4-3

Word Embeddings

Nlp means all the text data

Text data can not use to develop ML model, it is very important to oocnvert

Text data to math representation

Categorical column = Numerical columns

Map, One hot encoding, np.where, label encoder these are single word columns

So, traditional machine learning are not suitable

These Techniques are word Embedding
1. Bag of words

2. Tf-Idf (term frequency – inverse document frequency)

3. Word2 vec

Bag of words:

Vocabulary → Dictionary of words

Every use case data set has its own vocabulary, unique words with index

1. Game of Thrones is an amazing tv series
2. Game of Thornes is the best tv series
3. Game of Thornes is so great

Vocabulary = [An, Amazing, best, Game, Great, is, of, series, so, The, Thrones,

Tv]

1. Sky is nice
2. Clouds are nice
3. Sky is nice and clouds are nice

Vocabulary = [sky, is , nice, cloud, are , and]

When we apply stop word or text preprocessing vocab will change

Vocabulary = [cloud, nice, sky] → these are features

Pre-processed sentence	Cloud(f1)	Nice(f2)	Sky(f3)	Feature vector
Sky nice	0	1	1	[0,1,1]
Cloud nice	1	1	0	[1,1,0]
Sky nice cloud nice	1	2	1	[1,2,1]

N-gram:

Bi-gram or Tri gram:

Sky nice cloud nice: sky nice, nice cloud, cloud nice

This is not combination of words

Bi gram:

Pre-processed sentence	Cloud(f1)	Nice(f2)	Sky(f3)	Sky nice (f4)	Nice cloud (f5)	Cloud Nice (f6)	Feature vector
Sky nice	0	1	1	1	0	0	[0,1,1,1,0,0]
Cloud nice	1	1	0	0	0	1	[1,1,0,0,0,1]
Sky nice cloud nice	1	2	1	1	1	1	[1,2,1,1,1,1]

Advantages : is easy to implement

Disadvantage: Spares matrix [Curse of dimensionality], rare words occurrence

2. TF-IDF (Term Frequency- inverse document Frequency)

Tf*Idf

Tf = probability of words in a given sentence

$$\text{TF} = \frac{Number\ of\ times\ the\ wordsoccurence}{toal\ no\ of\ words}$$

Idf = $inverse\ document\ frequency = \log\left(\frac{number\ of\ \ documnets}{number\ of\ documnets\ have\ that\ particular\ words}\right)$

1. Good Movie
2. Good Snacks
3. Movie Snacks good

Vocabulary = [Good, Movie, snacks]

TF:

Vocab	Sen1(Good Movie)	Sen2(Good Snacks)	Sen3(Good Movie snacks)
Good	1/2	1/2	1/3
Movie	1/2	0	1/3
Snacks	0	1/2	1/3

IDF: Total document word is present (because inverse n/)

Vocab	IDF
Good	Log(3/3)
Movie	Log(3/2)
Snacks	Log(3/2)

TF*IDF:

Sentence	Good	Movie	Snack
1	½ * 0 =0	½*log(3/2)	0
2	½*0 =0	0	½*log(3/2)
3	1/3*0	1/3*log(3/2)	1/3*log(3/2)

TF-IDF vectorizer:

Practice the code in given ipynb

5-3:

Generally, if you want to develop any model

1. Read the data
2. Preprocess the data EDA
3. Develop the model
4. Predictions
5. Evaluation

NLP and DL or ML input as number

NLP use case: Text Data

Text data we need to convert into number representations

Then we can pass to the model

Preprocess of data

1. Regex
2. String Punction
3. Lower
4. Stop Word
5. Stemming

Use this data == feature vectors

The good vector always try to give the best predictions

1. BoW → count or frequency
2. Tf-idf → count or frequency
   Main drawback of above two is Sparse matrix, no Semantic relationship

Word2vec:

Developed by google with 3billion words

Every word has a vector representation 300 values

Every word has 300 dimensions vector

Features(300)	Boy(1 out of 3b)	
Royal	0.2	

Fruit	0	
Gender	0.6	
Education	0.5	
human	0.7	
animal	0.1	

Boy vector = [0.2,0,0.6,0.5,0.7,0.1.....(300 values)]

King-man+women = queen

The distance between the two words(semantic) using Euclidean distance, cosine similarity

Research and Development going to identify the good embeddings[pre trained model]

CBoW – continuous bag of words

Neurons, weights are the vectors

Cosine similarity:

$$\cos\theta = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\ \sqrt{\sum_{i=1}^{n} B_i^2}}$$

GloVe:

Practice

6-2 practice

ML: Preprcocess == Numerical data ==== > Model

DL : Preprocess == Numerical data (Pixels) === > Model

Nlp: Preprocess ==== Numerical data (Word Embeddings) == > MLModel or NN

In NLP the main focus on representation of vector features: word embeddings

1. Bag of words(Sparse- not giving the weights)
2. Tf-idf (spare – it is giving weights but relationship between words)
3. Word2vec : king – man + woman = queen

Model is already we are using that model → Transfer learning that model we are applying on our own data (fine tuning)