End Semester Exam 2020-21

Name ⇒ Himanshu Gupta
Class/Sem. ⇒ 7th Sem, 4th year
Roll No ⇒ BT17CSE093
Branch ⇒ CSE
Subject ⇒ NLP
Date ⇒ 24/11/2020

1) Lemmatisation & Stemming

(1) Lemmatisation :- It is the task of determining that two words have the same root, despite their surface differences. It takes into consideration the morphological analysis of words. Below we illustrate the method with examples in ~~both~~ two languages English & ~~Spanish~~ Sanskrit.

Language	Form	Lemma	Morphological information
English	Studies	Study	Third person, singular, present tense.
"	Studying	Study	Gerund of Verb
"	Studied	Study	Third person, singular, past tense.
Sanskrit	पठति	पठ	Third person, singular, present tense
Sanskrit	पठितवान्	पठ	~~Third~~ 2nd person, ~~singular~~ singular, present.
Sanskrit	पठत्	पठ	Third person, singular, past tense.

(2) Stemming ⇒ It work by cutting off the end or the beginning of the word. It considers a list of common prefixes & suffixes that can be found in an inflected word.

Below we have examples in both langage in English & Sanskrit.

form	Suffix	Stem
studies	-es	studi
studying	-ing	study
studied	-ed	studi
पठति	-ति	पठ
पठसि	-सि	पठ
पठत्	-त्	पठ

So, from example it is clear that a lemma is a base form of all its inflectional form whereas a stem isn't.

Ans → 2 Python code (Remember to install nltk library first)

```
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
antonyms = [ ]
text = input( ).strip( )
# tokenization
token1 = word_tokenize(text)
print (token1)
# antonyms of all words
for word in token1:
    for syn in wordnet.synsets(word):
        for abc in syn.lemms( ):
```

```
if abc.antonyms():
    antonyms.append(abc.antonyms()[0].name())

print(word, set(antonyms))
```

3] Let

doc-1 ⇒ my name Himanshu Gupt, a CSE student at IIIT Nagpur

doc-2 ⇒ I live in a small town of Jaipur in Rajasthan.

doc-3 ⇒ Current I am living in seminary hills of orange city.

doc-4 ⇒ I am writing the NLP end sem exam paper today.

doc-5 ⇒ my hobbies is reading, dancing, singing, playing chess & and sleeping.

To find tf-idf weight of ~~doc-1~~ all words of doc-1

(1) ~~my~~ term = my

$tf = \frac{1}{10}$

$idf = \log_{10}\left(\frac{5}{2}\right)$

Ans,

$tf\text{-}idf = tf \times idf = \frac{1}{10} \times 0.398 = 0.0398$

(2) ~~my~~ name

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(3) Himanshu

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(4) Gupta

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(5) a

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{2}\right) = 0.0398$

(6) CSE

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(7) Student

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(8) at

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(9) IIIT

$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$

(c) Nagpur:

$$tf \cdot idf = \frac{1}{10} \times \log_{10}\left(\frac{5}{1}\right) = 0.0699$$

Inverted index construction

Term	docID →	
my	1	5
name	1	
Himanshu	1	
Gupta	1	
a	1	2
CSE	1	
student	1	
at	1	
IIIT	1	
Nagpur	1	

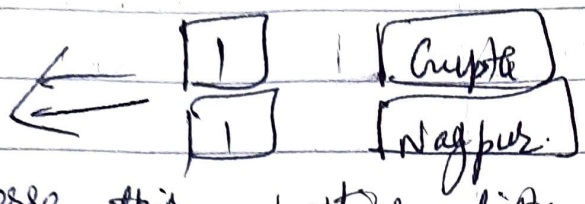
Vocab

Term	doc. freq	→ Posting lists.
my	2	1 → 5
name	1	1
Himanshu	1	1
Gupta	1	1
a	2	1 → 2
CSE	1	1
Student	1	1
at	1	1
IIIT	1	1
Nagpur	1	1

Query Processing

Let Query = Crupta and Nagpur

Locate Crupta in dictionary & Nagpur in Dict.



Merge this posting lists

→ walk through this list simultaneously and merge.

Limitation of Bag-of-words feature in sentiment classification:

When we consider bag of words method to generate vectors for large documents. then the resultant vectors will be of large dimensions and will contain far too many values resulting in sparse vectors.

Apart from resulting in Sparse, Bag of words does a poor job in making sense of text data for example, consider two sentences:-

"I love tea but not coffee."

"I love coffee but not tea".

Now, Bag of words approach will create similar vectors even both sentences have different meaning.

We can solve this problem by using following techniques.

(i) Positional Indexing :- Creating positional indexing for each word of sentence will solve this problem. It is able to distinguish between two sentences/documents.

(ii) ~~Converting all words to lower case~~ :-

(iii) ~~Removing stop words~~ Using term frequency (tf)

(iv) ~~Stemming and lemmatization~~. Using inverse df (idf)

Ans 6 The process of converting words into numbers are called vectorisation.

Step 1 → Tokenize the sentence

Sentence 1	Sentence 2	Sentence 3
I	Nagpur	Nagpur
am	is	is
studying	~~the~~ summer the	famous
in	~~capital~~ summer	for
~~IIIT~~	~~of~~ capital	~~orange~~ oranges.
Nagpur.	of	
	Maharastra	

Step 2 → Creating Dictionary of word frequency

Word	frequency
I	1
am	1
Studying.	1
in	1
IIIT	1
Nagpur	3
is	2
~~the~~	1

Teacher's Signature

Summer	1
capital	1
of	1
Maharashtra	1
famous	1
for	1
oranges	1

Now, sort word according to frequency.

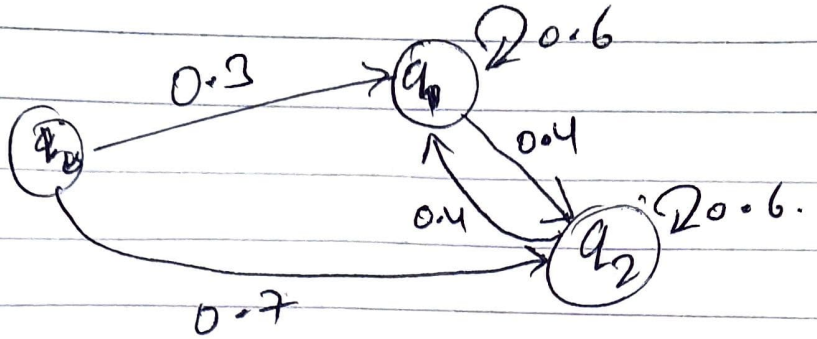
Now, creating vector of word having bit values.

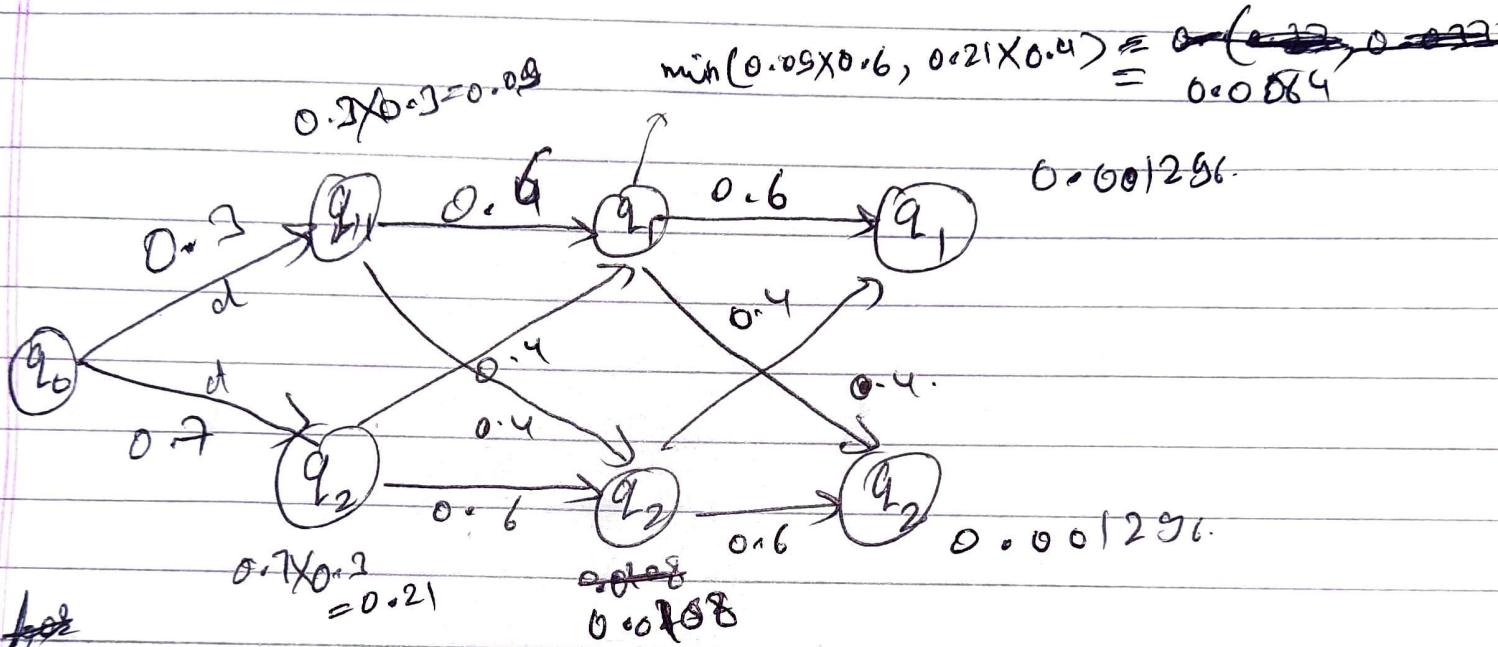
word	Sentence 1	Sentence 2	Sentence 3
I	1	0	0
am	1	0	0
Studying	1	0	0
in	1	0	0
IIIT	1	0	0
Nagpur	1	1	1
is	0	1	1
the	0	1	0
Summer.	0	1	0
capital	0	1	0
of	0	1	0
Maharashtra	0	1	0
famous	0	0	1
for	0	0	1
oranges	0	0	1

Ans) $Q = [1, 2]$. , $M = [Q, \Sigma, \sigma, Y, q_0]$.

Let start $= q_0$,

Now,





~~for~~

for string "dab", state sequence is

$$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2$$

Prob $= 0.09 \times 0.0108 \times 0.001296$.

$$= 1.26 \times 10^{-6}$$