CS5560 Knowledge Discovery and Management

Problem Set 4
June 26 (T), 2017

Name:
Class ID:

I. N-Gram

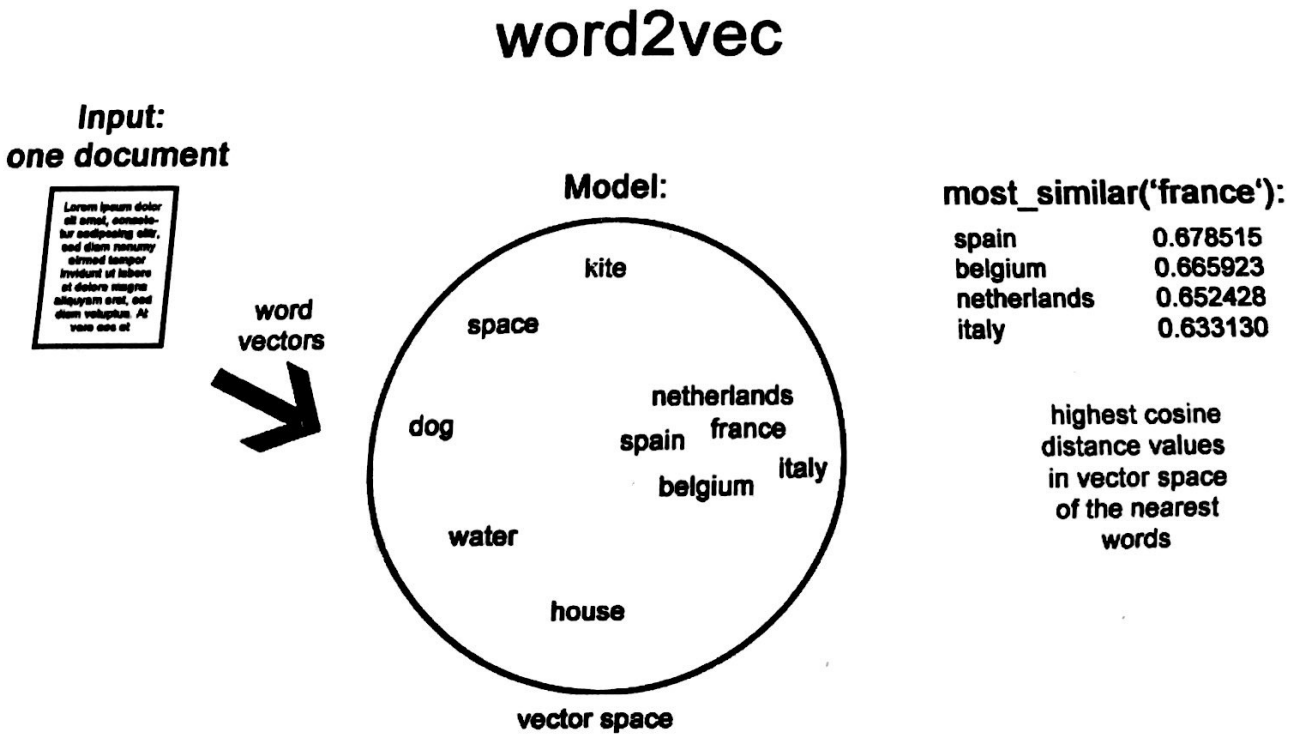
Consider a mini-corpus of three sentences
<s> I am Sam </s>
<s> Sam I am </s>
<s> I like green eggs and ham </s>

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

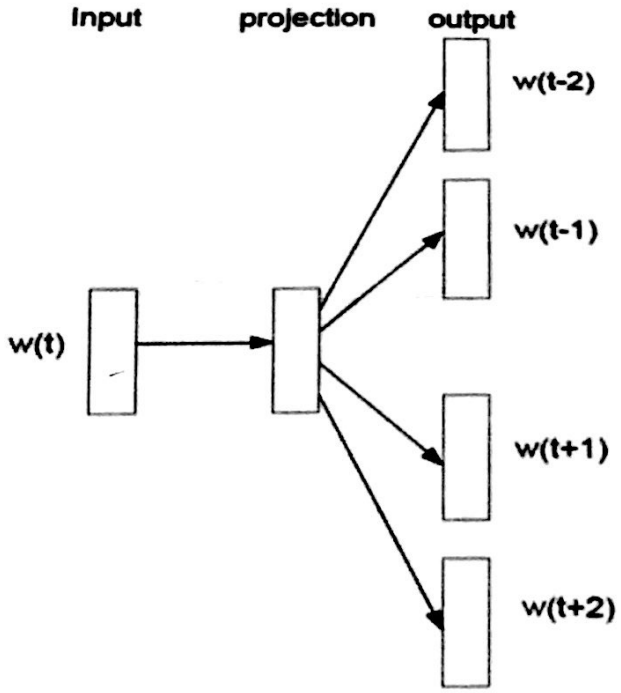
b. Describe How to extend this model for multiple documents. Also draw a similar diagram for the extended model.

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

⇒ N-gram Based Approaches create probabilistic models of n-grams from a given corpus of text and tag new utterances using these models.

Given a min-corpus of three 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

$$P(I|s) = \frac{2}{3}$$
$$P(like|I) = \frac{1}{3}$$
$$P(green|like) = \frac{1}{1}$$
$$P(eggs|green) = \frac{1}{1}$$
$$P(and|eggs) = \frac{1}{1}$$
$$P(ham|and) = \frac{1}{1}$$

2. Calculating the probability of sentence "I like green eggs and ham" using trigram probabilities.

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

probability that we saw $word_{i-1}$ followed by $word_{i-2}$ followed by $word_i$ = [Num times we saw the three words in order] / [Num times we saw $word_{i-1}$ followed by $word_{i-2}$]

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

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

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

$$P(\text{ham} | \text{eggs and}) = \text{Count}(\text{ham eggs and}) / \text{Count}(\text{eggs and}) = 0$$

II a) **Word2Vec model:**

A two-layer neural net that processes 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.

Measuring cosine similarity,
→ No similarity is expressed as a 90 degree angle,
→ Total similarity of 1 is a 0 degree angle.

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

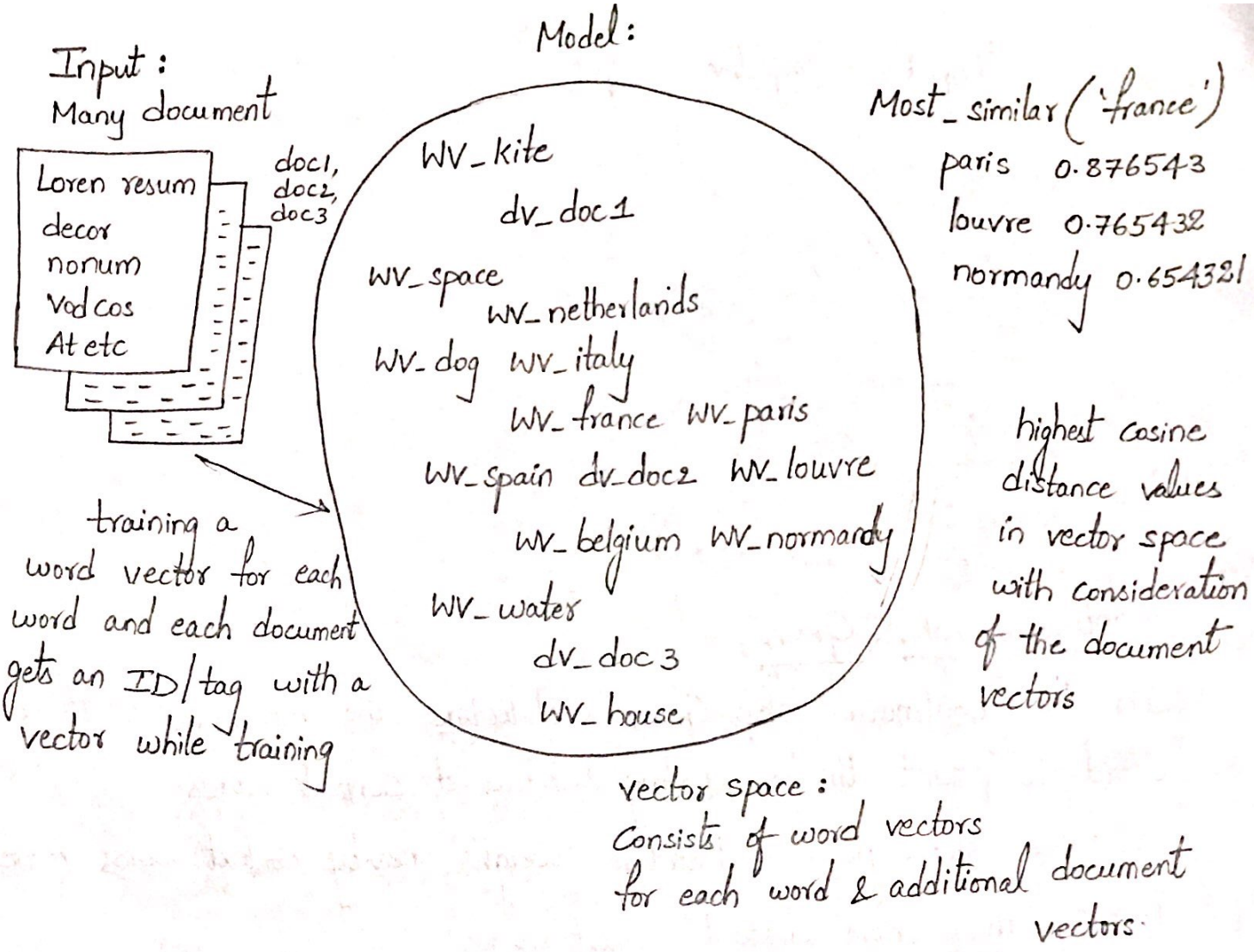
Word2vec "vectorizes" about words for natural language computer-readable performing operations on words to detect their similarities. Word2vec trains words against other words that neighbor them in the input corpus

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 adaption 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 to 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

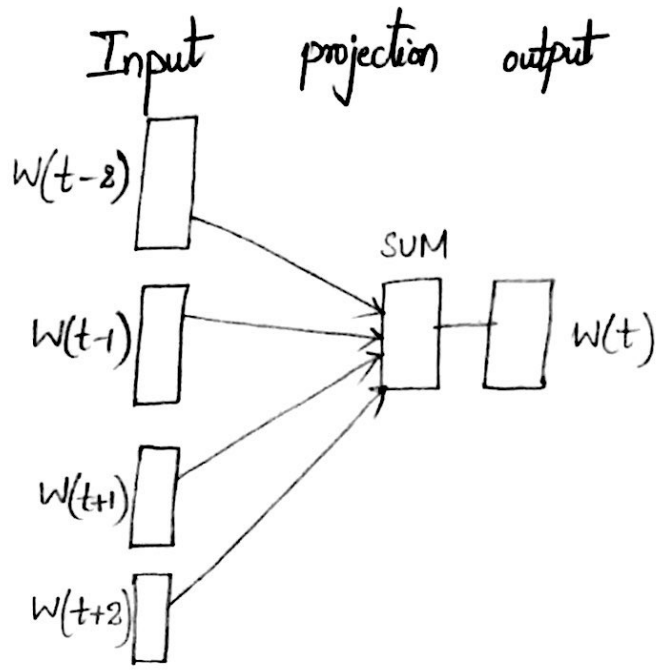


Word2Vec can utilize either of two model architectures to produce a distributed representation of words.

a) Continuous bag of words (CBOW)

b) Continuous skip-gram.

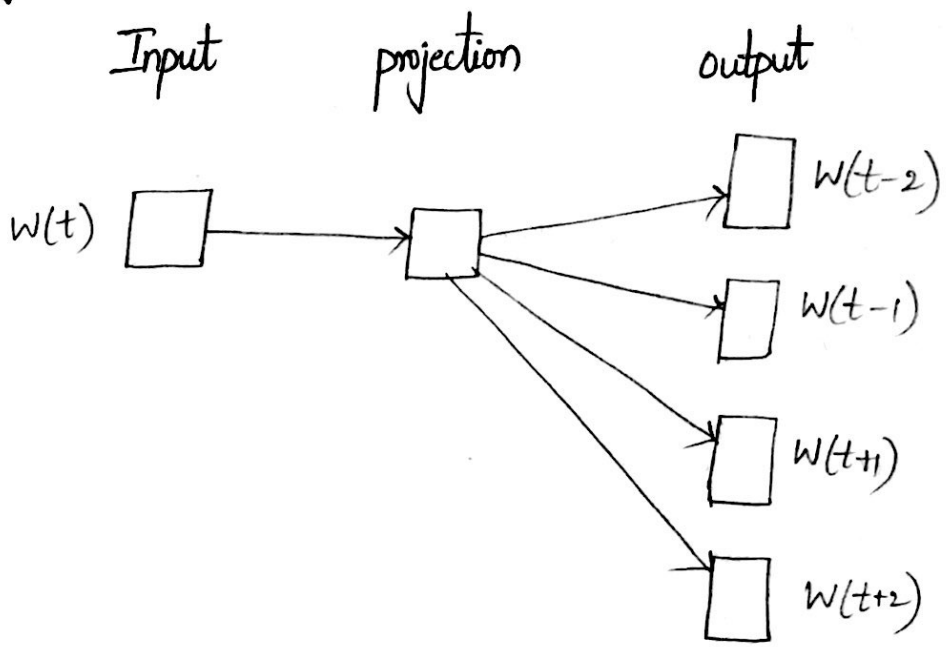
CBOW : In the continuous bag of words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption).



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.



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 frequency 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 window size is 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.

(morning, fog, afternoon, light, rain)

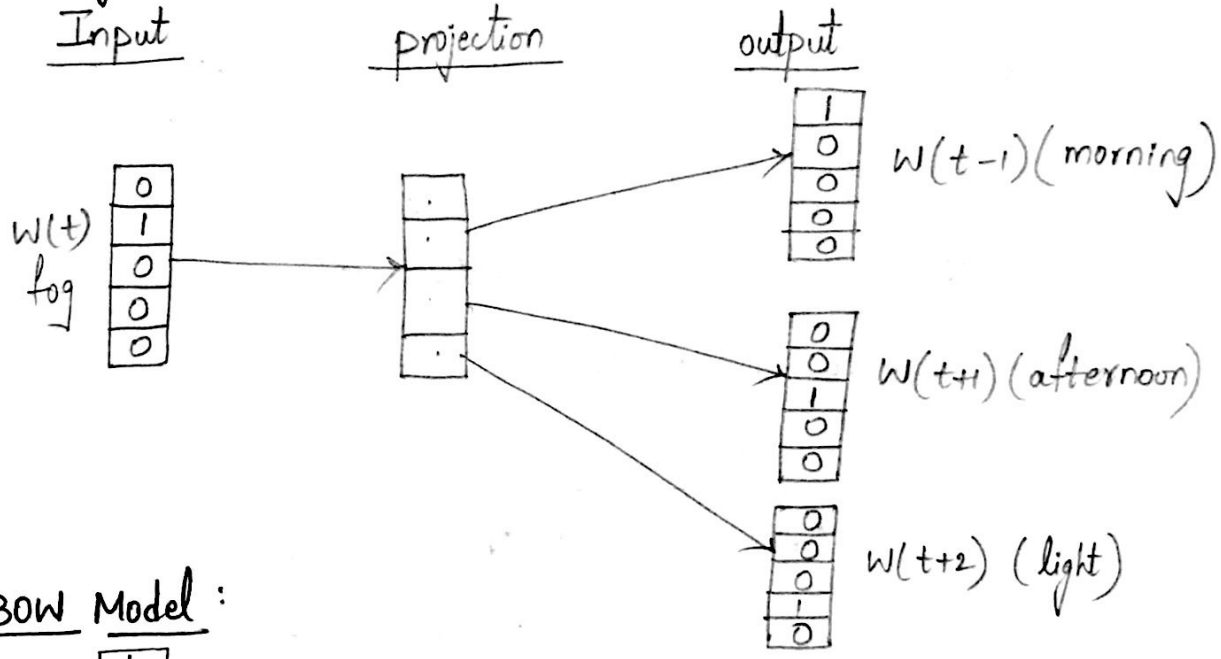
Consider input is fog then vector representation is (0,1,0,0,0)

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)



CBOW Model :