CS5560 Knowledge Discovery and Management

Problem Set 4

June 26 (T), 2017

Name: SAIJYOTHI AUDIBANDI

Class ID: 7.

I. N-Gram

Consider a mini-corpus of three sentences

<s> I am Sam </s>

<s> Sam I am </s>

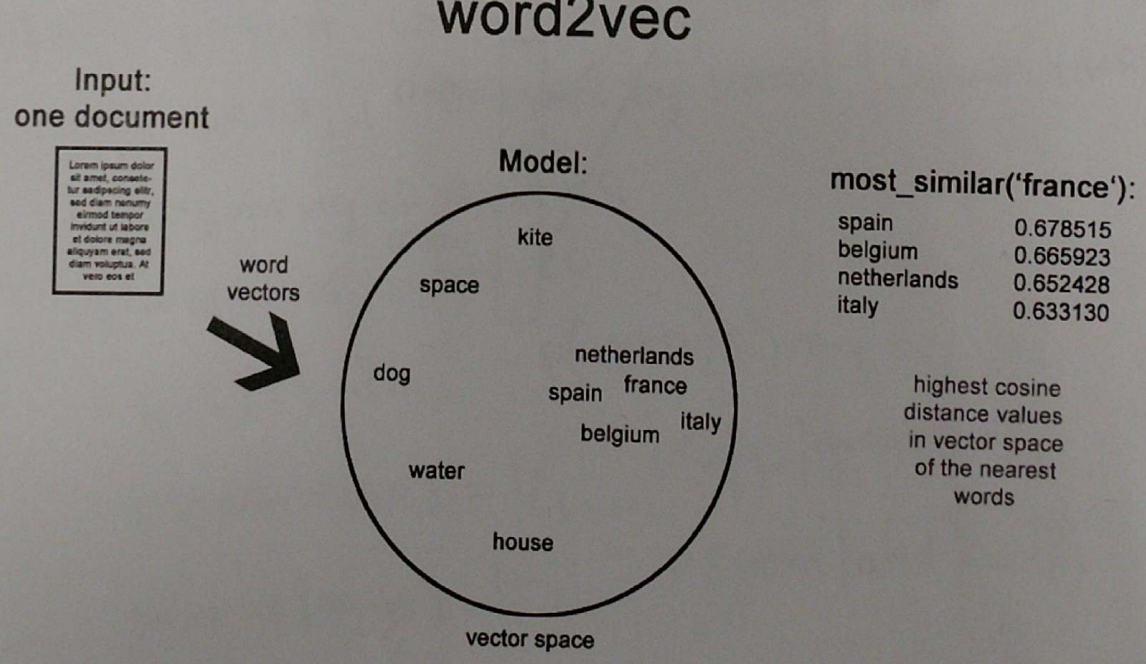
<s> I like green eggs and ham </s>

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.

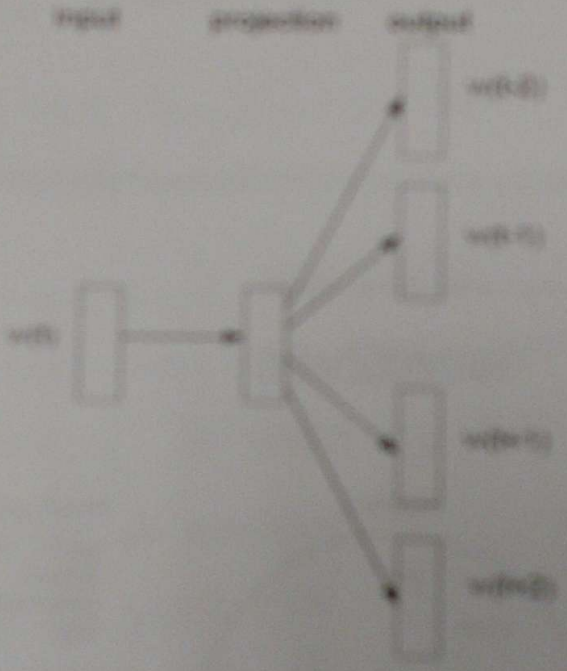
II. Word2Vec

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



I

N-Gram: A sequential list of the n words, often used in information retrieval and language modeling to encode the likeho that the phrase will appear in the future.

N-Gram based approaches create applications probabilistic models of n-grams from a given corpus of text and tag new utterances using these models.

Given a min-corpus of 3 sentences

<s> I am sam </s>

<s> sam I am </s>

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

1. Calculating the bigram probability of sentence "I like green eggs and ham".

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

probability that word$_{i-1}$ is followed by word$_i$ = [Num times w saw word$_{i-1}$ followed by word$_i$] / Num times we saw word$_{i-1}$.

s - beginning of sentence

/s - end of sentence.

$P(I|s) = 2/3$ $P(like|I) = 1/3$

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

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

2. Calculating the probability of sentence "I like green eggs and ham" using 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 $word_i$ = [no. of times we saw the 3 word in order] / [no. of times we saw $word_{i-1}$ followed by $word_{i-2}$]

$P(green | I\ like) = count(green\ I\ like) / count(I\ like) = 0/1 = 0.$

$P(eggs | like\ green) = count(eggs\ like\ green) / count(like\ green) = 0$

$P(and | green\ eggs) = count(and\ green\ eggs) / count(green\ eggs) =$

$P(ham | eggs\ and) = count(ham\ eggs\ and) / count(eggs\ and) = 0.$

II

(a) Word2vec model:-

A two layer neural net that process text.

- Input is a text corpus.
- output is a set of vectors.
- Not a deep neural n/w, but numerical form that deep net can understand.

Measuring cosine similarity:-

- No similarity is expressed as a 90 degree angle.
- Total similarity of 1 is a 0 degree angle.

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

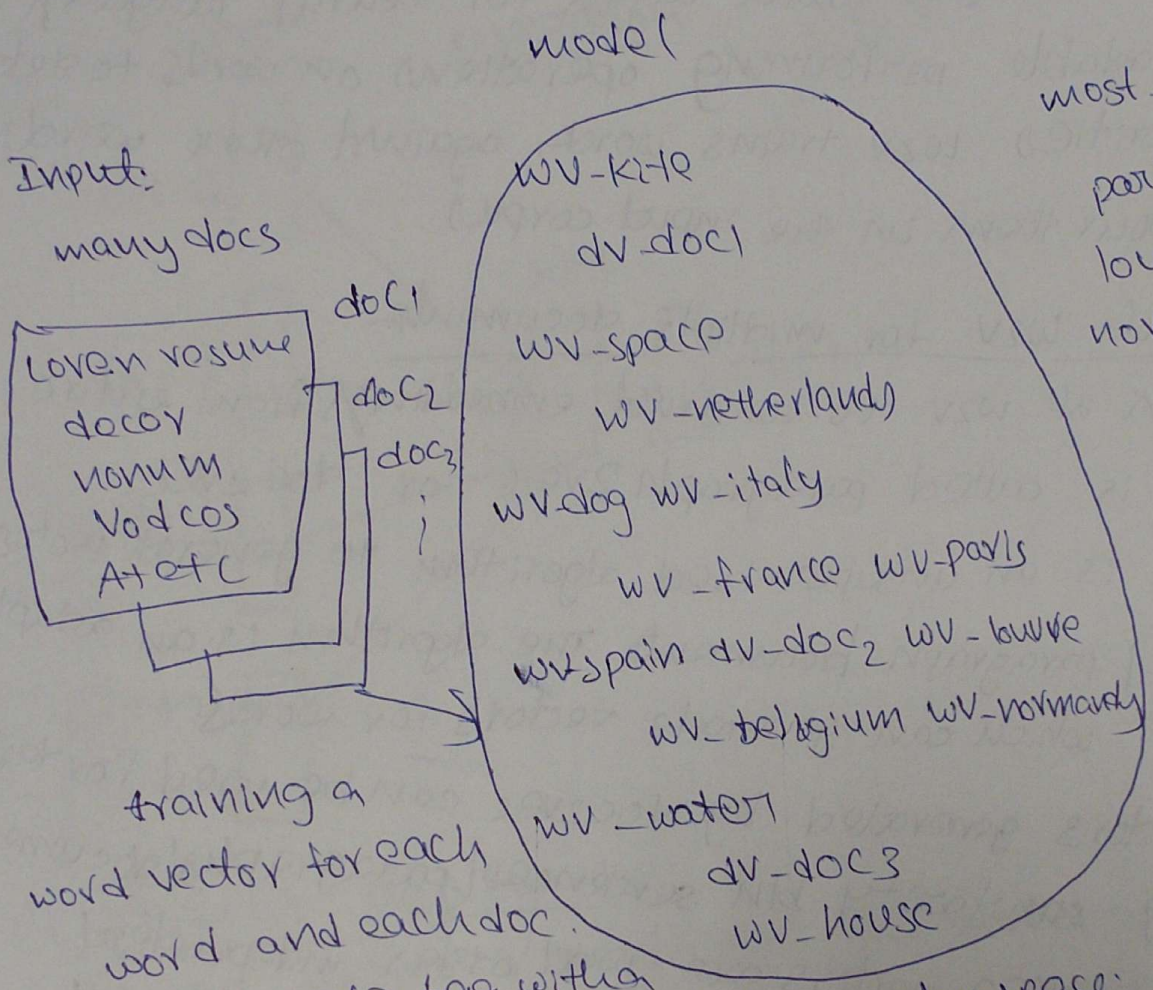
word 2vector "vectorizes" about words for neural language computer readable performing operations on words to detect their similarities. W2V trains words against other words that neighbour them in the input corpus.

(b) Extension of W2V for multiple documents:-

An extension of W2V to construct embeddings from entire documents is called paragraph2vec or doc2vec.

Doc2vec is an unsupervised algorithm to generate vectors for sentence / paragraph documents. The algorithm is an adaption of word2vec which can generate vectors for words.

The vectors generated by doc2vec can be used for tasks like finding similarity b/n sentences/ paragraphs/ documents. Doc2vec sentence vectors are word order independent. It generate word vectors constructed from character n grams and then adding up to the word vectors to compose a sentence vector. It generated by predicting the adjacent sentences, that are assumed to be semantically related.



most_similar('france')

paris 0.876543
louvre 0.765432
normandy 0.654321

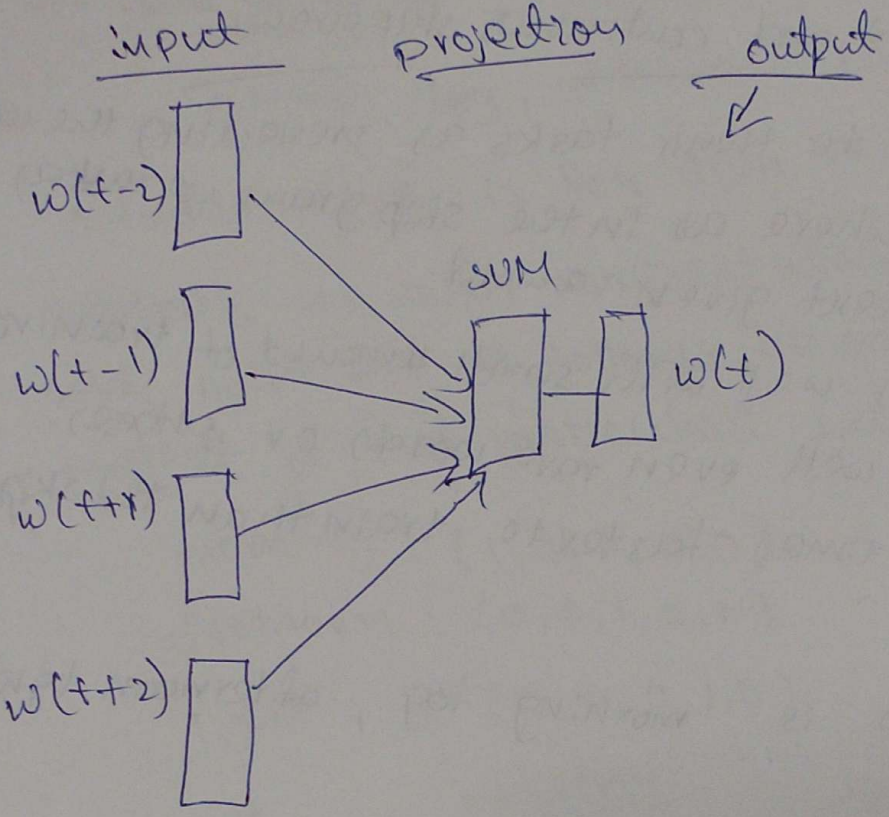
higher cosine distance values in vector space with consideration of the document vectors

training a word vector for each word and each doc. gets an ID tag with a vector while training

vector space: consists of word vector for each word & additional doc. vectors.

w2v can utilize either of two model architectures to produce a distributed representation of words.

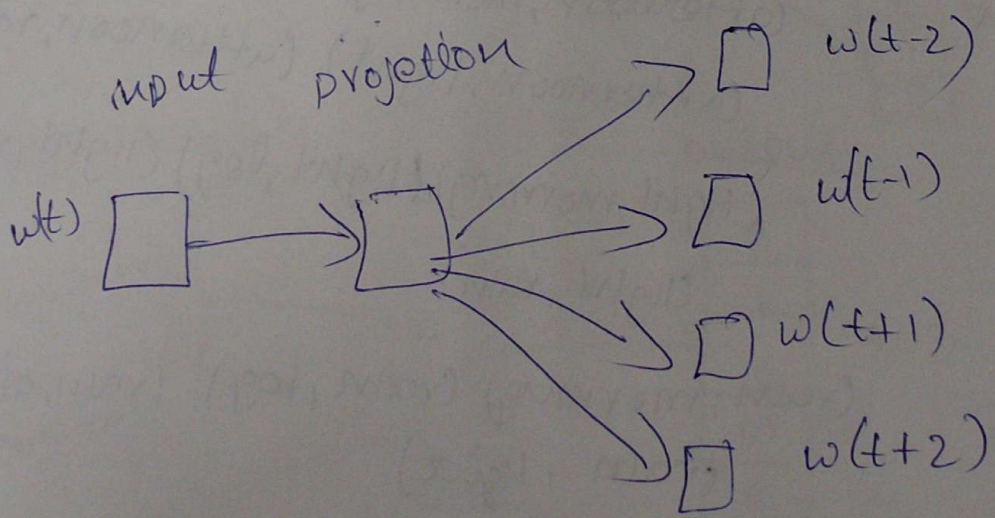
(a) continuous bag of words:- In the continuous bag of words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction.



(b) continuous - skip Gram:-

In the continous skip-Gram, 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 words.



Differences b/w CBOW and continuous skip Gram-

1. In CBOW we need to think tasks as 'predicting the word given its context' where as in the skip gram we 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. CBOW is several times faster to train than the skip gram.

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

Skip-Gram:-

consider window size as '1'.

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:-

(words), fog, afternoon, light, rain)

consider, input is fog then vector representation is (0,1,0,

similarly the vector 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).