Student id: 16241583

CS5560 Knowledge Discovery and Management

Problem Set 4

June 26 (T), 2017

Name: Revanth Chakilam
Class ID: 02 .

I. N-Gram

Consider a mini-corpus of three sentences

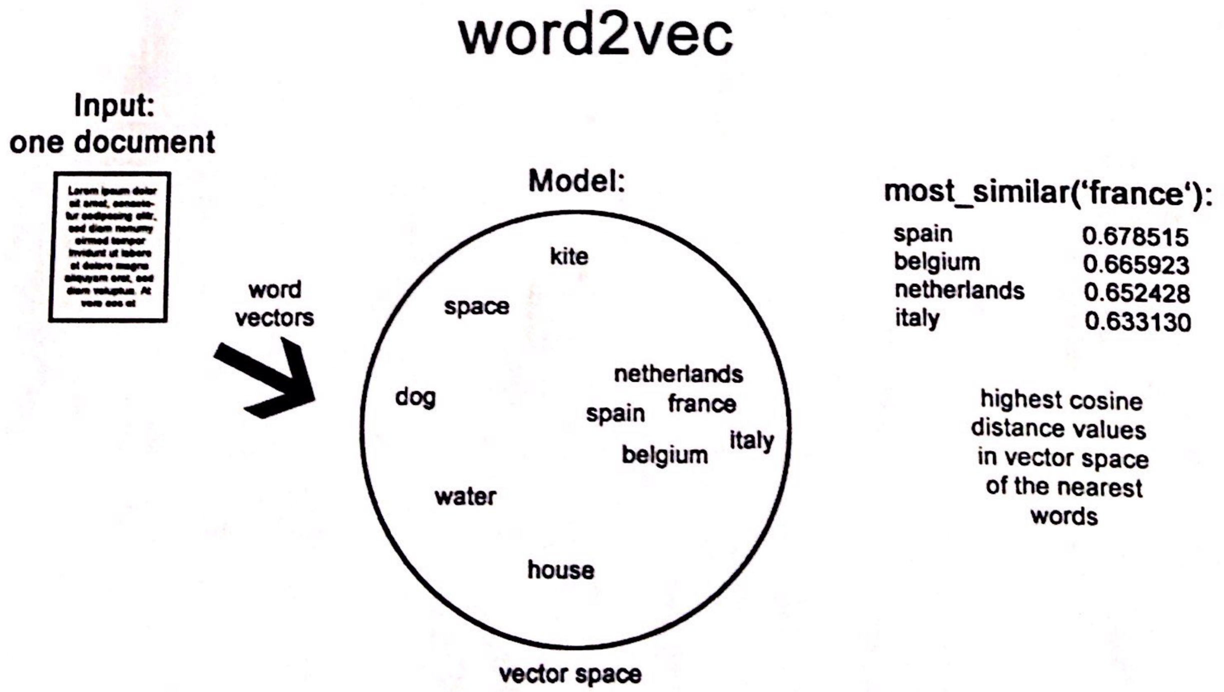
<s> I am Sam </s>
<s> Sam I am </s>
<s> I like green eggs and ham </s>

1) Compute the probability of sentence "I like green eggs and ham" using the appropriate bigram probabilities.
2) Compute the probability of sentence "I like green eggs and ham" using the appropriate trigram probabilities.

II. Word2Vec

Word2Vec reference: https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

Consider the following figure showing the Word2Vec model.



a. Describe the word2vec model

Scanned by CamScanner

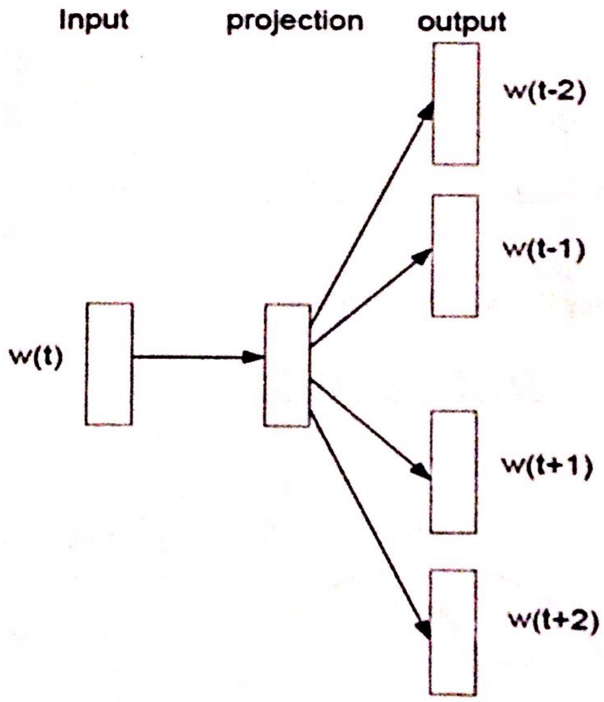
b. Describe How to extend this model for multiple documents. Also draw a similar diagram for the extended model.

Describe the differences of the following approaches

- Continuous Bag-of-Words model,
- Continuous Skip-gram model

For the sentence "morning fog, afternoon light rain,"

- Place the words on the skip-gram Word2Vec model below.
- Draw a CBOW model using the same words.



Scanned by CamScanner

Revanth Chakilam

(1).

N-Gram: N-gram includes encoding of key words and also word ordering automatically.

1) Bi-gram probabilities:

Calculation:

$$P(w_i/w_{i-1}) = Count(w_{i-1}, w_i) / Count(w_{i-1})$$

Probability that word$_{i-1}$ is followed by word $_i$ =

$$\frac{[\text{No. of times we saw word}_{i-1} \text{ followed by word}_i]}{[\text{No. of times we saw word}_{i-1}]}$$

s – beginning of sentence

/s – end of sentence

$P(I/s) = 2/3$

$P(like/I) = 1/3$

$P(green/like) = \frac{1}{1} = 1$

$P(eggs/green) = 1/1 = 1$

$P(and/eggs) = 1/1 = 1$

$P(ham/and) = 1/1 = 1$

$P(is/ham) = 1/1 = 1$

Scanned by CamScanner

2) Tri-gram probabilities:

Calculation:

$$P(w_i / w_{i-1}, w_{i-2}) = Count(w_i, w_{i-1}, w_{i-2}) / Count(w_{i-1}, w_{i-2})$$

Probability that we saw $word_{i-1}$ followed by $word_{i-2}$ followed by $word_i$ = $\frac{\text{Num times we saw the 3 words in order}}{\text{Num times we saw } word_{i-1} \text{ followed by } word_{i-2}}$

$P(green / I\ like) = Count(green\ I\ like) / Count(I\ like) = \frac{0}{1} = 0$

$P(eggs / like\ green) = Count(eggs\ like\ green) / Count(like\ green) = \frac{0}{1} = 0$

$P(and / green\ eggs) = Count(and\ green\ eggs) / Count(green\ eggs) = \frac{0}{1} = 0$

$P(ham / eggs\ and) = Count(ham\ eggs\ and) / Count(eggs\ and) = \frac{0}{1} = 0$

II Word 2 Vec:

a) Description: It is a two layer neural network that process the text. The input is a text corpus.

Output which we receive is a set of vectors: feature vectors for words in that corpus.

W2V is not a deep Neural network but a numerical form that deep nets can understood. There is no similarity as expressed 90° angle,

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

Scanned by CamScanner

Revanth Chakilam

In the representation, the input document is used to build a word 2 vec model contains word in the document and found the nearest words using cosine similarity.

b) Description of the Extension of W2V for multiple documents :

An extension of W2V model to construct embeddings from entire documents is called paragraph 2 vec or doc 2 vec.

Doc2vec is an unsupervised algorithm to generate vectors for sentence / paragraphs / documents. This algorithm is an adoption of W2V model which generate vectors for words.

The vectors generated by doc2vec can be used for tasks like finding similarity between sentences / paragraphs / document. Doc2Vec sentence vectors are word order independent. It generate word vectors constructed from character n grams & adding up the word vectors to compose a sentence vector. It generate vector where the vector for a sentence is generated by predicting the adjacent sentences, that are assumed to be semantically related.

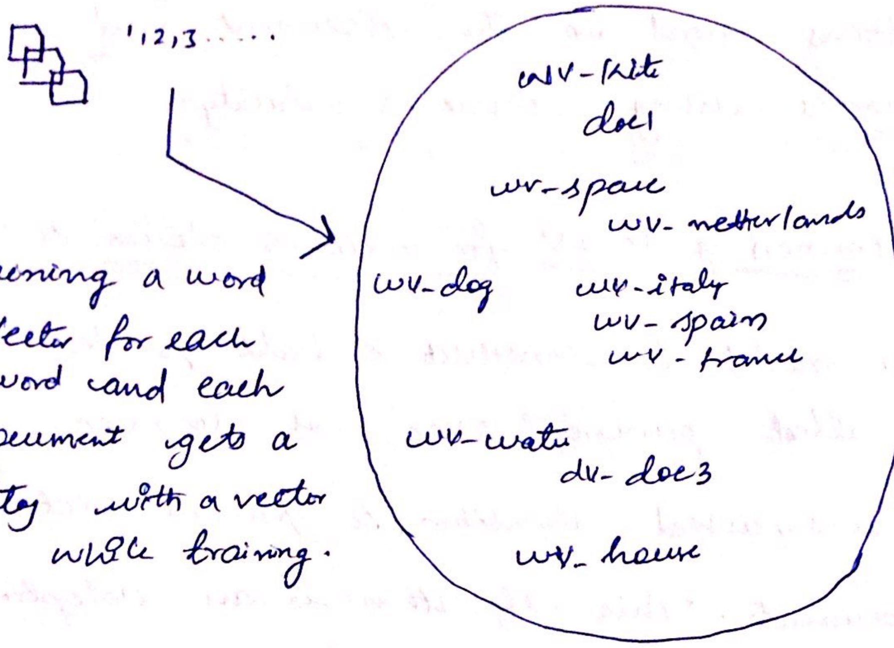
Scanned by CamScanner

Model:

Input

'n' no. of docs

'1,2,3 . . . .



training a word vector for each word and each document gets a Id/tag with a vector while training.

most-similar ('france')

paris	0.5
louvre	0.7
normandy	0.6.
-----	

highest cosine distance values in vector space with consideration of the document vectors.

Vector space: Consists of word vectors for each word & additional document vectors.

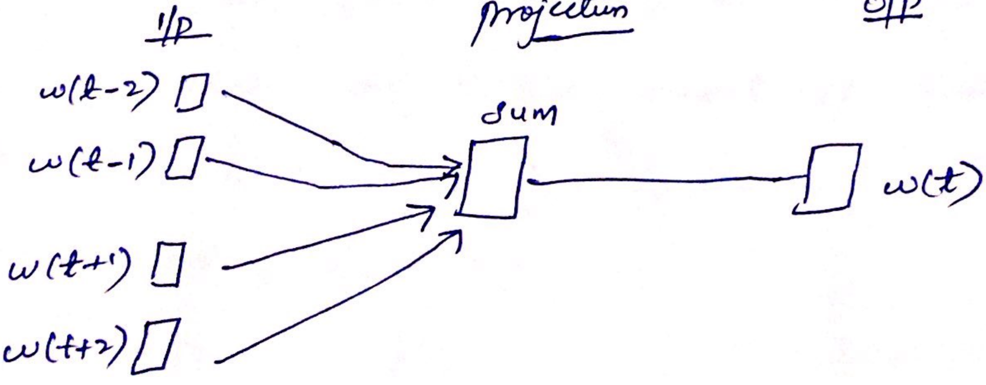
W2V model can utilize either of two model architectures to produce a distributed representation of words

a) Continous Bag of words (CBOW)

b) Continous skip-gram

CBOW:

In the CBOW architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction.

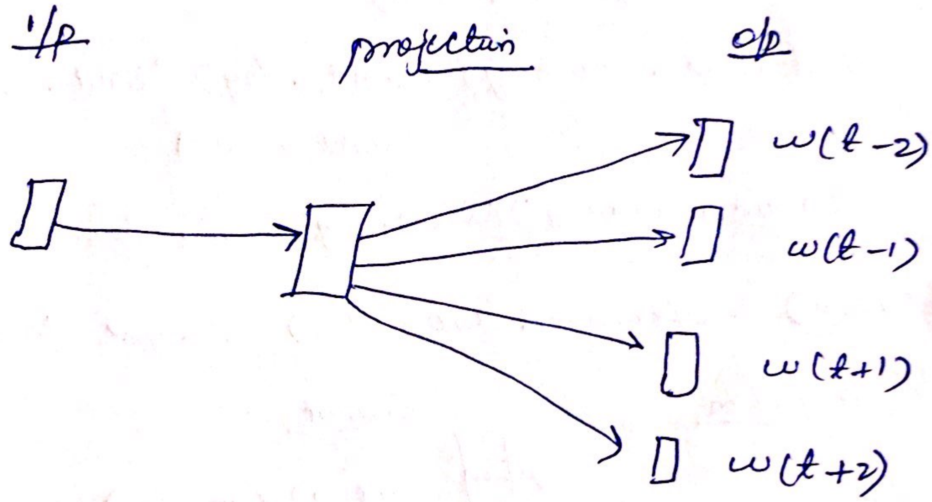


Scanned by CamScanner

Revanth Chakilam

Continous skip-Gram:

In this architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words.



Diff b/w CBOW & Continous skip gram:

① In CBOW, we need to think task as "predicting the word gi" where as "we think predicting given a word"

② skip gram suits small amount of training data.

③ CBOW is faster to train, & slightly has a better accuracy

④ In skip-gram we need to create a lot more training instances from limited amount of data & for CBOW, we need more since the conditioning on context, which can get exponentially huge.

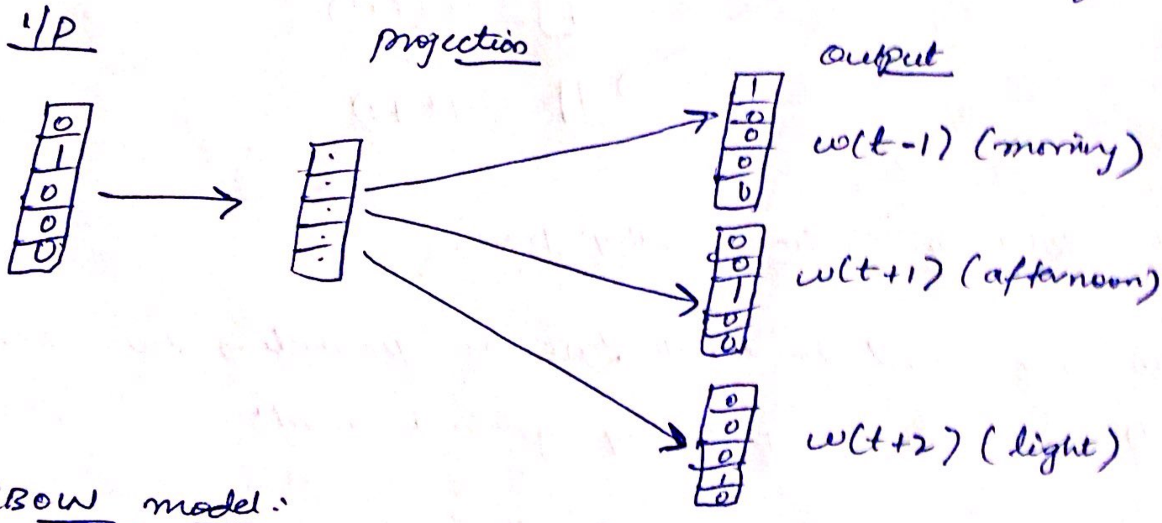
Scanned by CamScanner

•

" Morning fog , afternoon light rain "

<u>I/P</u>	<u>Training samples</u>
Morning	(morning , fog), (morning , afternoon)
fog	(fog, morning) (fog , afternoon) (fog , light)
afternoon	(afternoon, morning)(afternoon , fog) (afternoon , light) (afternoon , rain)
light	(light , morning) (light , fog) (light , afternoon), (light , rain)
rain	(rain , morning)(rain, fog) (rain, afternoon) (rain, light)

morning :- (1,0,0,0,0)   afternoon: (0,0,1,0,0) , light (0,0,0,1,0)



• CBOW model: