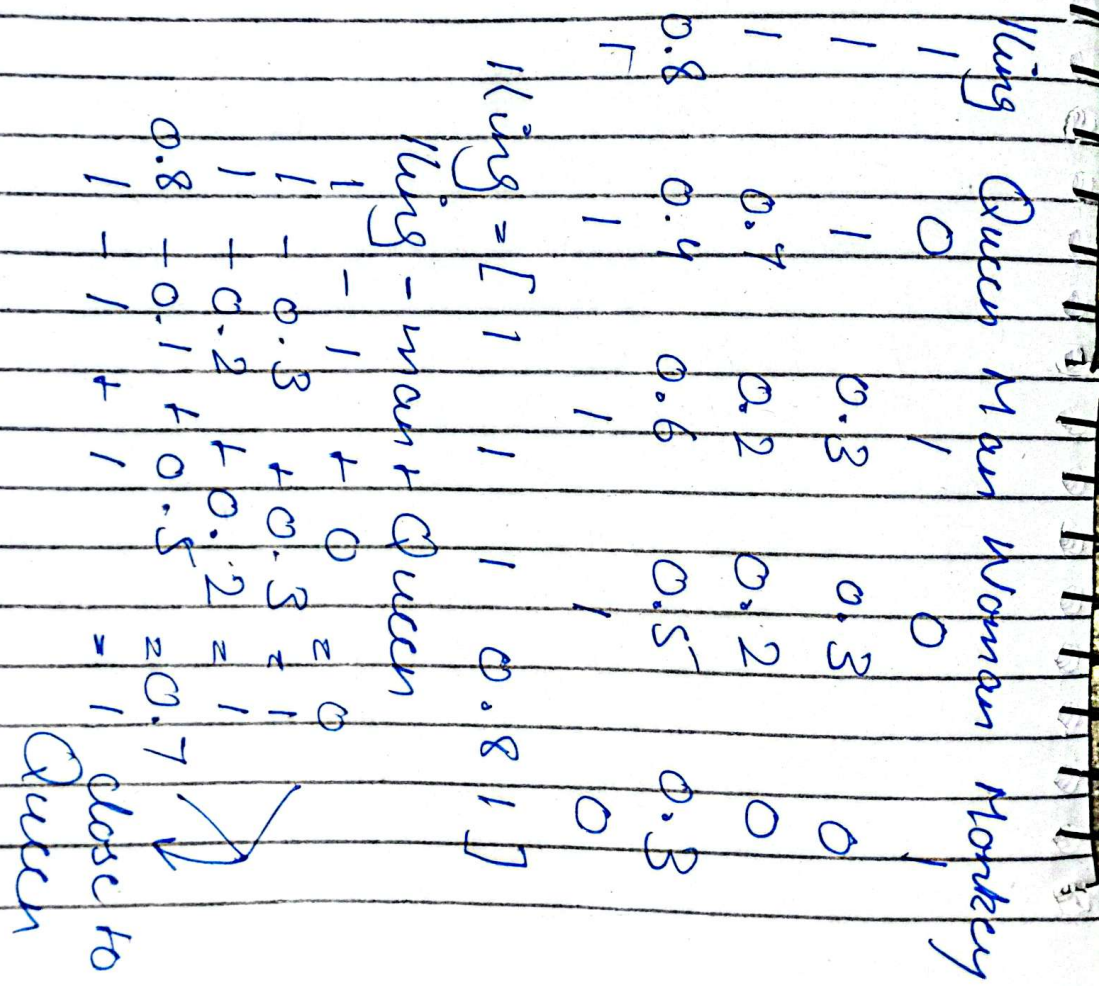
↳ In Short => Converting words into vector

→ 2 types => Frequency based
Prediction based

Word 2 Vec (uses DL)
↳ Prediction base word embedding
similarity
Can see (difference) between
words that have same meaning
i.e.: happy
joy

Advantages

→ Can Capture Semantic meaning.
→ Improve sparsity (Compute fast).
→ Dense vector. (most words are
non zero).

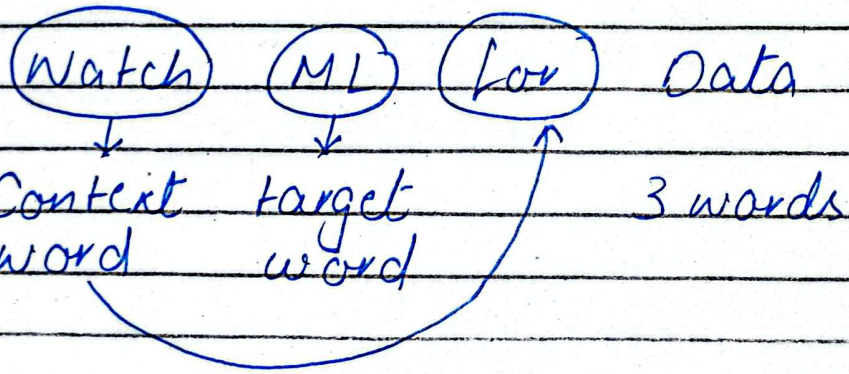


→ Types of Word 2 Vec

CBOW Skip gram

→ CBOW (Continuous Bag of words)

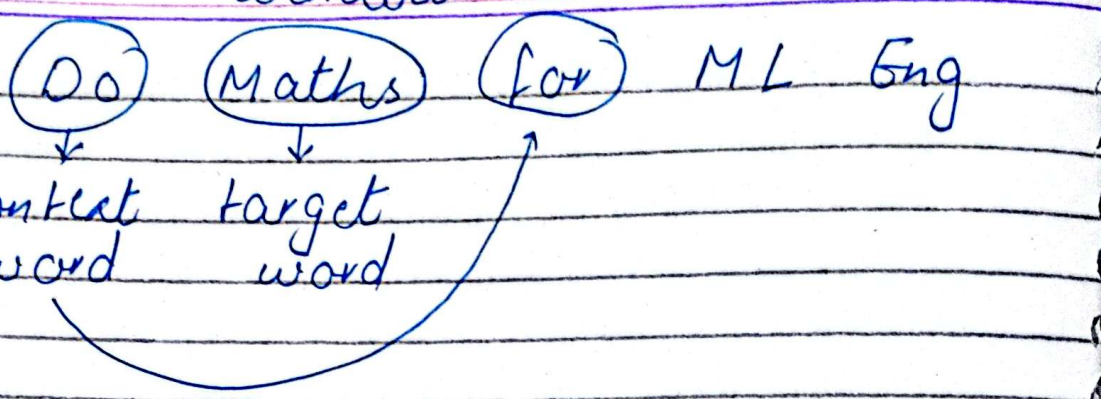
fake problem
↳ Solve
↳ Vector
↳ by Product



X	for	Y
watch, (ML)		ML
ML, Data		for
for, Science		Data

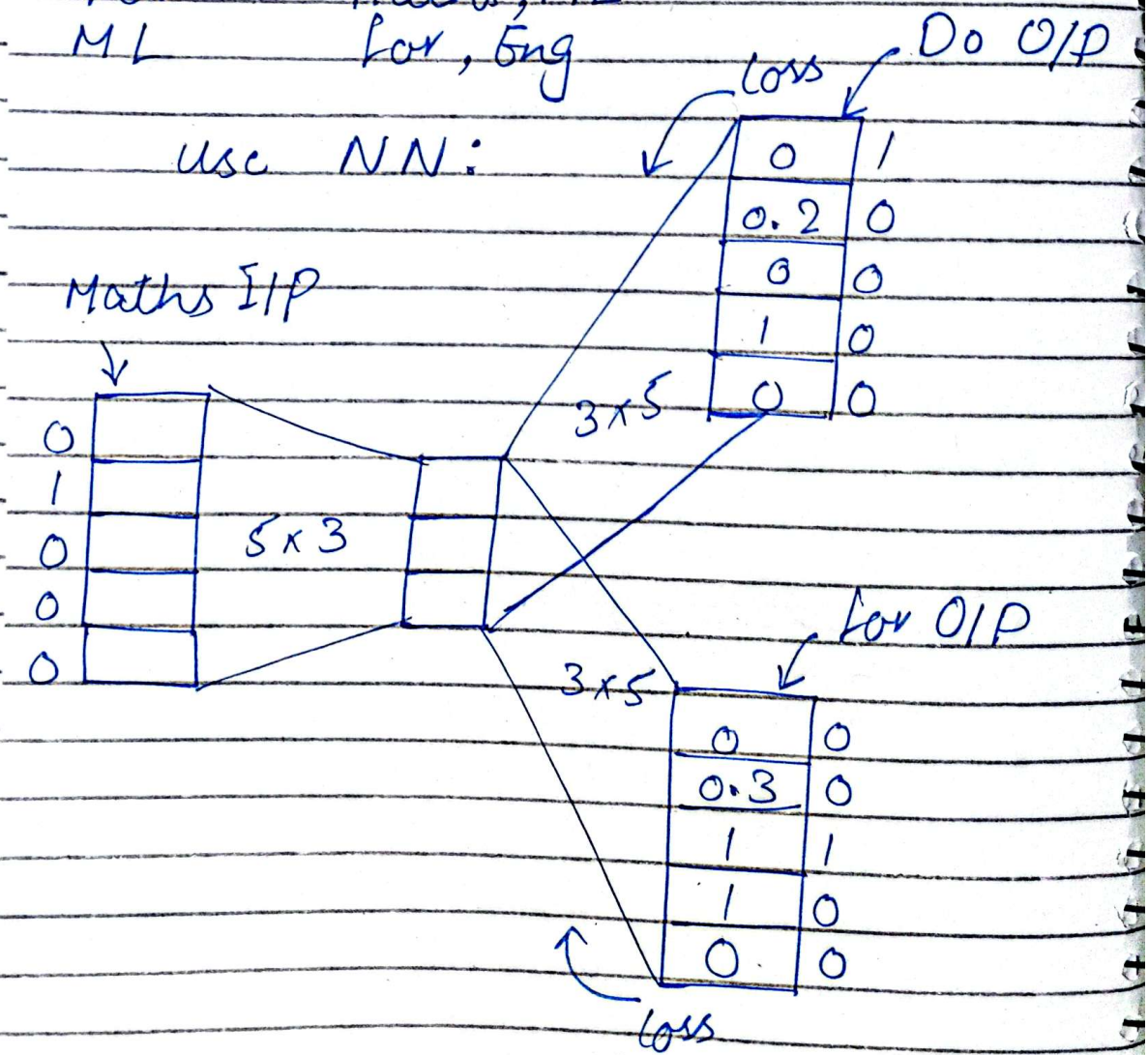
→ Skip gram (Opposite of CBOW)
↳ given target words you have to predict the context word (Dumny Problem).

window = 3



X	Y
Maths	Do, for
for	Maths, ML
ML	for, Eng

use NN:



→ When to use CBOW / Skip gram

Small data
↳ CBOW

Large data
↳ Skip gram

→ How to improve Word 2 Vec
↳ Increase training data.
→ increase dimension of hL vector
→ increase window size for words.

→ Training our own model (using Word 2 Vec)
↳ by gensim lib

→ Text Classification:
  ↳ Types
    ↳ Binary Classification
    → multi class classification.
    → multi label classification
      ↳ For multiple input you'll have multiple output.
      ↳ example => one particular news can come under multiple news

→ Applications
  ↳ Gmail Spam Classification.
  → Customer support.
  → Sentiment Analysis.
  → Language Detection.
  → Fake news Detection.

→ Pipeline
  → Data Acquisition
  → Text Preprocessing
  → Feature Extracting
  → Modelling
  → Evaluation

→ Different approaches to do text classification

- Heuristic approach
- APIS
- ML
- DL

→ APIS:

- Amazon and Google have their own cloud platform that have built solutions for most frequently occuring problem.