CS5560 Knowledge Discovery and Management

Problem Set 4

June 26 (T), 2017

Name: Yalamanchili Soomya
Class ID: 30

I. N-Gram

Consider a mini-corpus of three sentences

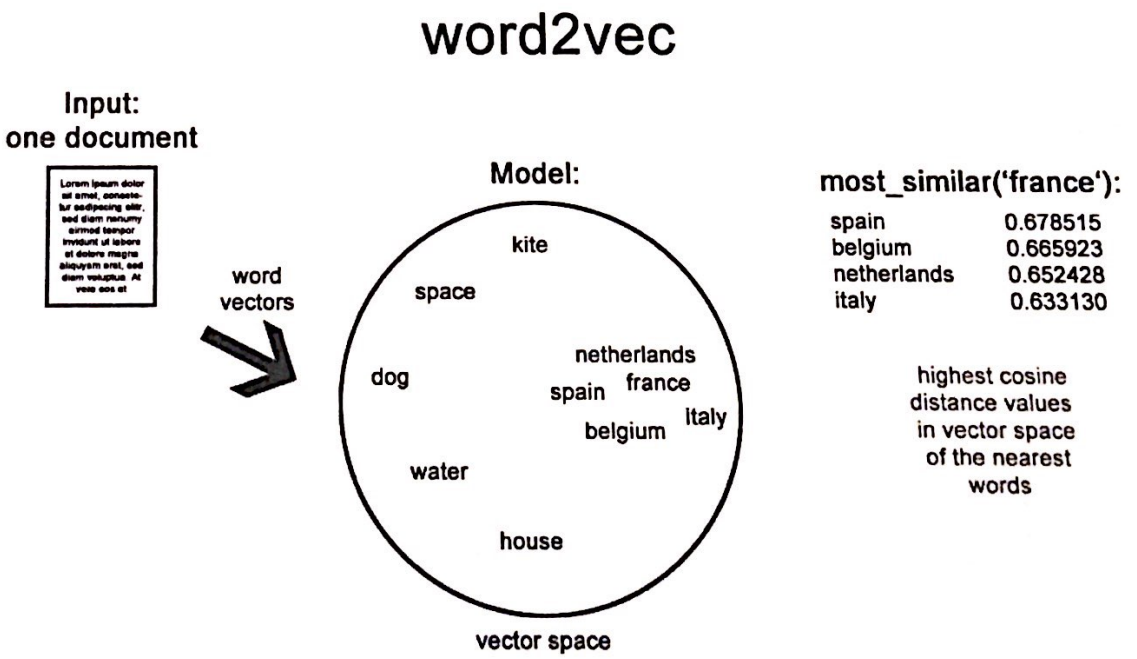
<s> I am Sam </s>
<s> Sam I am </s>
<s> I like green eggs and ham </s>

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

Name: Yalamanchili Soumya
ClassID: 30

I) N-Gram:- A sequential list of n words, often used in information retrieval and language modeling to encode the likelihood that the phrase will appear in the future. Some of the advantages of the N-Gram include it encode not just keywords, but also wordordering automatically

Given 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 Bigram probabilities of the 3rd sentence

calculating bigram probabilities:

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

In English

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

S = beginning of sentence
/S = end of sentence.

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

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

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

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

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

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

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

② Trigram probabilities of the 3rd sentence

Calculating Tri-gram probabilities

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

In english:

probability that we saw word$_{i-1}$ followed by word$_{i-2}$ followed by word$_i$

$$= \frac{\text{Num times we saw the 3 words in order}}{\text{Num times we saw word}_{i-1} \text{ followed by word}_{i-2}}$$

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

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

$P(\text{and} | \text{green eggs}) = \text{count}(\text{and green eggs}) / \text{count}(\text{green eggs}) = \frac{0}{1} = 0$

$P(\text{ham} | \text{eggs and}) = \text{count}(\text{ham eggs and}) / \text{count}(\text{eggs and}) = \frac{0}{1} = 0$

II Word2Vec

(a) word2vec 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 word2vec model contains word in the document and found the nearest words using cosine simlarit

Name: Yalamanchili Sowmya

class ID: 30

(b) Extension of word2vec for multiple documents

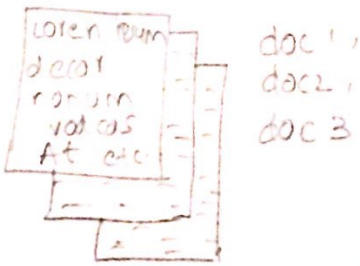
An extension of word2vec to construct embeddings from entire documents is called paragraph2vec or doc2vec.

Doc2vec is an unsupervised 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. It generate vectors where the vector for a sentence is generated by predicting the adjacent sentences, that are assumed to be semantically related.

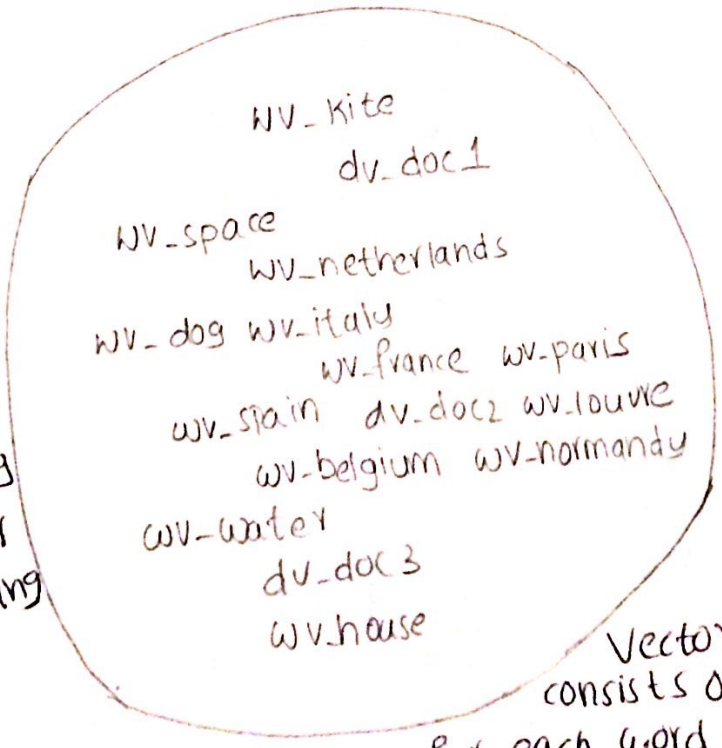
Doc2vec for diagram mentioned

Input:
many document



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

Model:



Vector space:
consists of word vectors for each word & additional document vectors

most_similar ('france'):

paris	0.576543
louvre	0.765432
normandy	0.654321
...	

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

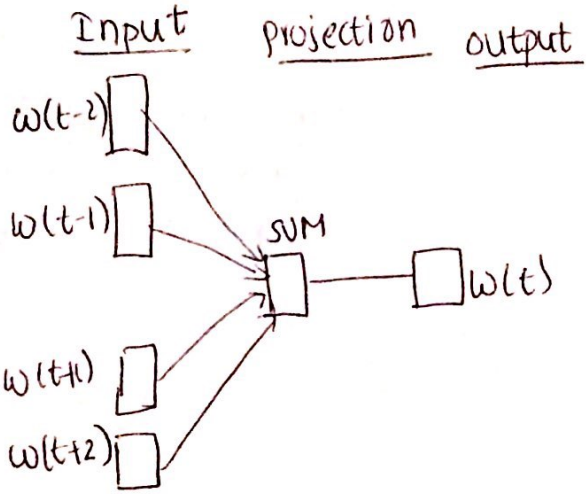
Word2vec can utilize either of two model architectures to produce a distributed representation of words.

ⓐ Continuous bag of words (CBOW)

ⓑ Continuous skip-gram.

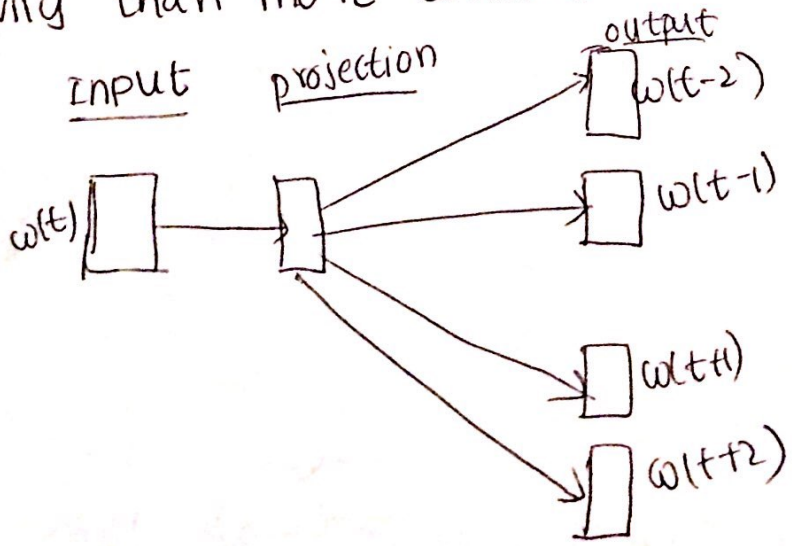
CBOW

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).



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 weighs nearby context words more heavily than more distant context words



Name: Yalamanchili Soumya

ClassID: 30

Differences 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 the training data, represents well even rare words or phrases.

③ CBOW is several times faster to train than the skip-gram, slightly better accuracy for the frequent words.

④ 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,"

skip-gram word2vec model for above sentence is

Consider the sentence :-

Morning fog, afternoon light rain

consider window size is 2

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 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)