Natural Language Processing using Deep Learning

Chapman Siu, Wanming Huang

August 30, 2017

Understanding Word2Vec

Vectors allow you to measure things:

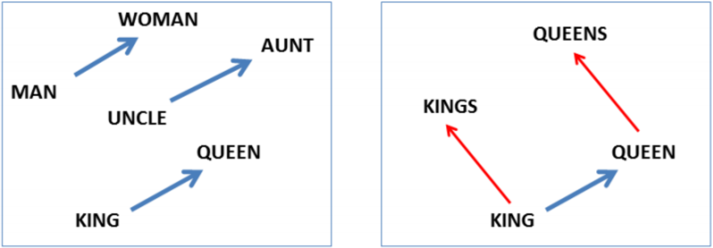
- How close vectors are (how similar)
- Are vectors orthogonal to each other (dissimilar)
- Can do "arithmetic"!

Examples from the original paper:

$$vec(King) - vec(man) + vec(woman) = vec(Queen) \tag{1}$$

$\cdots$ and many other examples.

Understanding Word2Vec



(Mikolov et al., NAACL HLT, 2013)

Understanding Word2Vec

Differences with other approaches:

- Context!

Word2Vec considers context of a word in its construction. The 2 approaches in "converting" the unsupervised problem to a supervised one:

- skip-gram: $Pr(context|targetword)$
- continuous bag of words (CBOW): $Pr(targetword|context)$

Building Word2Vec from Scratch (Building Training Set)

Skip-gram/Continuous Bag of Words (CBoW)

Training set construction:

1. Pick window size (odd number)
2. Extract all tokens based on this chosen window size
3. Remove the middle word in each window; this becomes your target word, other words are your context

Building Word2Vec from Scratch (Building Training Set)

Skip-gram(window size 3)

The cat sat on the mat

window size of 3

1. the cat sat
2. cat sat on
3. sat on the
4. on the mat

Building Word2Vec from Scratch (Building Training Set)

Skip-gram(window size 3)

The cat sat on the mat

window size of 3

context: the sat,	target: cat
context: cat on,	target: sat
context: sat them,	target: on
context: on mat,	target: the

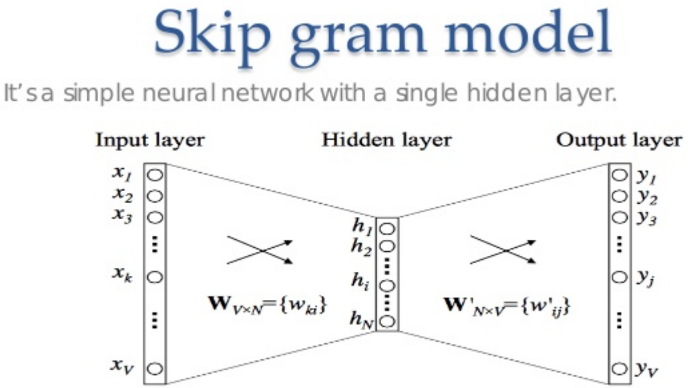
Now we can perform some supervised learning!

Building Word2Vec from Scratch (Building Training Set)

With this frame work we can create a (shallow) neural network

$$Pr(context|targetword) \tag{2}$$

Building Word2Vec from Scratch (Skip gram model)



Building Word2Vec from Scratch (Skip gram model)

Skip gram model:

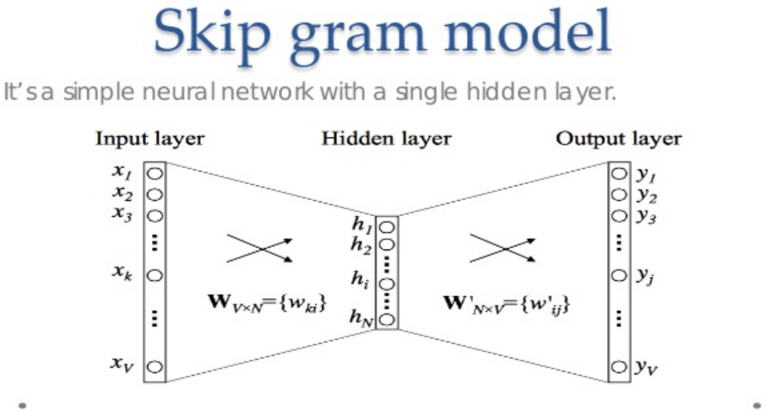
- Predict context words given a center word
- For every word w' in the model, it has 2 representations $u_{w'}$ and $v_{w'}$, one for input and one for output

Let's start with a simple one word to one word scenario

Building Word2Vec from Scratch (Skip gram model)

Question: where does the 2 word representations $u_{w'}$ and $v_{w'}$ come from?

Building Word2Vec from Scratch (Skip gram model)



- We have two weight matrix $W \in R^{V\times N}$ and $W \in R^{N\times V}$
- Let's see some codes

Building Word2Vec from Scratch (Skip gram model)

- We essentially perform calculation between a row from W and a column from W'
- the row: u_{w_t} is the input representation, or we can denote it as h
- the column: v_{w_c} is the output representation

Building Word2Vec from Scratch (Skip gram model)

w_t is the target word with input vector representation u_{w_t}
w_c is the context word with output vector representation v_{w_c}
scoring function:

$$s(w_t, w_c) = u_{w_t}^\top v_{w_c} \tag{3}$$

Building Word2Vec from Scratch (Skip gram model)

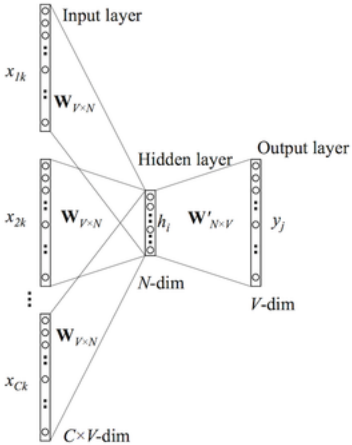
Training objective: predict the surrounding word given a center word

$$p(w_c|w_t) = \frac{exp(u_{w_t}^\top v_{w_c})}{\sum_{w'} exp(u_{w'}^\top v_{w_c})} \tag{4}$$

Building Word2Vec from Scratch (Skip gram model)

- Question: What if we now have multiple context words around a single center word?
- We will treat it as multiple one to one scenario

Building Word2Vec from Scratch (CBoW model)



Building Word2Vec from Scratch (CBoW model)

We now have a different training objective

- Predict the center word given multiple context words

Building Word2Vec from Scratch (CBoW model)

- h now is the average of input vectors
- We are then back to one to one scenario

$$p(w_c|w_t) = \frac{exp(u_{w_t}^\top v_{w_c})}{\sum_{w'} exp(u_{w'}^\top v_{w_c})} \tag{5}$$

Global Vectors for Word Representation(GloVe)

- Authors suggest that **ratios of co-occurrence probabilities** are more useful than raw probabilities
- GloVe learns word vectors through *word co-occurrences*

Global Vectors for Word Representation

- co-occurrence matrix X
- X_{ij}: how often the word i appears in the context of word j

Global Vectors for Word Representation

Loss function:

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(X_{ij})(w_i^\top w_j - log X_{ij})^2 \tag{6}$$

f is the weighting function to prevent common word pairs(X_{ij} is large) from skewing the objective too much.

FastText

A library created by Facebook research team for

1. efficient learning of word representations(Enriching Word Vectors with Subword Information)
2. sentence classification(Bag of Tricks for Efficient Text Classification)

FastText

So how is it different from Word2Vec?
Instead of words, we now have **ngrams**

1. Helpful for finding representations for rare words
2. Give the vector representations for the words not present in the dictionary

FastText

So how is it different from Word2Vec?
Instead of words, we now have **ngrams**

word *where*, $n = 3$
wh, whe, her, ere, re

FastText

We then represent a word by the num of the vector representations of all its n-grams

Given a word w, g_w is the set of n-grams appearing in w, z_g is the representation to each individual n-gram

$$s(w, c) = \sum_{g \in g_w} z_g^\top v_c \qquad (7)$$

FastText

Why subword info is used?

1. Helpful for finding representations for rare words
2. Give the vector representations for the words not present in the dictionary