Knowledge Management and Discovery

Problem Set 4

Name: Shalin Patel

Class ID: 21

Ans 1:

I: <s> I am Sam </s>

<s> Sam I am </s>

<s> I like green eggs and ham </s>

a) Bigram Approach

ngram	count
I am	2
I	1
am Sam	1
Sam Sam	1
Sam I	1
am I	1
I like	1
like green	1
green eggs	1
eggs and	1
and ham	1

=>P(I)*(P(I like)/P(I)))*(P(like green)/P(like)))*(P(like green)/P(green)))*(P(green eggs)/P(green)))*(P(eggs and)/P(eggs)))*(P(and ham)/P(and))

=> (1/11)*((1/11)/(1/11)) *((1/11)/(1/11)) *((1/11)/(1/11)) *((1/11)/(1/11)) *((1/11)/(1/11)) *((1/11)/(1/11))

1/11

=>

b) Trigram Approach

Total number of tokens: 12 Types: 12

ngram	count
I	1
I am	1
I am Sam	1
am Sam Sam	1
Sam Sam I	1
Sam I am	1
I am I	1
am I like	1
I like green	1
like green eggs	1
green eggs and	1
eggs and ham	1

=>P(I)*(P(like)/P(I)))*(P(I like green)/P(I like)))*(P(like green eggs)/P(like green)))*(P(green eggs and)/P(green eggs)))*(P(eggs and ham)/P(eggs and)))

=>(1/12)*((1/12)/(1/12)) *((1/12)/(1/12)) *((1/12)/(1/12)) *((1/12)/(1/12)) *((1/12)/(1/12))

=> 1/12

Ans 2)

A

Word to vectors is vector or weights. In word vocabulary, each word is represented by a 1 of n vector. The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

Suppose our vocabulary has only five words: a, b, c, d, e. We could encode the word ‘b as:

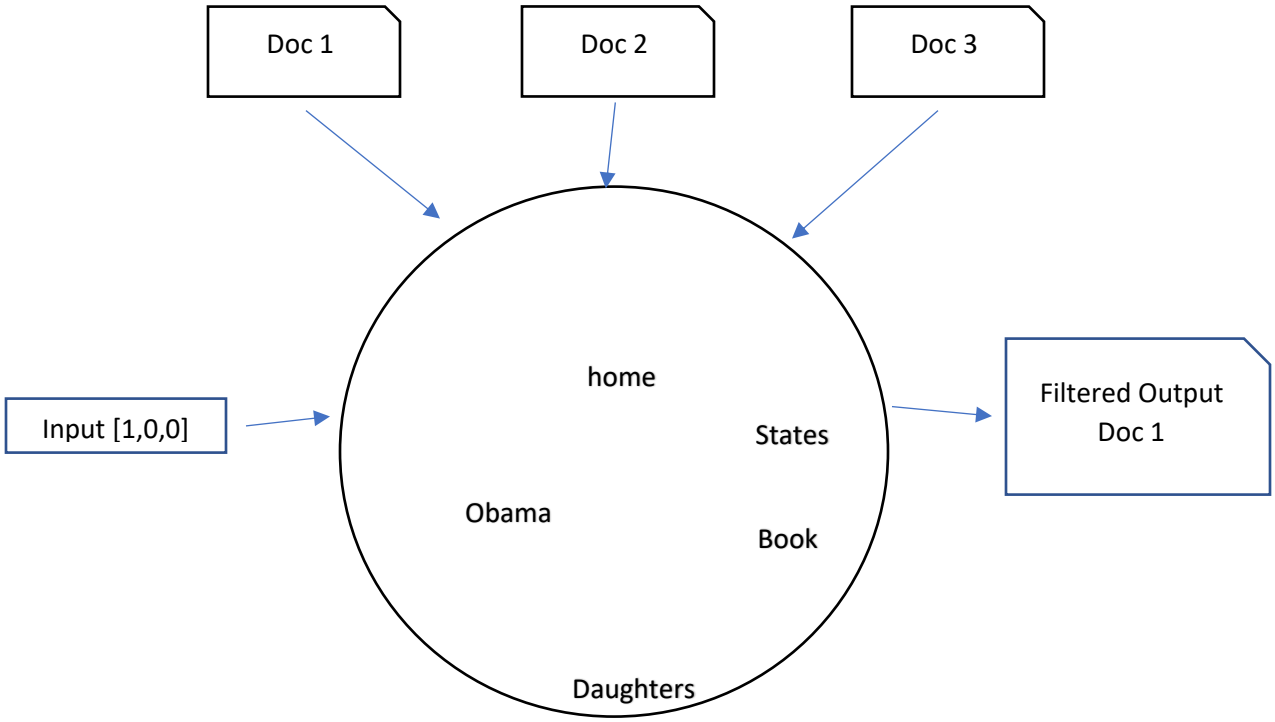
0	1(b)	0	0	0

Using such an encoding, there’s no meaningful comparison we can make between word vectors other than equality testing.

In word2vec, a *distributed* representation of a word is used. Take a vector with several hundred dimensions (say 1000). Each word is represented by a distribution of weights across those elements. So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all the elements in the vector, and each element in the vector contributes to the definition of many words.

b)

For given model we can separate the vectors depending on context.



We can give make vectors for each document individually and give it to make new model where an advanced level model can be build depending on relation between the words of cosine similarity. Input we can pass the vector with which brings directly documents as per passed value.

So, this model can be used when you are confident which document you are looking for otherwise it checks throughout all the documents.

Difference Between CBOW and Skip Gram

CBOW: The input to the model could be wi−2, wi−1,wi+1,wi+2wi−2,wi−1,wi+1,wi+2, the preceding and following words of the current word we are at. The output of the neural network will be wiwi. Hence you can think of the task as "predicting the word given its context"
Note that the number of words we use depends on your setting for the window size. The main application is to identify the missing word in a sentence or some long phrase. Also used to extract semantically reached bigram.

Skip-gram: The input to the model is wi, and the output could be wi−1, wi−2, wi+1,wi+2wi−1,wi−2,wi+1,wi+2. So, the task here is "predicting the context given a word". In addition, more distant words are given less weight by randomly sampling them. When you define the window size parameter, you only configure the maximum window size. The actual window size is randomly chosen between 1 and max size for each training sample, resulting in words with the maximum distance being observed with a probability of 1/c while words directly next to the given word are always(!) observed.

Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.
CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words

morning fog, afternoon light rain,

CBOW model

W1 (morning)
1
0
0
0
0

W2 (fog)
0
1
0
0
0

W3 (afternoon)
0
0
1
0
0

W4 (light)
0
0
0
1
0

W5 (rain)
0
0
0
0
1

W2 VxN

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Skip Gram Model:

Input Layer

W1	morning
W2	fog
W3	afternoon
W4	light
W5	rain