NLP

Start with Example Use case with define problem → then solve by NLP

(1) How Represent the language in NLP ← Explore → by Embedding

eg:- Translation
(2) Text Generation
(3) text Summarization
(4) Speech to text or Text to speech

eg :- Sentiment Analysis

• Swiggy Review Analysis example code

ML

(1) • Clean text → Lower upper, special char Remove
(2) • Tokenization → create list of words
↳ use NLTK
↳ word tokenize
↳ Stopword Removal (corpus from
↳ Lemmatization (Root word
↳ Stem

(3) How ML Models work
(1) ↳ Intro ML
(2) ↳ Convert words / Text Representation
(3)(4) BOW (Bag of words / (5) TFidf → convert sentences into number
↳ CountVectorizer (sklearn

eg:- John likes to watch movies to. → sent1
Tom likes.

	John	likes	to	watch	movies	too	like
sent1	1	2	1	1	1	0	0
sent2	0	0	0	0	0	1	1

(2) TFIDF

$$TF(t,d) = \frac{\text{count of term } t \text{ in doc } d}{\text{total terms in doc } d}$$

$$IDF(t,D) = \log\left(\frac{\text{total docs in Corpus } |D|}{\text{num of docs containing term } t}\right)$$

John = 1/6
to = 2/6

John → 2 sentences/docs
log(2/1) → 1 time John

TF-IDF Score:

$$TFIDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

• ONE HOT REPRESENTATION OF WORDS.
• One Model from sklearn by MultinomialNB
Logistic, Random, SVM

↳ Now Start Predict Review is positive or Negative. ✓

→ 1st solve the
→ Solve NLP without any Specific Model like (RNN LSTM)
(1) Solve NLP using basic ML
(2) then go NLP → Embedding ↳ Encode the words ↳ array like
(3)

• R.

ML में features का sequence important नहीं होता है but in NLP word sequence matter

eg: pizza or pizzas (same na

sklearn.feature_extraction.text
import CountVectorizer
" TfidfVectorizer

① How Represent words .

Ⓐ Use One Hot Representation of words

eg Cat → [0,0,1]
Dog → [0,1,0]
Giraffe → [1,0,0]

What issue there.
① Millions of words
② dog or dogs same but here differs (no make similarity meaning)
③ impossible vector – infinite size one Hot)
④ Also word meaning depends on whole sentence context.

↳ Use **Word Vectors** : Representation of word → A dense vector is built such that word vectors of words appearing in it's context are similar to this word vector.

→ 2 major types → ① Word2Vec & GLOVE

① **Word2Vec**

Meaning ← The words appearing around a word defines the word.

- we have extensive text corpus, serving as a collection of words (means → जितना word आप दो उतना ही meaning समझेगा)
- Each word within this text corpus needs to be donoted by vectors.
- We Identify a center word c and its surrounding context words. (window)
- By leveraging the similarity b/w vectors representing c & o.

· from gensim.models import Word2Vec

w = Word2Vec (sentences = tokenword, vector size = 100 (→ Dimension of one word), window = 5 (1 cat, 2 Left, 2 Right), min_count = 1, workers = 4)
↳ for Parallelization for process

W.WV['food']
↳ Get the vector representation of a word (food)

W2V
→ It is also Model like other ML

· 100 Dimension

Similar words means → probability occuring together is higher

, Window size not so much large or small.

⇒ Loss func / objective function → $J(\theta)$

T = total no. of words

$= \frac{1}{T} \log(\text{Likelihood})$ ↗ तो इसे Maximize करना है तो कैसे करेंगे ↓ तो हम इसके Mini को -ve कर देंगे

$= \frac{-1}{T} \log \left(\prod_{t=1} \prod_{j=-m,\, j \neq 0}^{m} P(w_{t+j} \mid w_t, \theta) \right)$

| Likelihood ↳ आटा से Bakery & then Serve कैसे की Probability क्या है तो हम Likelihood से देखते हैं

obj. func $= \frac{-1}{T} \sum_{t=1}^{T} \sum_{j=-m \text{ to } m,\, j \neq 0} \left[\log P(w_{t+j} \mid w_t, \theta) \right]$ → ये find करना पड़ता है

② Glove embedding

```
df = pd.read_csv('/content/glove.6B.50d.txt', sep=" ", quoting=3, header=None, index_col=0)
```

- `glove_vectors = {key: val.values for key, val in df.T.items()}`

→ It is pretrained ek already → So load and use.

Reduce dimensionality using PCA or T-SNE

- numpy, matplotlib
- from sklearn.decomposition import PCA
- from sklearn.manifold import TSNE

↑ for Visual of 50D into Glove into 2D.

- means हमें PCA से through वो 2 Dimension चाहिए जो सबसे ज्यादा Variance Cover करता है वो minimum error deta है।

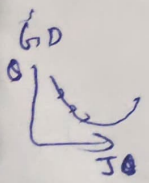
→ Cosine-similarity

Word2Vec Continue :-

$P(w_{t+j} | w_t)$

- कभी word outside में होगा तो कभी वो center में होगा तो इसे time दोनो की Probability different-different होगा इसलिए दोनो में P calculate कर रहे हैं।

$J(\theta)$ - minimize this



$P(o|c)$ → u_o, v_c

$= \frac{\exp(u_o v_c)}{\sum_i \exp u_i v_c}$ ← $u_o \cdot v_c$ ↑ Similarity b/w the 2 Vectors.

eg : Sigmoid $\sigma(x_i) = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$

$P(o|c) = \frac{\exp(u_o \cdot v_c)}{\sum \exp(u_j \cdot v_c)}$ → Similarity Prob

o & c are occurring together such that 'o' is an outside word & 'c' is a centre word

u_o → o^{th} word

$\sum$ → इसका word की Prob based on c word

→ Normalization term.

Here

→ Train a binary logistic reg Model

- Positive Pair (o & c when are occuring together
- vs
- Noise pair (o & c randomly chosen.

- 2 kinds of Word2Vec
  - ① SG (Skip-Gram) ← more use ← more fine — $P(o|c)$
  - ② CBOW (continuous Bag of words) $P(c|o)$ ← Less time & use for Less frequent words in doc
- which one should use

So

$$J(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^{k} E_{j \sim P(w)} [\log \sigma(u_j^T v_c)]$$

noise pairs

expectation → k times draw a Sample from word Prob distn

⇓

So reduce -ve to maximize

↳ So called

→ Negative Sampling method.

eg $V = 1000$ words, $k = 5$

How to decide k-Value by using Unigram

$P(w) = \frac{U(w)^{3/4}}{Z}$ ← Normalization ⇒ why 3/4 · bcz rare words have Less Prob of being drawn from distribution

1:05 change here by NLSM to reduce Complexity (Compute)

$\frac{\exp(u_o^T v_c)}{\sum \exp(u_j^T v_c)}$ - True pair

Issues all pairs with C by k sample

$$\log[\sigma(u_o^T v_c)] + \sum_{i=1}^{k} [\log \sigma(-u_k^T v_c)]$$

True Pair

$$\left\{ \sum_{i=1}^{k} E_{j \sim P(w)} [\log \sigma(-u_j^T v_c)] \right\}$$

k sample ↳ Noise Pair

Language Modeling:

if word Bat → was, Object, Verb, Adj, Noun.

then

→ vector =

→ कितना Meaning में कितना we हो रहा है उसी Content मे यहाँ show करेगा।

what is the word x_5

given $x_1\, x_2\, x_3\, x_4 \ldots \to x_5 \to x_6 \to x_7$

Give all previous words in a text
Predict next occuring word.

eg:- Students open their ____ → books, notebook, Pens, Assignment

eg:- google keyboard, search bar prediction

→ in Probability terms

$$P(x_1, x_2 \ldots x_T) =$$

$$P(A|B) = \frac{P(AB)}{P(B)}$$

$$P(A \cdot B) = P(A|B) \cdot P(B)$$

$$P(\underbrace{x_1, x_2 \ldots}_{B} \underset{A}{x_T}) = P(x_T | x_{T-1} \ldots x_1) \cdot P(x_{T-1}, \ldots x_1)$$

$$\Downarrow$$

$$P(x_T | x_{T-1} \ldots x_1) \cdot P(x_{T-1} | x_{T-2} \ldots x_1) \cdot P(x_{T-2} \ldots x_1)$$

$$\Downarrow$$

$$\Rightarrow P(x_T | x_{T-1} \ldots x_1) \cdot P(x_{T-1} | x_{T-2} \ldots x_1) \cdot P(x_3 | x_2 x_1) \cdot P(x_2 | x_1) \cdot P(x_1)$$

π-Probability $\quad \prod_{t=1}^{T} P(x_t | x_{t-1} \ldots x_1)$

⇒ How to learn it.

① L.M (Pre Deep Learning)

use → ⓐ **n-gram L.M**:- → Very Powerful for Scratch

word occuring at step 't' is not dependent on All words before it but only on 'n-1' word sequence occuring before it.

eg:- $n=3$ $\quad x_1\, x_2\, x_3\, x_4\, x_5$ — (x_6)

4-gram model

$$P(x_t | x_{t-1} \ldots x_1) \sim P(\underbrace{x_t | x_{t-1} \ldots x_{t-n+2}}_{n-1 \text{ words before } t})$$

$P(x_6 | x_5\, x_4\, x_3\, x_2\, x_1)$

→ $P(x_6 | (x_5\, x_4\, x_3))$

for 4 ngram.

→ be deep learning can use it

N-grams?

n=1 → unigram → eg:- 'Student', 'opend', 'their', 'book'

n=2 → bigrams → "Student opend", "opend their"

n=3 → triagrams → "Student opend their"

n=4 → " -:- "Student opend their book"

b) $P(x_t \mid x_{t-1} \cdots x_1) \sim P(x_t \mid \underbrace{x_{t-1} \cdots x_{t-n+2}}_{n-1 \text{ words before } t})$

↓
How it works

$$\frac{\text{Count}(x_t\, x_{t-1} \cdots x_{t-n+2})}{\text{Count}(x_{t-1} \cdots x_{t-n+2})} \Rightarrow \frac{\text{Count(ngram)}}{\text{Count(n-1 gram)}}$$

eg:- n=4

- Students opend their books ___ on

$P(\text{"on"} \mid \text{"opend their books"})$

$$\Rightarrow \frac{\text{count("oped their books on")}}{\text{Count("oped their books")}}$$

- what is Problem in N-gram?

- Drawback

① Loosing the context of words some times when depends on other part.

eg: Examiner asked the students to Open their ___ ?

book exam assignment

Examiner asked the students to open their exam

but n-gram use most frequent word like → book

↓ to we are loosing context

②

eg: $P(\text{"exams"} \mid \text{"students to open their "})$ → So its prob is 0 but not be should be

$$\frac{\text{"students to open their exams)"}}{\text{ngram} \rightarrow \text{does not exist}}$$

↓
So do Smoothing

give some little Probability to every word n-gram seq ... is something.

(3) n-1 gram did not occurs

eg: Students to open their

अगर यहाँ Book नहीं occurs नहीं है तो

Prob of this is 0

↓

How to Solve it

↓

Using Backoff

5 gram

eg: Students to open their -

अगर 5 gram है तो → 4 gram check करेंगे वहाँ भी नहीं तो ... 3 gram

(5) Store all data

(4) What if multiple n-gram have the same count

P("book") = P("exam") = 10

↓

Solve all drawback by using DL

by ~~by~~ to DL LM

① Neural NLM

② then RNN

7th dec 2021 NLP