CS5560 Knowledge Discovery and Management

Problem Set 4

June 26 (T), 2017

Name: Syed Moin

Class ID: 28

I. N-Gram

Consider a mini-corpus of three sentences

<s> I am Sam </s>

<s> Sam I am </s>

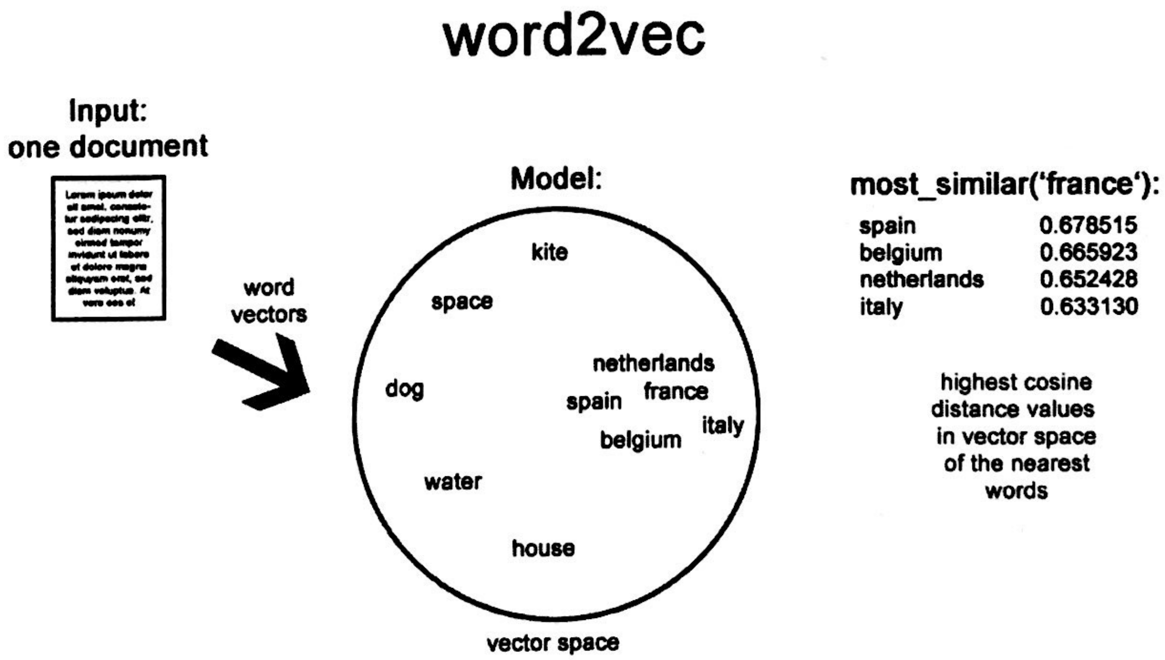
<s> I like green eggs and ham </s>

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

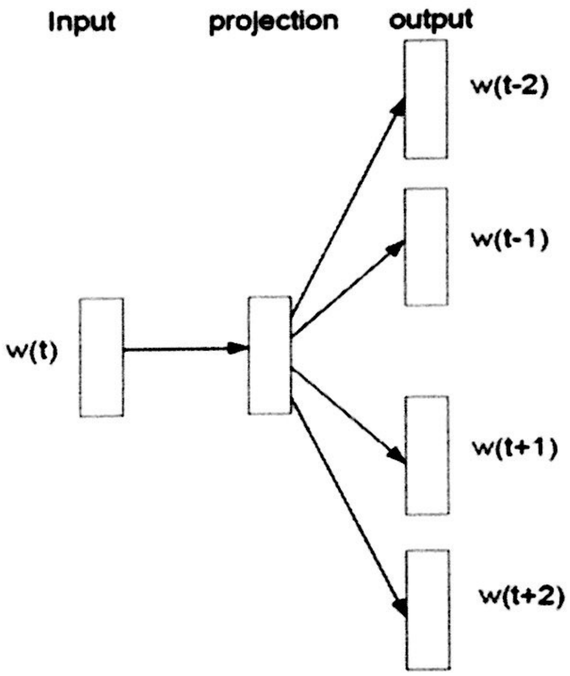
b. Describe How to extend this model for multiple documents. Also draw a similar diagram for the extended model.

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.



I) N-GRAM:-

1) compute the probability of sentence "I like green eggs and ham" using bigram probabilities

Calculating bigram probabilities:

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

ie,

probability that $word_{i-1}$ is followed by $word_i$ = $\frac{[\text{Num times we saw } word_{i-1} \text{ followed by } word_i]}{[\text{Num times we saw } word_{i-1}]}$

S = beginning of sentence

/S = end of sentence

$P(I/S) = \frac{2}{3}$

$P(like/I) = \frac{1}{3}$

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

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

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

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

$P(/S/ham) = \frac{1}{1} = 1$

2) compute for trigram probabilities

Calculating Trigram probabilities

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

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(~~green~~ 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) word 2 vec model.

It is a two-layer neural network that processes the text

Input is a text corpus

Output is a set of vectors, feature vectors for words in that corpus.

NOT a deep neural network, but a numerical form that deep nets can understand

No similarity is expressed as a 90° angle, total similarity of 1 is a 0 degree angle.

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

In the model represented in the diagram they have taken a input document and build a word 2 vec model contains word in the document and found the nearest words using cosine similarity.

b) Extension of word2vec for multiple documents:

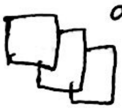
An extension of word 2 vec to construct embeddings from entire documents is called paragraph 2 vec or doc 2 vec.

Doc2Vec is an unsupervized algorithm to generate vectors for sentence/paragraphs/documents. The algorithm is an adaptation of word2vec which can generate vectors for words.

The vectors generated by doc2Vec can be used for tasks like finding similarity between sentences/paragraphs/documents. Doc2vec sentence vectors are word order independent. It generate word vectors constructed from character n grams and then adding up the word vectors to compose a sentence vector.

Doc2vec for diagram mentioned:

Input documents:



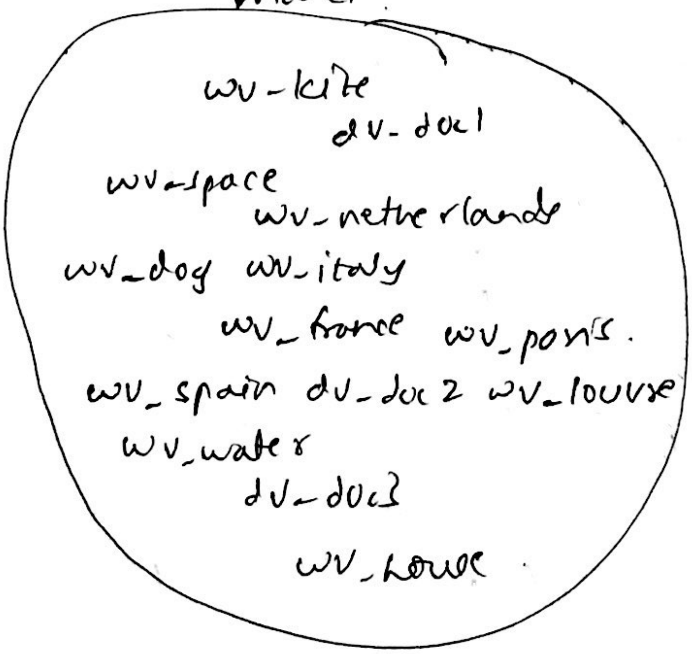
most_similar('france')

paris	0.875643
louvre	0.765432

1 - -

highest cosine distance values in vector space with consideration of the documents vector

model:



Training a word vector for each word and each document gets an ID/tag with a vector while training

vectorspace consists of word vectors for each word & additional document vectors.

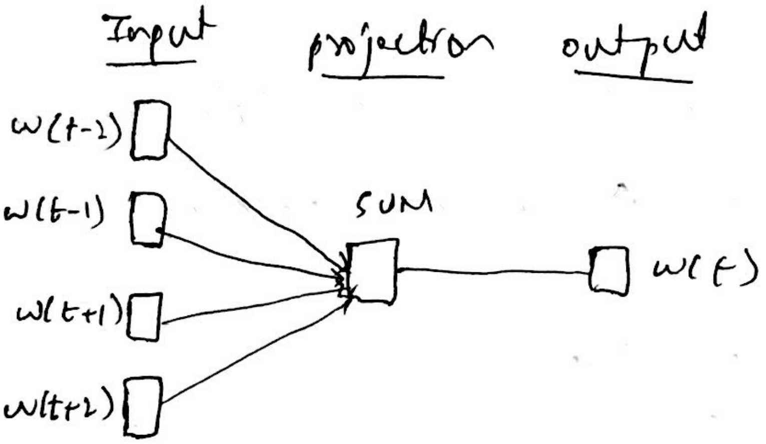
word2vec can utilise either of two model architectures to produce a distributed representation of words.

a) continuous bag-of-words (CBOW)

b) continuous skip-gram

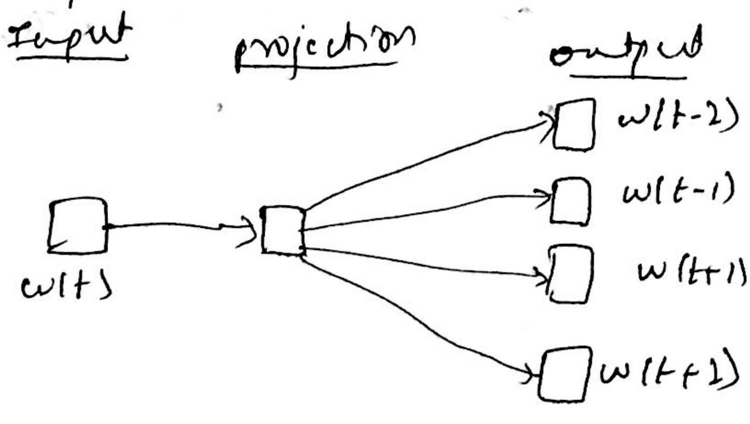
a) CBOW

In the continuous bag of words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag of words assumption).



b) continuous skip-gram:-

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



Difference between CBOW and continuous skip gram :-

1) In CBOW we need to think task as "predicting the word given its context" where as in the skip-gram we think task as "predicting the context given a word".

2) Skip-gram works well with small amount of training data, represents well even rare words or phrases

3) Skip gram, in this we need to create a lot more training instances from limited amount of data and for CBOW, we need more since you are conditioning on context, which can get exponentially huge.

→ Given the sentence is "morning fog, afternoon light rain",

i) Skip gram word2vec model for above sentence is

Consider the sentence :-

Morning fog, afternoon light rain.

consider window size is 7

Input	Training samples
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)

we need to build a vocabulary of words
(morning, fog, afternoon, light, rain)

consider input is fog then vector representation is.

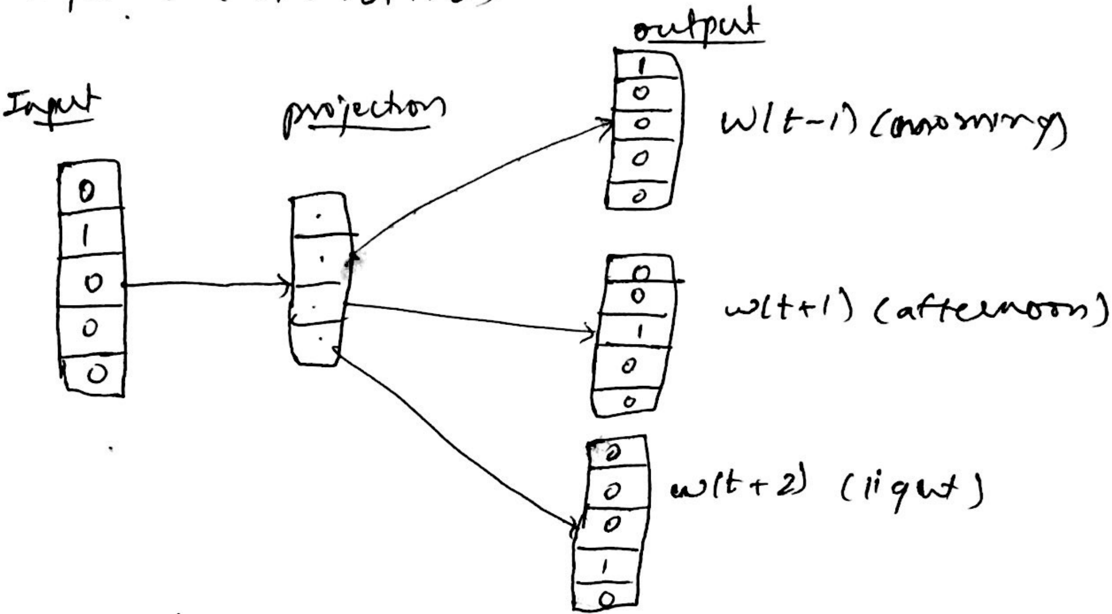
(0,1,0,0,0)

similarly the vectors representation for morning, afternoon and light are as follows, because there are in the context of that particular input word.

morning : (1,0,0,0,0)

afternoon : (0,0,1,0,0)

light : (0,0,0,1,0)



CBOW model