Data X

Introduction to NLP -II

(Word2vec)

By Sana Iqbal

Review

- Goal of NLP:
  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 knitting'***   ***2. 'I love dogs'***

  - Vocabulary, V = [i, love, knitting, dogs]
  - If we want to represent them as numbers in our machine we assign them an id. Eg:
    I=1,   love=2, knitting=3, dogs=4

  - *But words have no ordinal relationship*

Vector Representation Of Words

- One-hot-Encode

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

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

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

Discrete Representation Of Text

- But size of One-hot-Encoded word vectors will depend on the corpus vocabulary size.

- We want you use NLP on large corpuses that exist in the world, for example the *IMDB reviews, the yelp reviews, wikipedia articles etc or* ***Google corpus:*** *a vocabulary of 3 million words -Google News dataset*

Data X

Problems in Discrete Representation Of Text

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

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

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

   To capture similarity between **good & nice** using the above vectors:

   Cosine similarity= Dot(good,nice) = 0

Distributional Representation of Words

- We want to represent a word as a vector that encodes its meaning.
- Every word is represented as a function of other words.

- To do that we rely on **Distributional similarity** concept.

  - The idea is that if we look at the different contexts in which a word appears or is used in a language, we will be able to infer its meaning.

Distributional Representation of Words cntd …

Example:

Eating healthy is a key to fitness.
Junk **eating** causes obesity.
If you stop **eating**, you will die.
Too much **eating** will 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.

Distributional Representation of Words cntd ...

JR Firth, a British linguist: ''You shall know a word by the company it keeps.''

- In **Distributional Representation**, we represent words as vectors that capture the *context* of these words in the corpus.

- **Basic Method**: We can count the number of times the word groups co-occur in the corpus. ----**CO-OCCURRENCE MATRIX**

Distributional Representation of Words --- Co-Occurrence Matrix

Corpus: I like deep learning. I like NLP. I enjoy flying.

Window size:*1*

Vocab: *I , like, enjoy, deep,learning,NLP, flying*

Co-Occurrence Matrix

counts	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

→ Vector Representation of ***like***

Issue:
- Again High dimensional vectors for a large corpus

Ref: *http://mysite.science.uottawa.ca/phofstra/MAT2342/SVDproblems.pdf*

Distributional Representation of Words --- Co-Occurrence Matrix

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

Co-Occurrence Matrix

	the	quick	brown	fox	jumped	over	lazy	dog	and	killed	it
the	0	1	0	0	0	1	1	0	0	0	0
quick	1	0	1	0	0	0	0	0	0	0	0
brown	0	1	0	1	0	0	0	0	0	0	0
fox	0	0	1	0	1	0	0	0	0	0	0
jumped	0	0	0	1	0	1	0	0	0	0	0
over	1	0	0	0	1	0	1	0	0	0	0
lazy	1	0	0	0	0	1	0	1	0	0	0
dog	0	0	0	0	0	0	1	0	1	0	0
and	0	0	0	0	0	0	0	1	0	1	0
killed	0	0	0	0	0	0	0		1	0	1
it	0	0	0	0	0	0	0	0	0	1	0

→ Vector Representation of ***the***

Issue:

- High dimensional vectors for a large corpus

WORD2VEC

By Mikolov etal.

- WORD2VEC is a method of creating distributional representations of words called **word embeddings**, using backpropagation.

- Eg: apple $=[0.2, 0.35, 0.1, -0.2, 0.15, 0.4]^T$

Review: A simple Feed Forward Neural Net

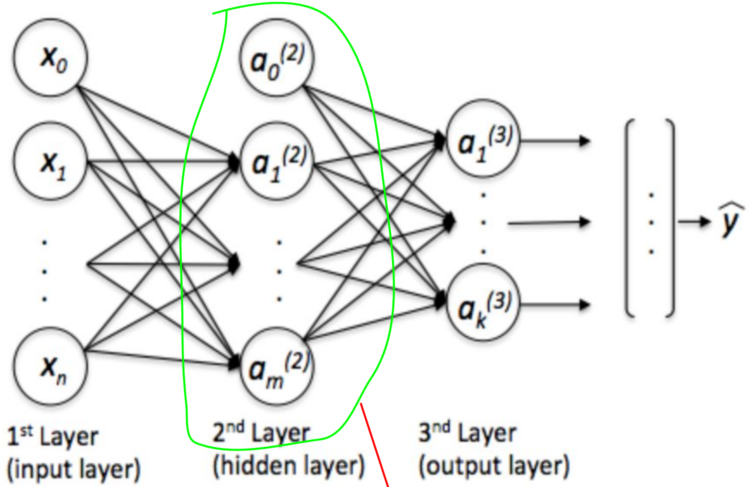
1. Accepts the Input.

2. Creates different combinations of weighted input

3. Induces Non-linearity in the input using Activations.

4. Uses backpropagation to update weights.

5. Uses the trained model to give Output on test data



Schematic of a multi-layer perceptron.

NOTE: *These are based on the combination of inputs*

WORD2VEC

- Neural Net Models that aim to predict contextual words/word of the input 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)

<u>WORD2VEC</u>

Input : word(s) Output: contextual word(s)

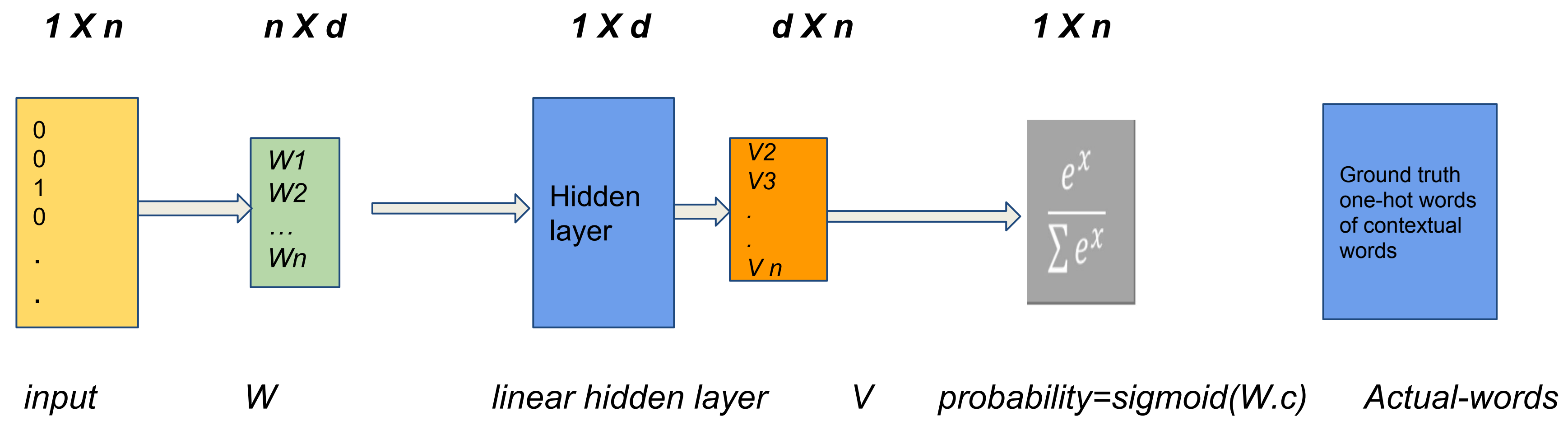
Like any other Neural Network model:

1. Word2vec Model is parametrized by weights.
2. Trained using a loss/ objective function
3. Weights are tuned to minimize loss BETWEEN *<u>actual</u> contextual word* and *<u>predicted</u> contextual word.*

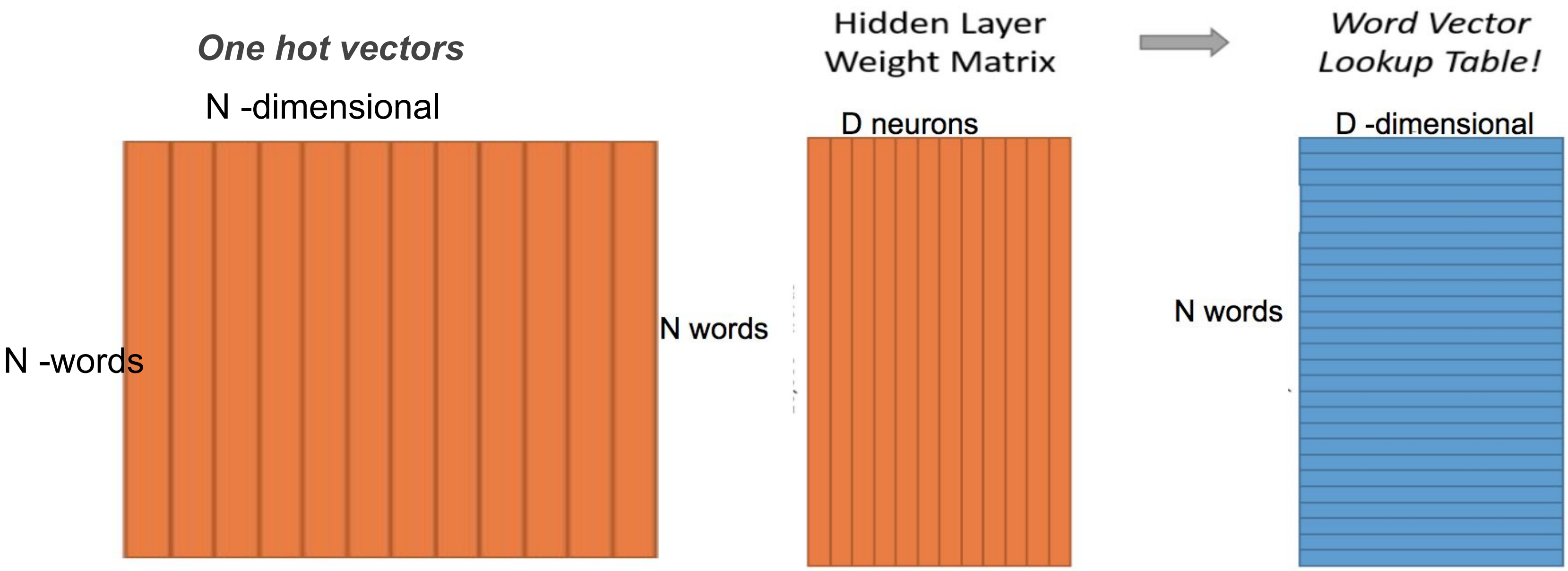
SPECIAL: These parameter/weight matrix between input and hidden layer of the model is called **'word vector representations'** of words.

WORD2VEC - Skip Gram Model

- **Input:** One hot encoded word, **c**
- Weight Matrix: **V** and **W**
- Objective Function= maximize p(context word|c)
- probability(x_i|c) = softmax($\mathbf{x_i.c}$)



One hot to Word2vec



Skip gram example data:

If we have:

n = Vocabulary in the corpus= 11

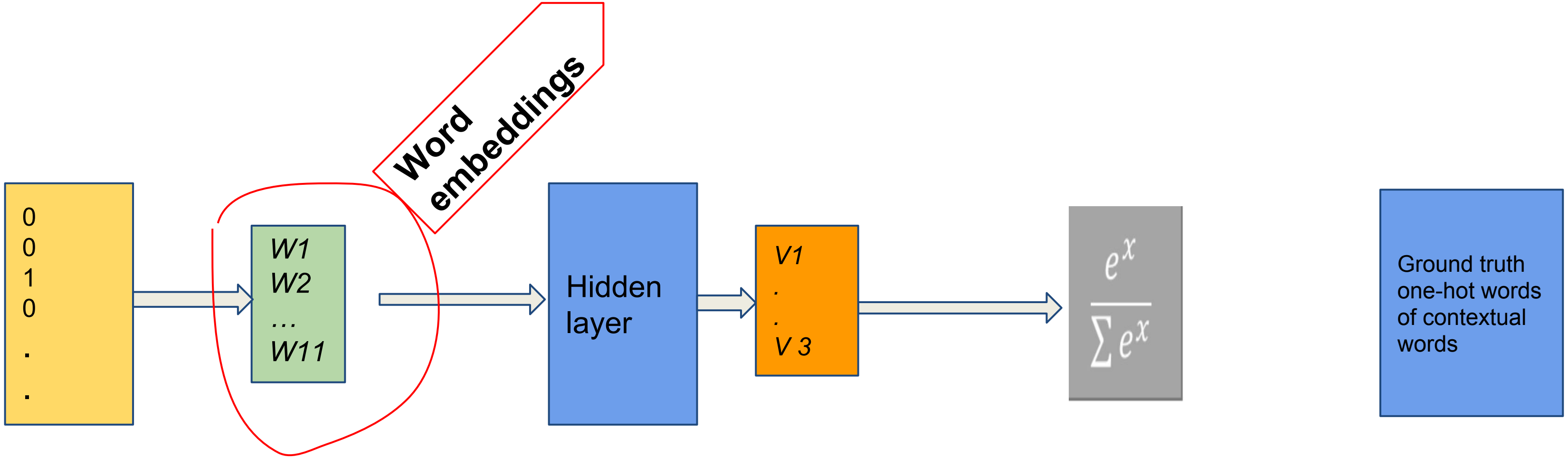
d = Word vector dimension=3

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
...	

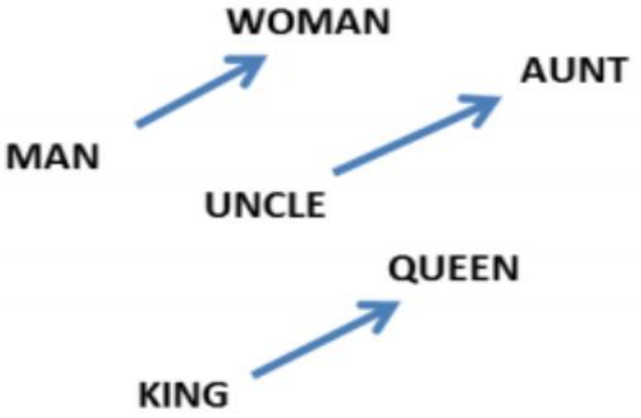
Skip gram example data:



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

Data X

Results with word2vec in the original paper trained on Google news dataset

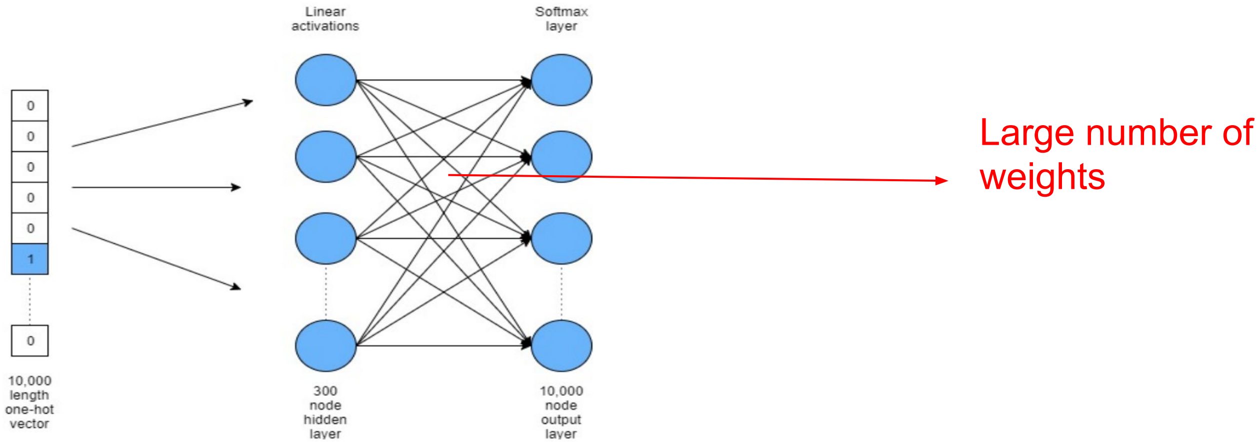


From Mikolov *et al.* (2013a)

More..

Negative Sampling Introduction

- ❑ Problem with using a vanilla skip gram
  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.



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]

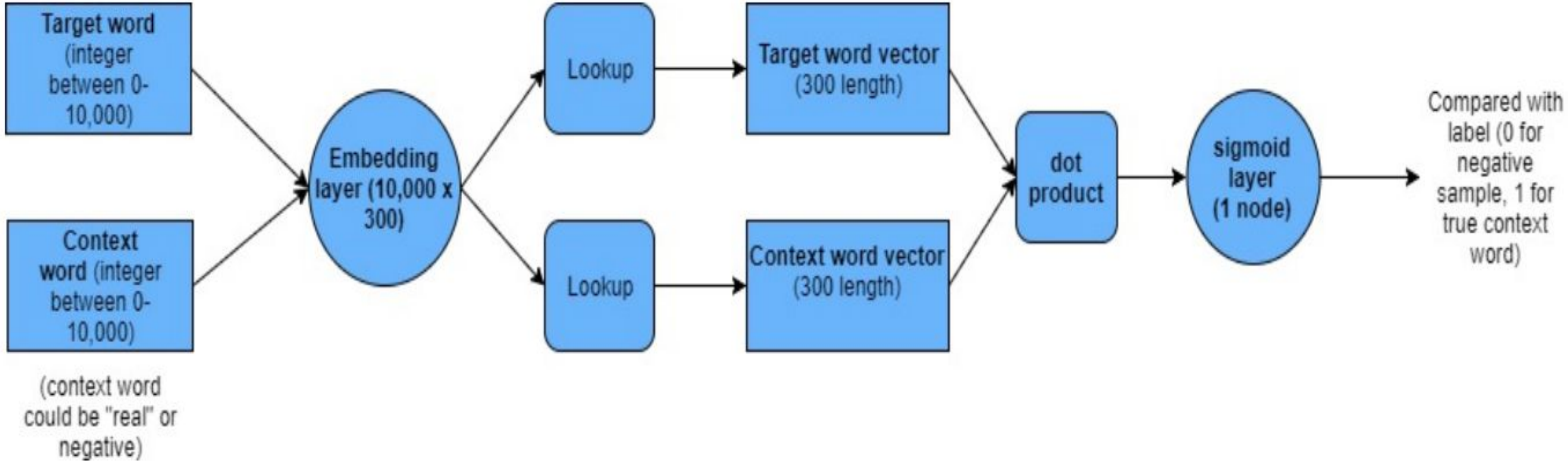
Negative Sampling

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

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.