Data ^{X}

Introduction to NLP -II

(Word2vec)

Srikar Varanasi
Sana Iqbal

Review

- ❑ Goal of NLP:
  Represent and understand the meaning of the text → that is to get to the semantic level analysis

- ❑ We looked at **bag of words** model, which keeps the count of words in a document. Disadvantages:
  Loss of the ordering of words --> Ignore semantics of the words

Information about order and context of words is important to understand the document.

Discrete Representation Of Words

- We represent words in our corpus as atomic symbols i.e each word is independant.

  For Corpus : ***1. 'I love cats'*** ***2. 'I love dogs'***

  Vocabulary, V = [i, love, cats, dogs]

  If we want to represent them as numbers in our machine we assign them an id. Eg:

  I=1, love=2, cats=3, dogs=4

Vector Representation Of Words

- ❑ One-hot-Encode

  - ❑ We can represent these **words as vectors.**

We can say in the vocabulary space V = [i, love, cats, dogs]

i= [1,0,0,0]
love= [0,1,0,0]
cats= [0,0,1,0]
dogs= [0,0,0,1]

Problems

1. Vocabulary size is big.
   We will end up having very very large sized sparse vectors.

   good =[**1**,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0………]
   nice = [0,**1**,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0………]

2. We are not able to capture any semantic relations between words.

   To capture similarity between **<u>good & nice</u>** using the above vectors:

   Cosine similarity= Dot(good,nice) = 0

Word2Vec

- Word2vec is a model created by T.Mikolov in 2013
- Works on the concept of **distributional** similarity where you can find value from the context (the neighboring words)
- Build a **dense** vector called **word embeddings**

cats = [0.2, -0.4, 0.58, -0.3, -0.24, 0.8]

Distributional Representation of Words

Example:

Eating healthy is a key to fitness.

Junk eating may cause obesity .

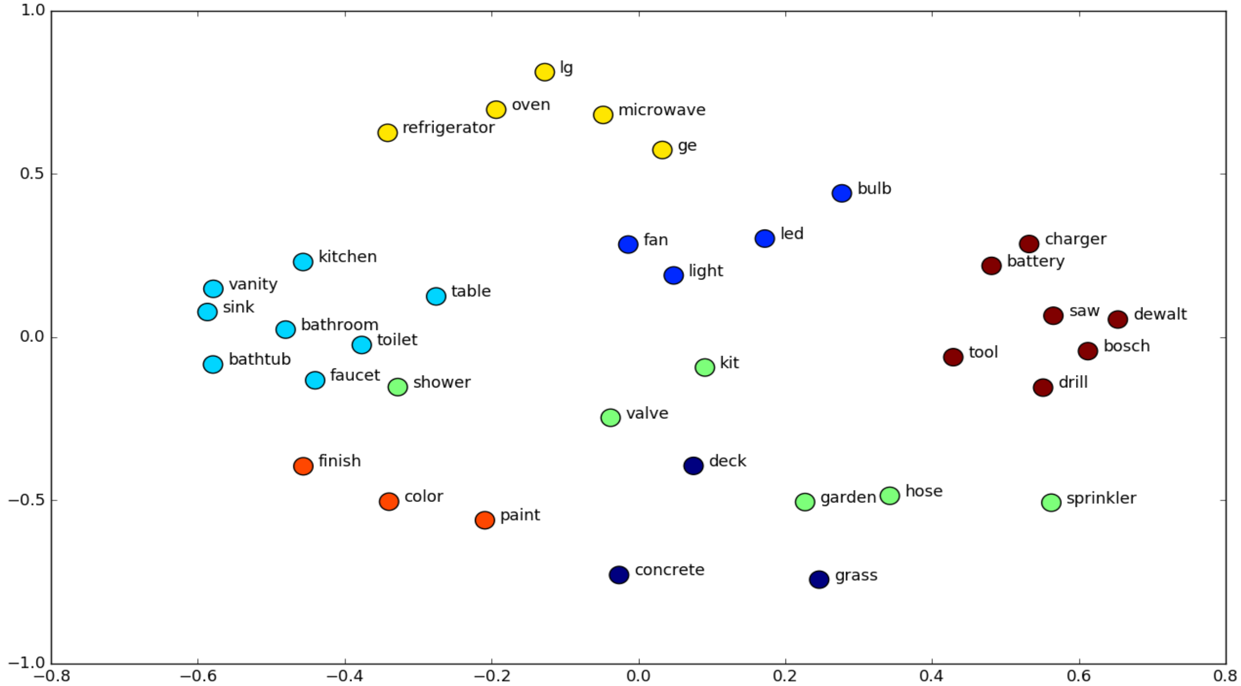
If you stop eating, you will die.

Too much eating may make you obese.

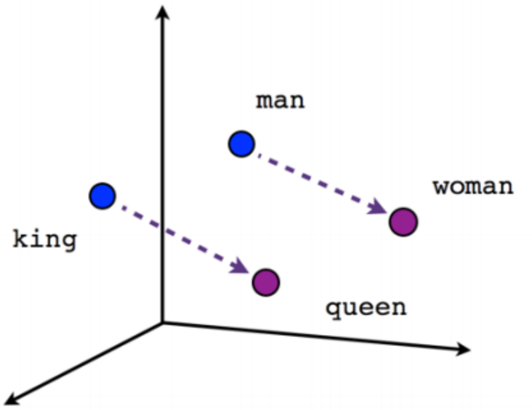
Not all cultures use spoons for eating food.

Eating seen in context of healthy, junk, food, fitness, spoons, die etc. gives the idea of its meaning**.**

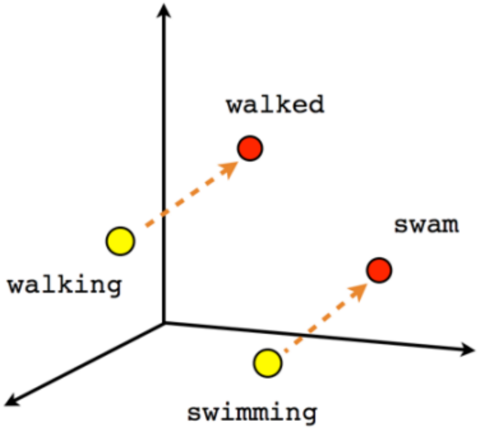
Word2Vec


lg
oven
microwave
refrigerator
ge
bulb
fan
led
charger
battery
kitchen
light
vanity
table
sink
saw
dewalt
bathroom
toilet
tool
bosch
bathtub
kit
faucet
shower
drill
valve
finish
deck
hose
garden
sprinkler
color
paint
concrete
grass
1.0
0.5
0.0
−0.5
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

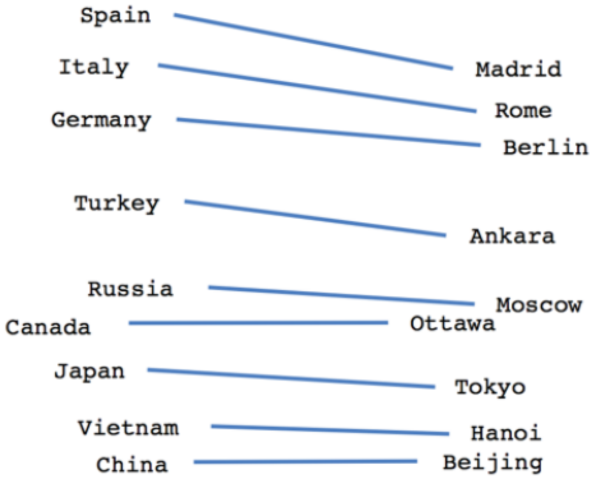
Word2Vec



Male-Female



Verb tense



Country-Capital

Word2Vec

- Neural Net Models that aim to predict contextual words/word.

- Two algorithms of word2vec:

  1. Skip-gram   2. Continuous Bag of words (Cbow)

Skip-gram:	CBOW:
the task is "***predicting the context given a word***".	the task as "***predicting the word given its context***"
- p(context words|word)	- p(word|context words)

Skip-gram model

If we have:

n = Vocabulary in the corpus=11

d = Word vector dimension=3

w= Window size on each side=1

Corpus: *the quick brown fox jumped over the lazy dog and killed it*

Output	Input
[the, brown]	quick
[quick, fox]	brown
[brown, jumped]	fox

...

Components for word2vec

- Dense representation of words – one hot encoded input
- Likelihood Function= maximize p(context word|c)

$$\prod_{w_c \in C_t} p(w_c|w_t;\theta).$$

$$p(\text{corpus};\theta) = \prod_{w_t} \prod_{w_c \in C_t} p(w_c|w_t;\theta).$$

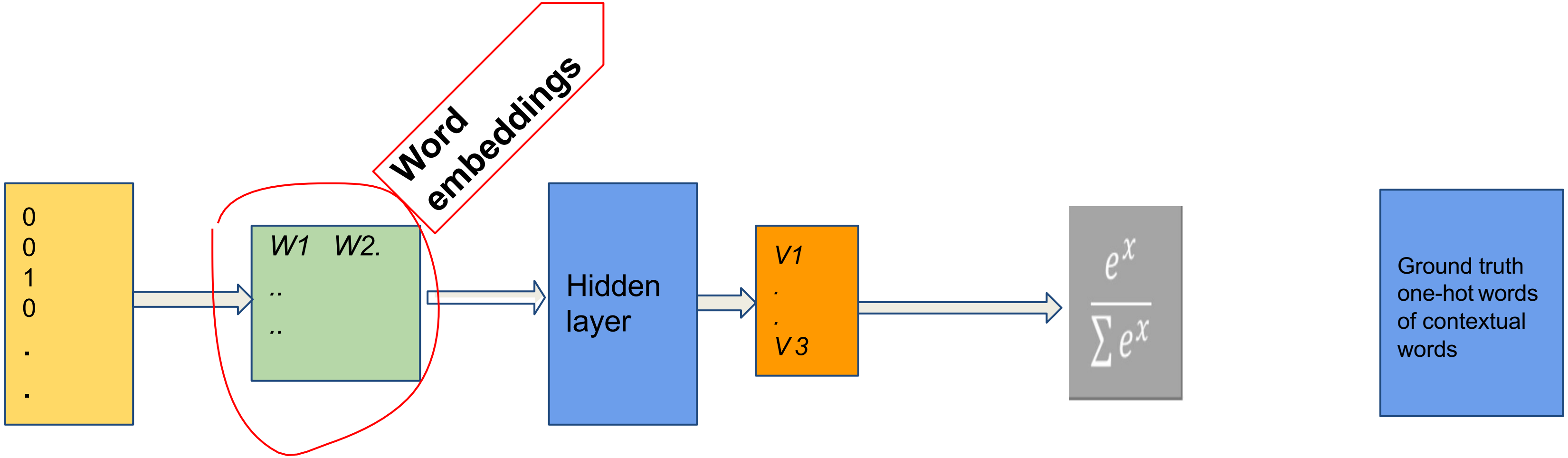
Optimization = -1/T Log (Likelihood function)

Components for word2vec

- We are using two vectors for context words and target words
- probability (c|t) = softmax (c|t) where t is target word, is c is the context word, V contains corpus of words
- Maximizing the probability

$$P(c|\,t) = \frac{exp\,(c^T.t)}{\sum_{w \in V} exp\,(c_w{}^T.t)}$$

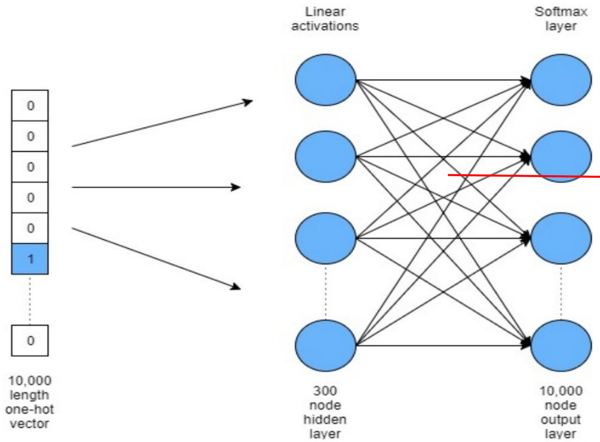
Skip gram example:



input	W	hidden layer	V	probability=sigmoid(w.V)	Actual-words
11 X 1	3 X 11	3 X 1	11 X 3	11 X 1	

Negative Sampling

- Problem with using a vanilla skipgram
  Full softmax output layer —> computationally expensive.
- When the output is a n-word one-hot vector, large number of weights that need to be updated in any gradient based training of the output layer.



Large number of weights

Negative Sampling

Instead of constructing a network that outputs a multi-class softmax layer, we change it into a simple binary classifier.

- ❑ **Input:** [*a target word* and *a real or negative context word*]
- ❑ **Output:** [0 or 1 based on *a real or negative context word]*
- ❑ An embedding layer
- ❑ Similarity Operation: To train the model to assign similar words with similar embedding vectors.
- ❑ The output sigmoid layer [0,1]

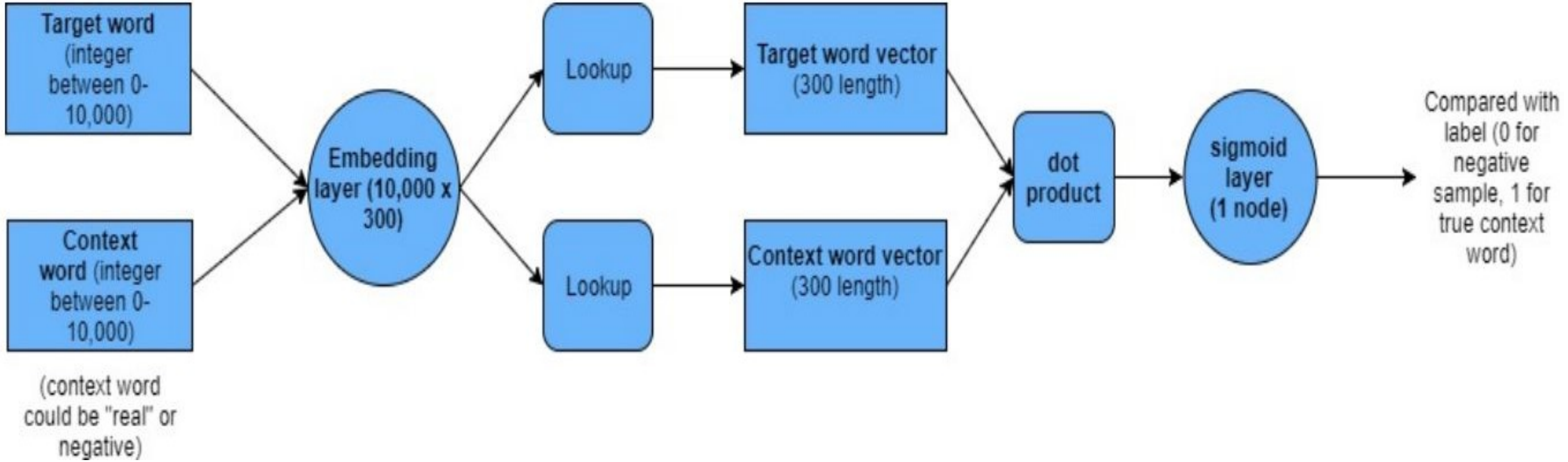
Negative Sampling

Corpus: *the quick brown fox jumped over the lazy dog*

Window size= 2 i.e 1 on each side of the input word

Input **(target word, context)**	**Output (label)**
[quick, brown]	1
[quick, dog]	0
[brown, fox]	1

Negative Sampling



Using pretrained word embeddings:

1. We can also use the word embeddings from pretrained models, eg the model trained on Google Data is available in many packages.

2. These are useful when we are working in the same domain or our own corpus is very small.