A word is worth a thousand vectors

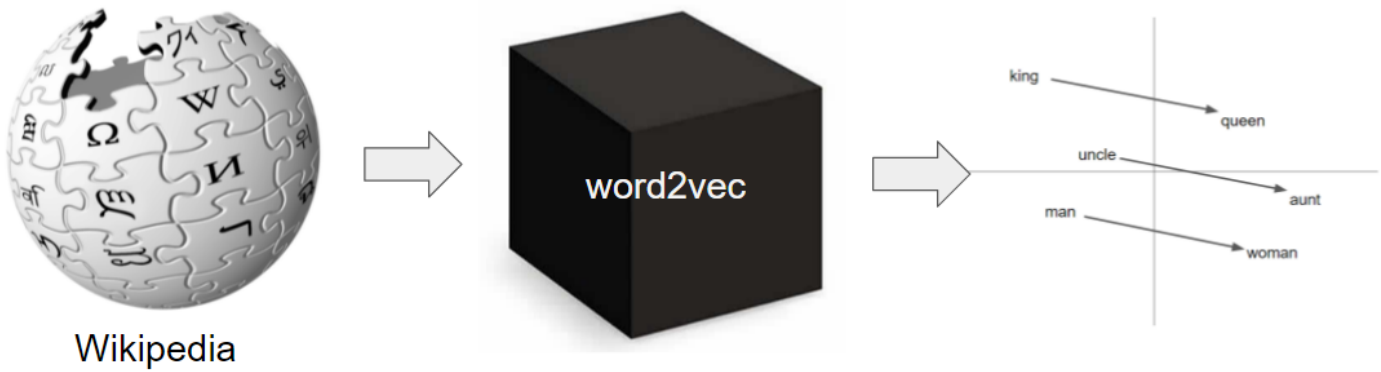
Pramod Srinivasan

June 9, 2016

Who's this guy?

- Pramod Srinivasan
- Machine Learning Engineer(MLE) Intern at Microsoft
- MS* in CS @ UIUC
- Deep Reinforcement Learning Enthusiast
- www.github.com/domarps
- www.linkedin.com/in/domarps

Why word2vec?



word2vec : An introduction

What is word2vec?

word2vec is a two-layer neural network that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While word2vec is not a deep neural network, it turns text into a numerical form that deep neural networks can understand.

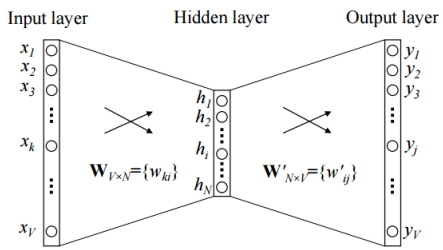


Figure: A simple ☺ Bag of Words model

No, seriously, what is word2vec?

What is a word?

“apple”

“juice”

What is a word?

“apple”	“juice”
2	299,999

What is a word?

"apple"

2

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

"juice"

299,999

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$$

What is a word?

"apple"

2

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Sparsity!

"juice"

299,999

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$$

Word Represented as an Identity

"apple"

Word Represented as an Identity

“apple”

2

“juice”

Word Represented as an Identity

“apple”	“juice”
2	299,999

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Word Represented as an Identity

"apple"	"juice"
2	299,999
$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

Understanding how humans understand words

Understanding how humans understand words

– Apple

Understanding how humans understand words

- Apple
- Orange

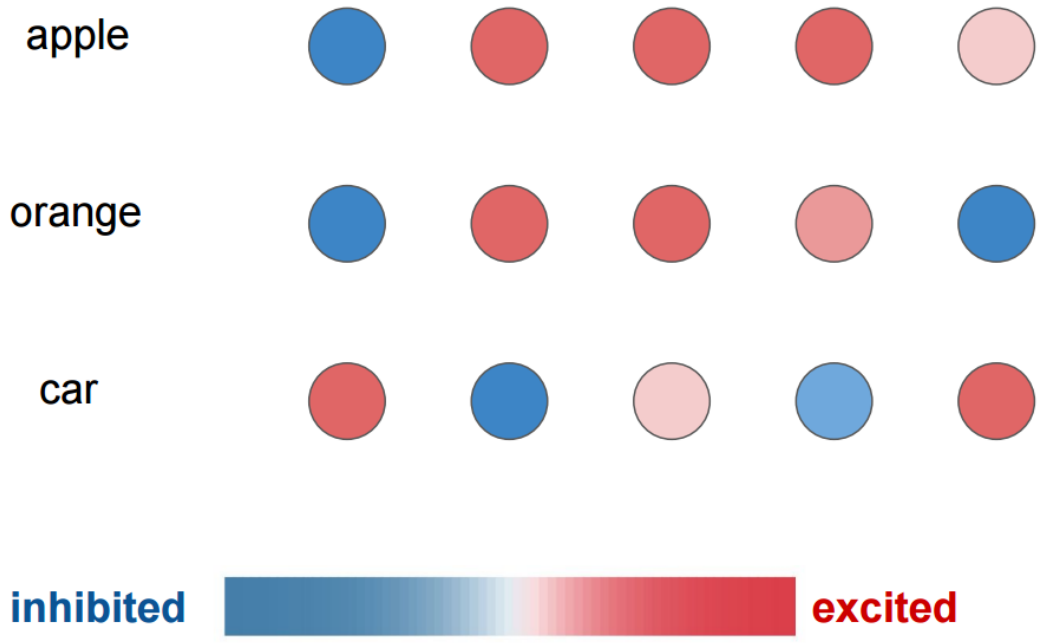
Understanding how humans understand words

- Apple
- Orange
- Car

Understanding how humans understand words

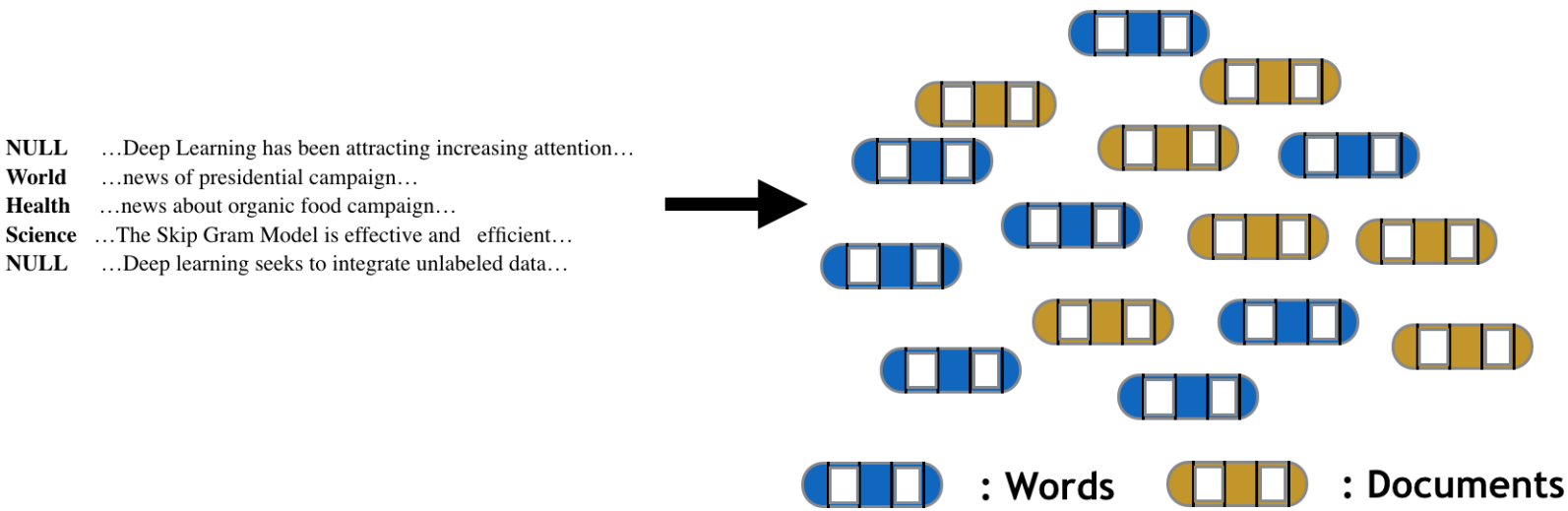
- Apple
- Orange
- Car

Word Represented as a Continuous level of activations



Solving the Problem of Data Sparseness($|V|$)

- representing words and documents in a reduced parameter space
- improving generalization
- somehow transfer or share knowledge between words



The Distributional Hypothesis

> "You shall know a word by the company it keeps"
>
> — (J. R. Firth, 1957)

Words with **high similarity** occur in the **same contexts** as one another.

- A **word** ought to be able to **predict its context**
- A **context** ought to be able to **predict its missing word**

Distributed Representations

- "secret sauce" for the success of many NLP systems
- Not treating words as blocks rather as relationships
- Comes pre-trained

Distributed Representations

- "secret sauce" for the success of many NLP systems
- Not treating words as blocks rather as relationships
- Comes pre-trained

$$\text{king - man} + \text{woman} = \text{queen}$$

Distributed Representations

- "secret sauce" for the success of many NLP systems
- Not treating words as blocks rather as relationships
- Comes pre-trained

$$\text{king - man} + \text{woman} = \text{queen}$$

The input was :

- ~~a dictionary~~

Distributed Representations

- "secret sauce" for the success of many NLP systems
- Not treating words as blocks rather as relationships
- Comes pre-trained

$$\text{king - man} + \text{woman} = \text{queen}$$

The input was :

- ~~a dictionary~~
- ~~a network of relationships between words~~

Distributed Representations

- "secret sauce" for the success of many NLP systems
- Not treating words as blocks rather as relationships
- Comes pre-trained

$$\text{king - man} + \text{woman} = \text{queen}$$

The input was :

- ~~a dictionary~~
- ~~a network of relationships between words~~
- ~~a network of relationships between humans~~

Distributed Representations

- "secret sauce" for the success of many NLP systems
- Not treating words as blocks rather as relationships
- Comes pre-trained

king - man + woman = queen

The input was :

- ~~a dictionary~~
- ~~a network of relationships between words~~
- ~~a network of relationships between humans~~
- huge mountain of Text

word2vec : Objective function

Objective :
learn a n-dimensional vector for the word from its surrounding context

word2vec : Objective function

Objective :
learn a n-dimensional vector for the word from its surrounding context

Three Steps :

1. Set up an objective function
2. Randomly initialize vectors
3. Do gradient descent

word2vec : Learning Algorithm

"The quick brown fox jumped over the lazy dog"

Maximize the likelihood of seeing this **context** given the word over

$$P(the|over)$$

$$P(quick|over)$$

$$P(brown|over)$$

$$P(fox|over)$$

$$P(jumped|over)$$

$$P(the|over)$$

$$P(lazy|over)$$

$$P(dog|over)$$

What should this be?

$$P(fox|over)$$

word2vec : Learning Algorithm(Contd.)

Two Key Observations:

- Assign two vectors for every word - input and output.

word2vec : Learning Algorithm(Contd.)

Two Key Observations:

- Assign two vectors for every word - input and output.
- Define a context window around every word.

word2vec : Learning Algorithm(Contd.)

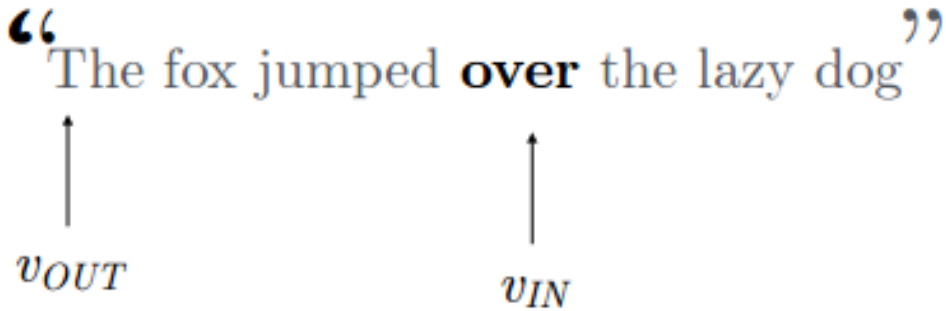
$$P(v_{OUT}|v_{IN})$$

"The fox jumped **over** the lazy dog"

v_{IN}

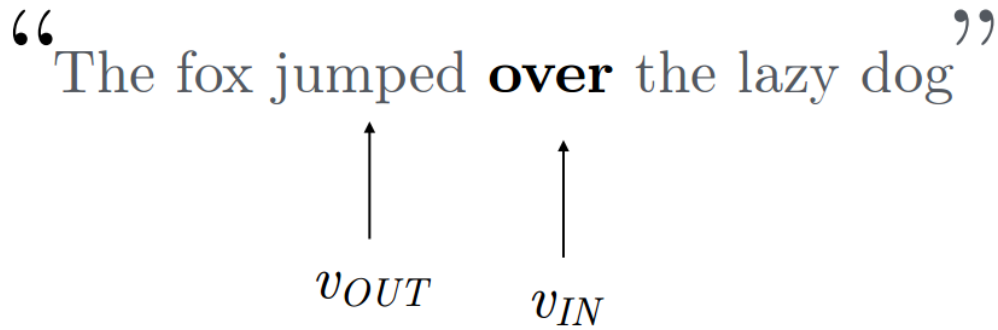
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



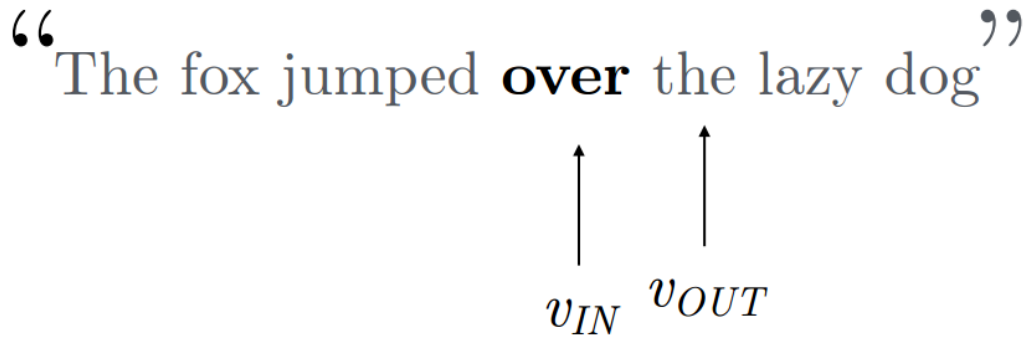
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



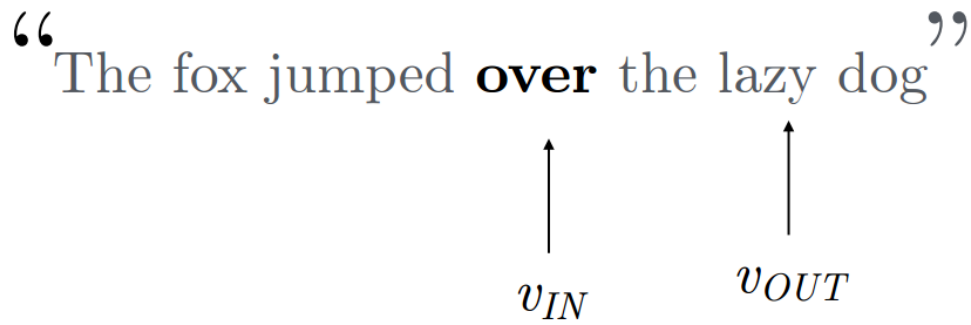
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



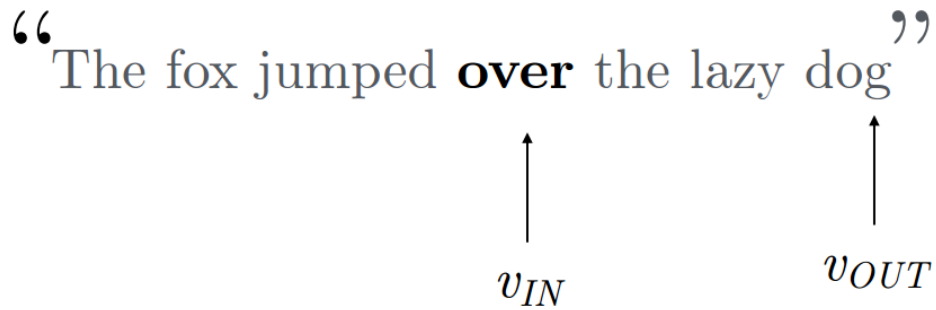
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



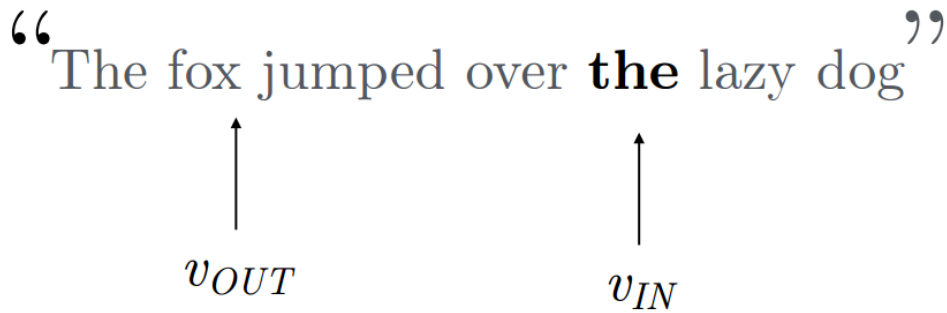
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



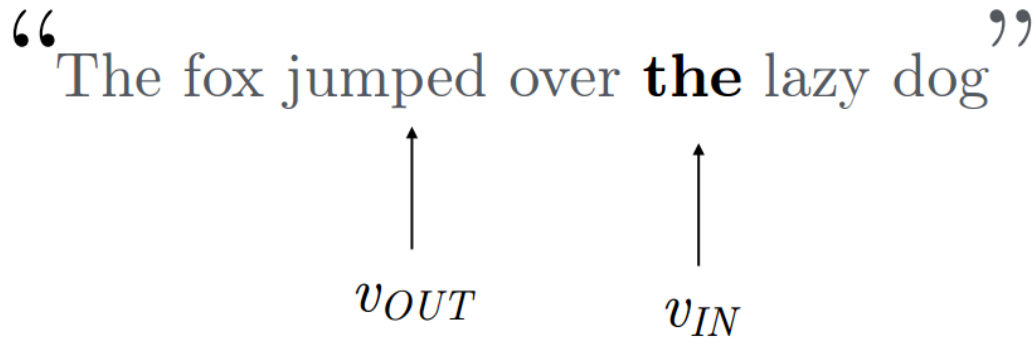
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



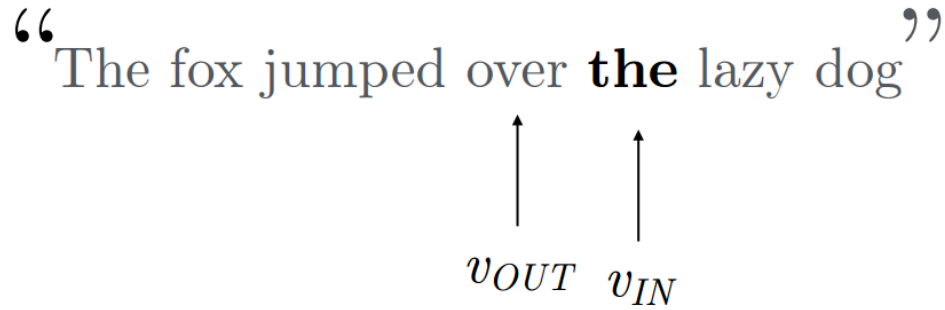
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



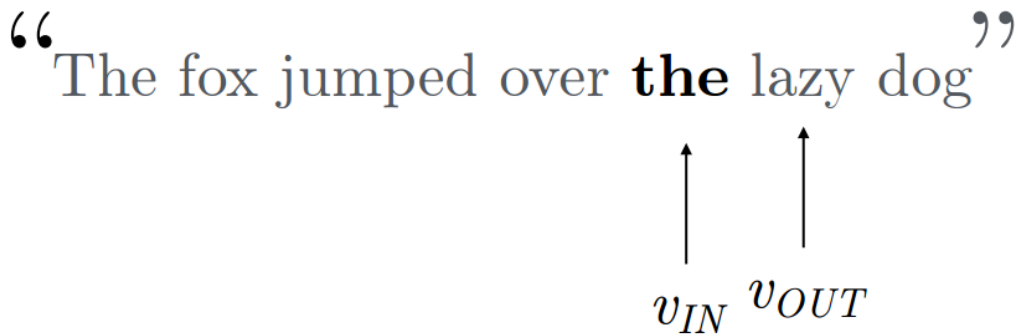
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



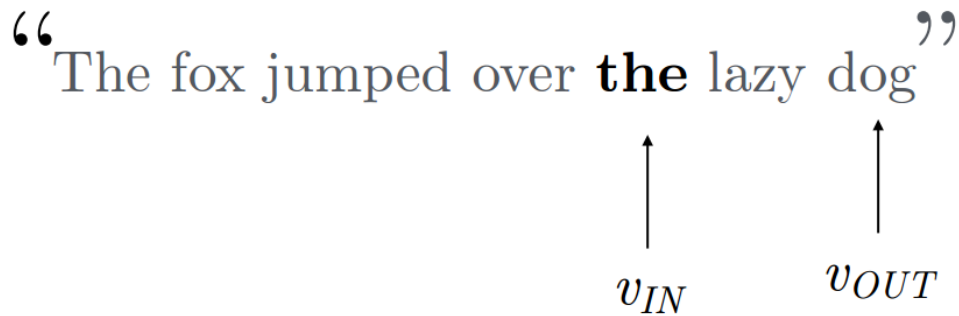
word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



word2vec : Learning Algorithm(Contd.)

$$P(v_{OUT}|v_{IN})$$



Measuring Loss

$$P(v_{OUT}|v_{IN})$$

Measuring Loss

$$P(v_{OUT}|v_{IN})$$

Measure loss between v_{OUT} and v_{IN}

$$v_{OUT} \cdot v_{IN}$$

Measuring Loss

$$P(v_{OUT}|v_{IN})$$

Measure loss between v_{OUT} and v_{IN}

$$v_{OUT} \cdot v_{IN}$$

Well, we had wanted to quantify probability but,

$$v_{OUT} \cdot v_{IN} \in [-1, 1]$$

Measuring Loss

$$P(v_{OUT}|v_{IN})$$

Measure loss between v_{OUT} and v_{IN}

$$v_{OUT} \cdot v_{IN}$$

Well, we had wanted to quantify probability but,

$$v_{OUT} \cdot v_{IN} \in [-1, 1]$$

So, we resort to a softmax

$$softmax(v_{OUT} \cdot v_{IN}) \in [0, 1]$$

softMax: The kernel of word2vec

$$softmax(v_{OUT} \cdot v_{IN})$$

softMax: The kernel of word2vec

$$softmax(v_{OUT} \cdot v_{IN})$$

– Probability of choosing 1 of N discrete items

softMax: The kernel of word2vec

$$softmax(v_{OUT} \cdot v_{IN})$$

- Probability of choosing 1 of N discrete items
- Mapping from vector space to a multinomial over words

softMax: The kernel of word2vec

$$softmax(v_{OUT} \cdot v_{IN})$$

- Probability of choosing 1 of N discrete items
- Mapping from vector space to a multinomial over words

softMax: The kernel of word2vec

$$softmax(v_{OUT} \cdot v_{IN})$$

- Probability of choosing 1 of N discrete items
- Mapping from vector space to a multinomial over words

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

softMax: The kernel of word2vec

$$softmax(v_{OUT} \cdot v_{IN})$$

- Probability of choosing 1 of N discrete items
- Mapping from vector space to a multinomial over words

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

Here, $\sum_{k \in V} exp(v_{in}.v_k)$ is the normalization term over all words.

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)} = P(v_{OUT}|v_{IN})$$

Learn by gradient descent on the softmax probability

Learn by gradient descent on the softmax prob.

For every example we see update v_{in}

$$v_{in} := v_{in} + \frac{\partial}{\partial v_{in}} P(v_{out}|v_{in})$$

$$v_{out} := v_{out} + \frac{\partial}{\partial v_{out}} P(v_{out}|v_{in})$$

word2vec – Performance

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

– V operations for every update, for $|V|$ dimensional vector

word2vec – Performance

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

- V operations for every update, for $|V|$ dimensional vector
- VC operations per input word for context of size C

word2vec – Performance

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

- V operations for every update, for $|V|$ dimensional vector
- VC operations per input word for context of size C
- VCN over the whole corpus of N words

word2vec – Performance

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

- V operations for every update, for $|V|$ dimensional vector
- VC operations per input word for context of size C
- VCN over the whole corpus of N words

Naive word2vec's O(VCN) v/s SVD's O(NV^2)

word2vec – Performance

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

- V operations for every update, for $|V|$ dimensional vector
- VC operations per input word for context of size C
- VCN over the whole corpus of N words

Naive word2vec's O(VCN) v/s SVD's O(NV^2)

Can we do better?

Addressing the Performance problem

Have an O(V) Problem?

Solution : Build a tree and get a O(logV) problem!!

Hierarchial Softmax

word2vec refinement : Hierarchial Softmax

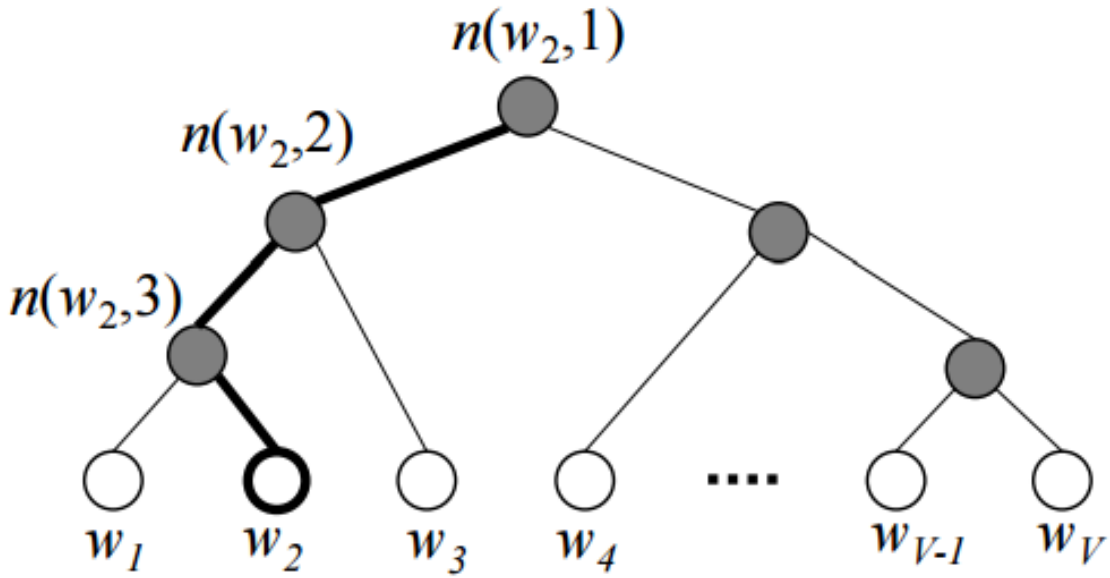


Figure: An example Binary Tree for hierarchial softmax

Just like Star Wars we have:

- Light Nodes are the words in the dictionary
- Dark Nodes are the decision nodes

Note : $n(w, j)$ means the j-th unit on the path from root to the word w

word2vec refinement : Performance Improvement

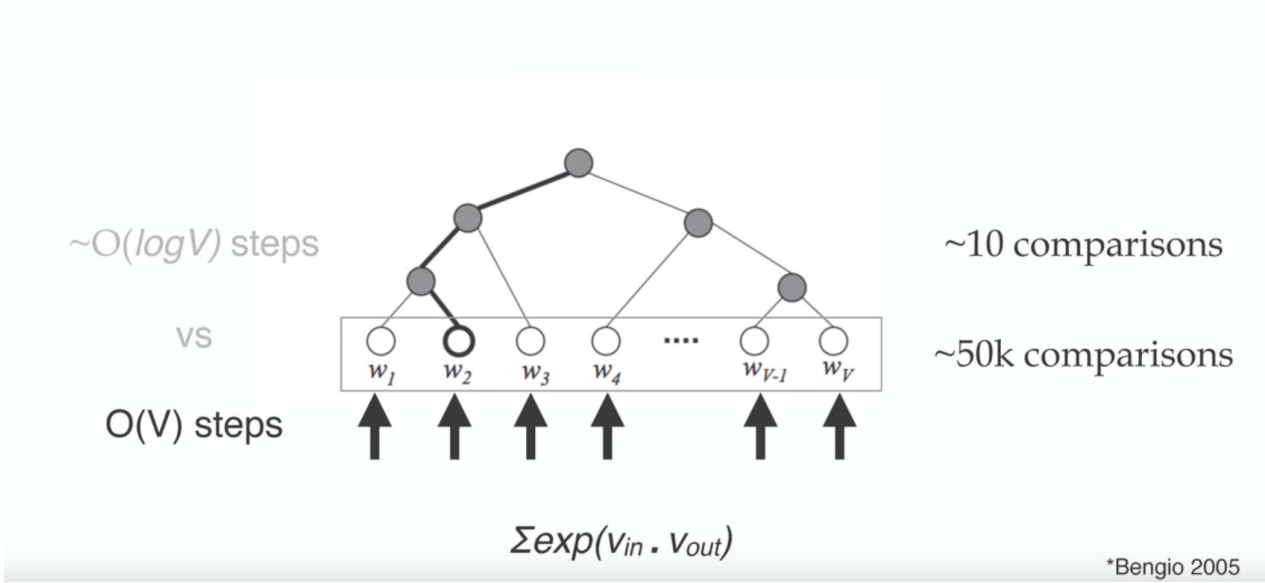
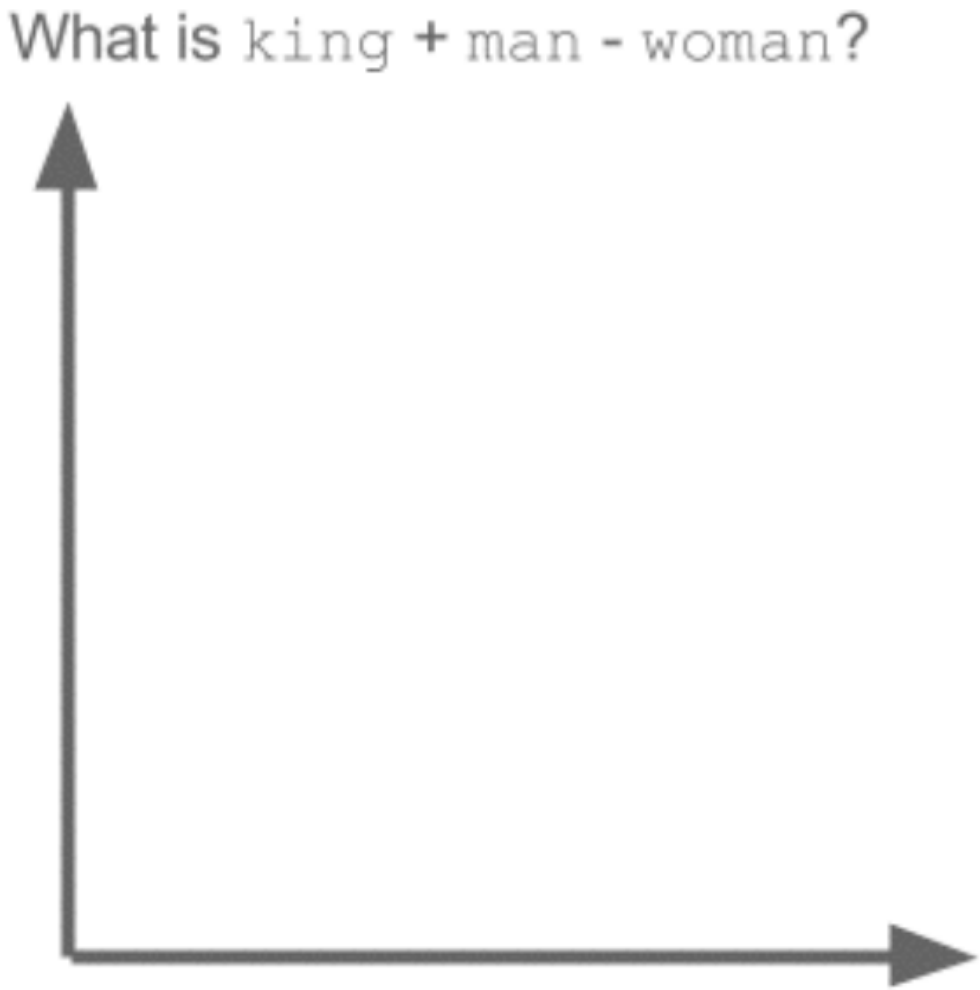


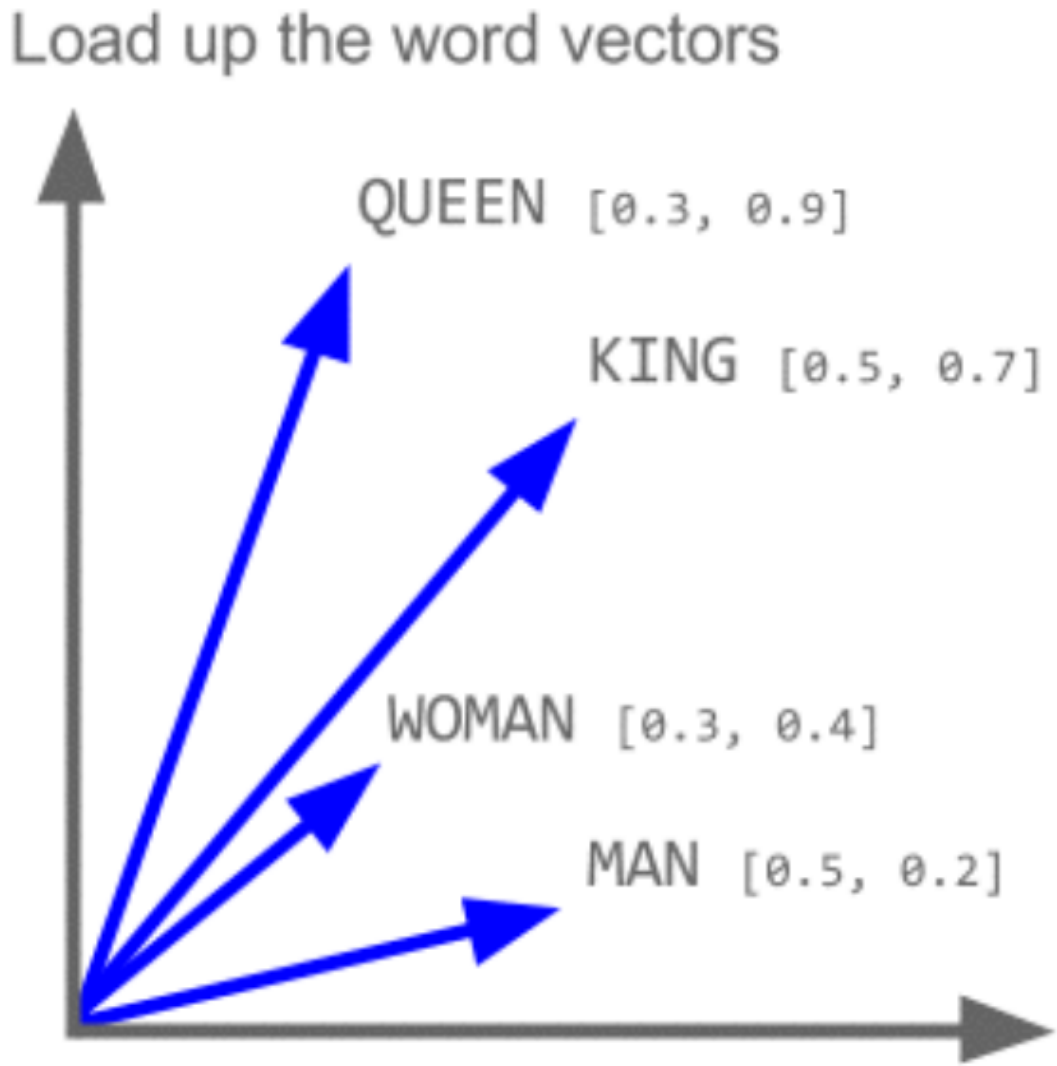
Figure: From O(V) to O(logV)

O(CNlog(V)) : license to scale and consume enormous amounts of data

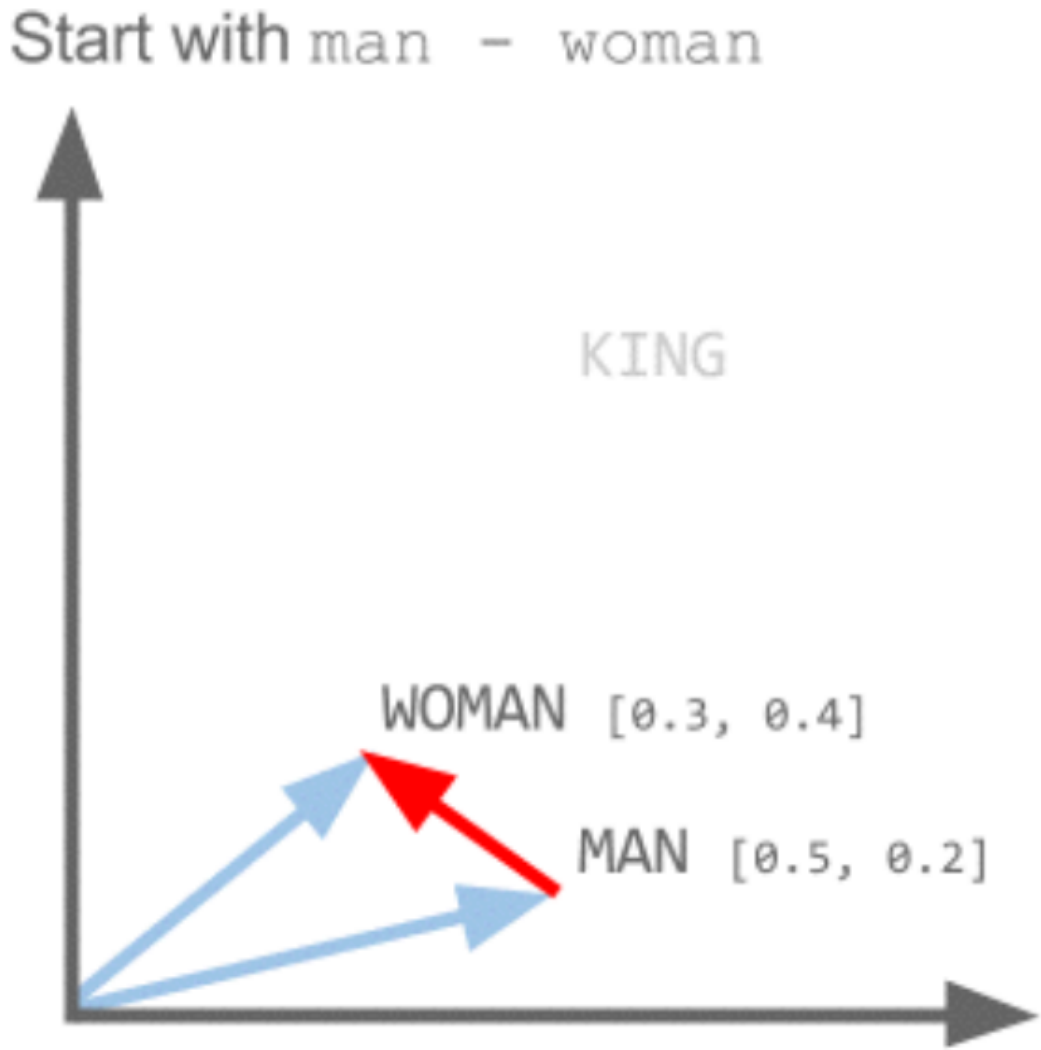
word2vec : nuts and bolts



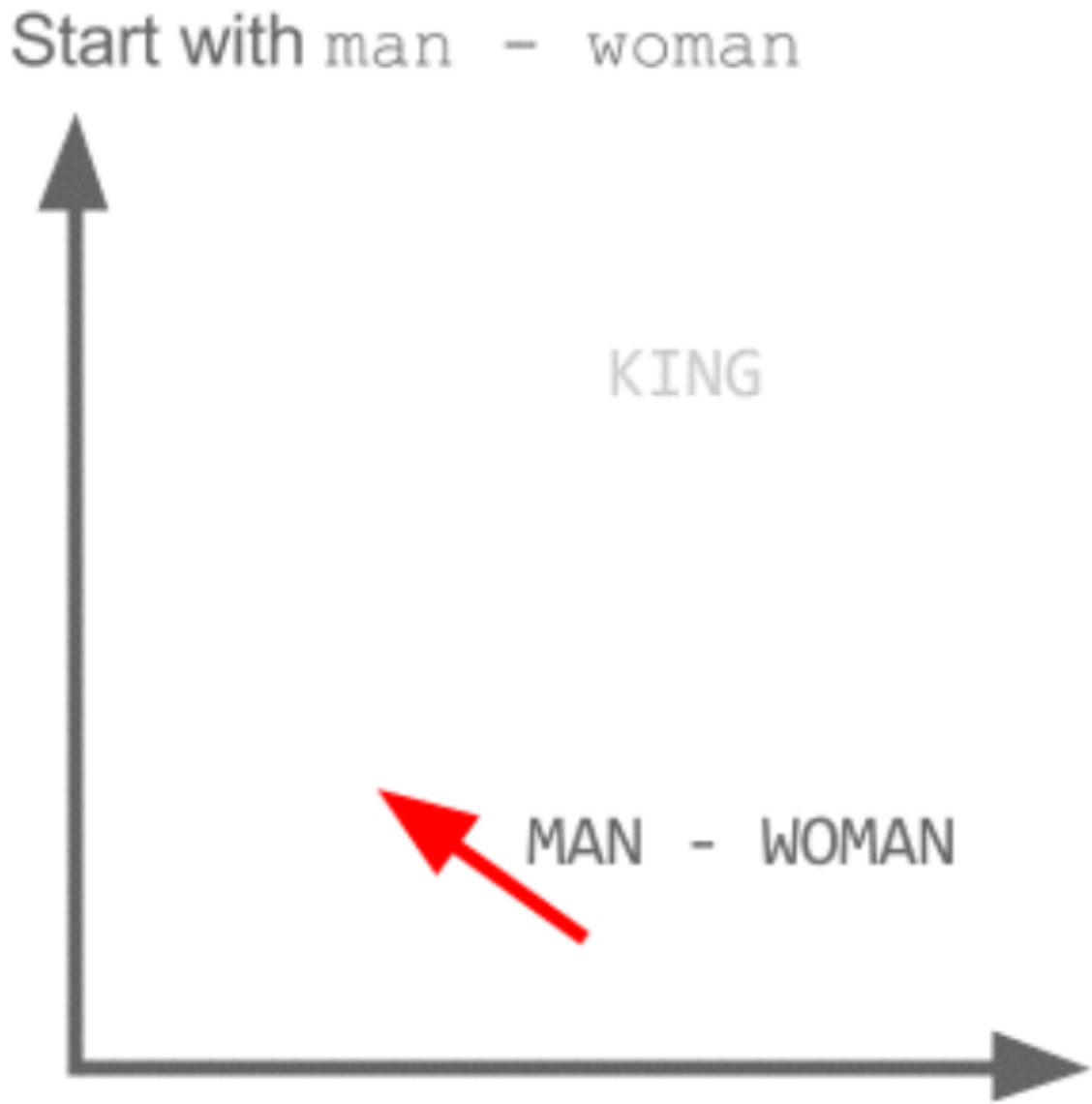
word2vec : nuts and bolts



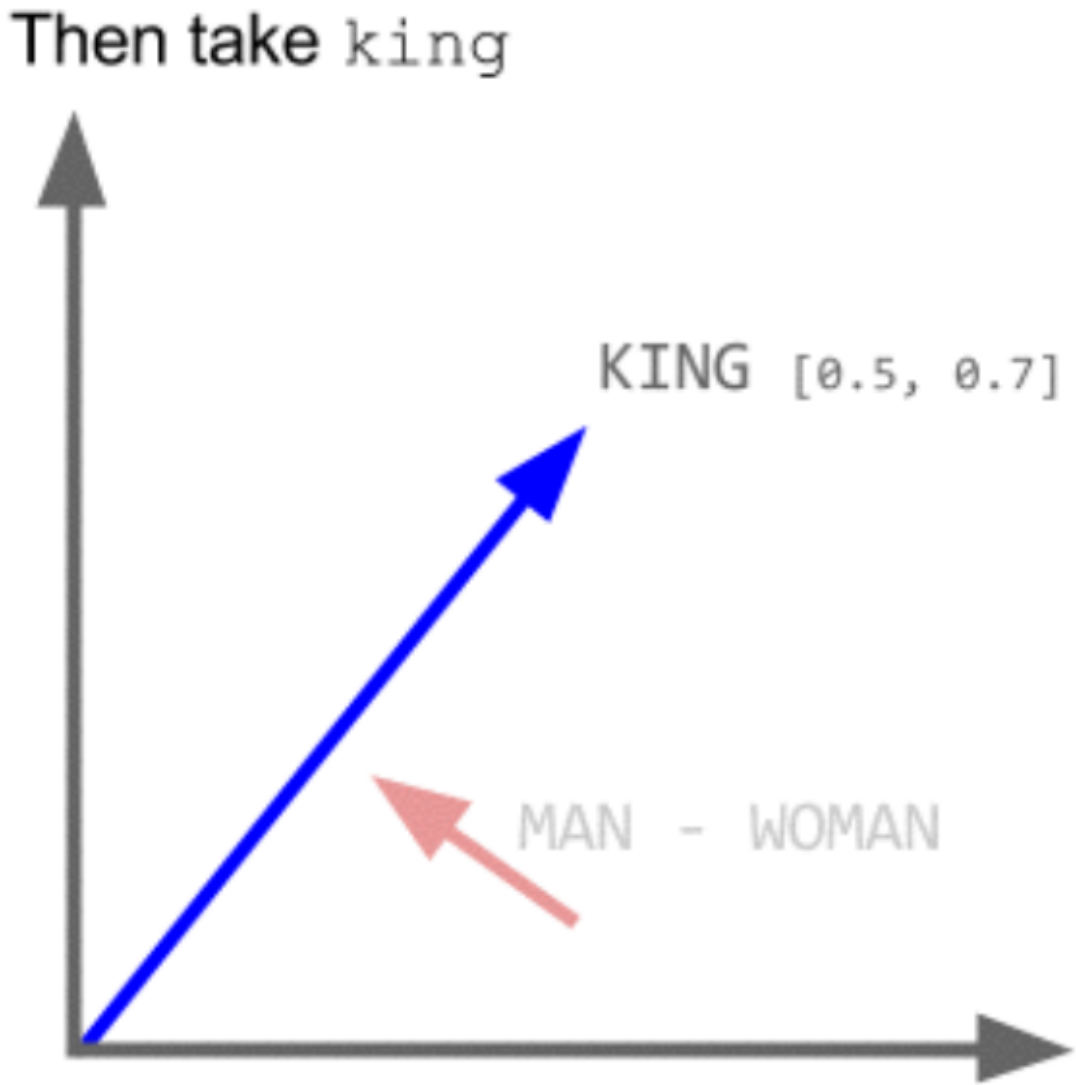
word2vec : nuts and bolts



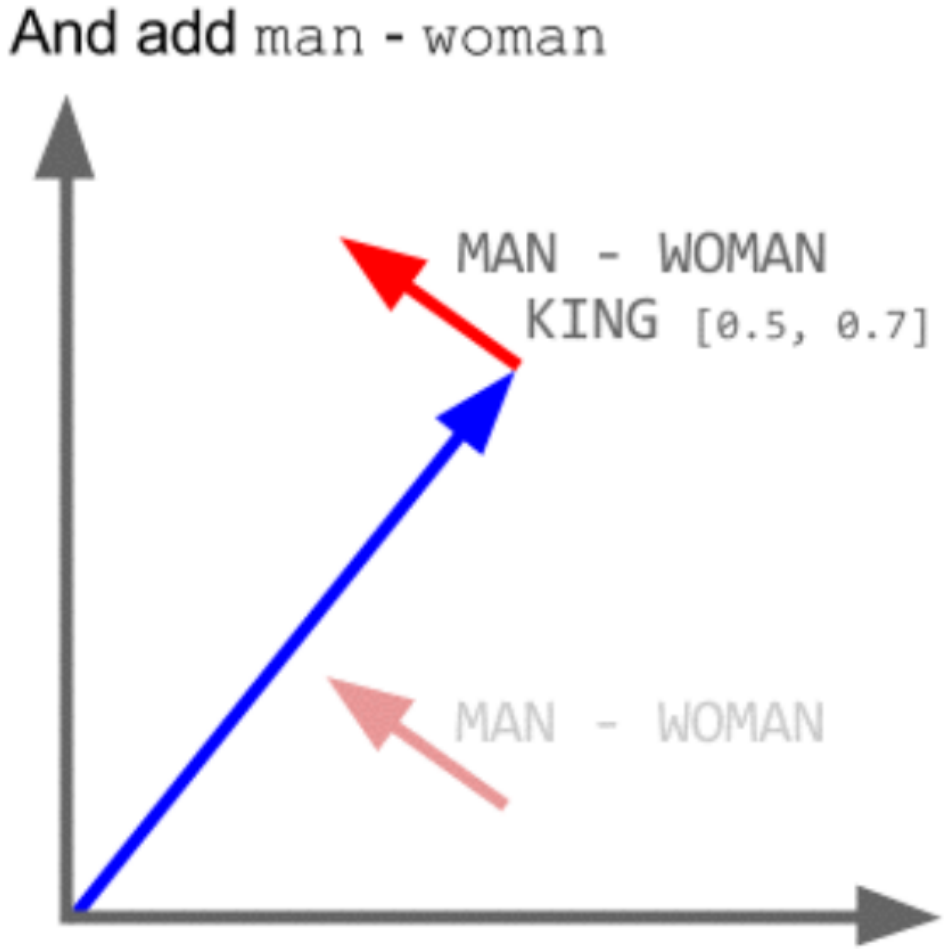
word2vec : nuts and bolts



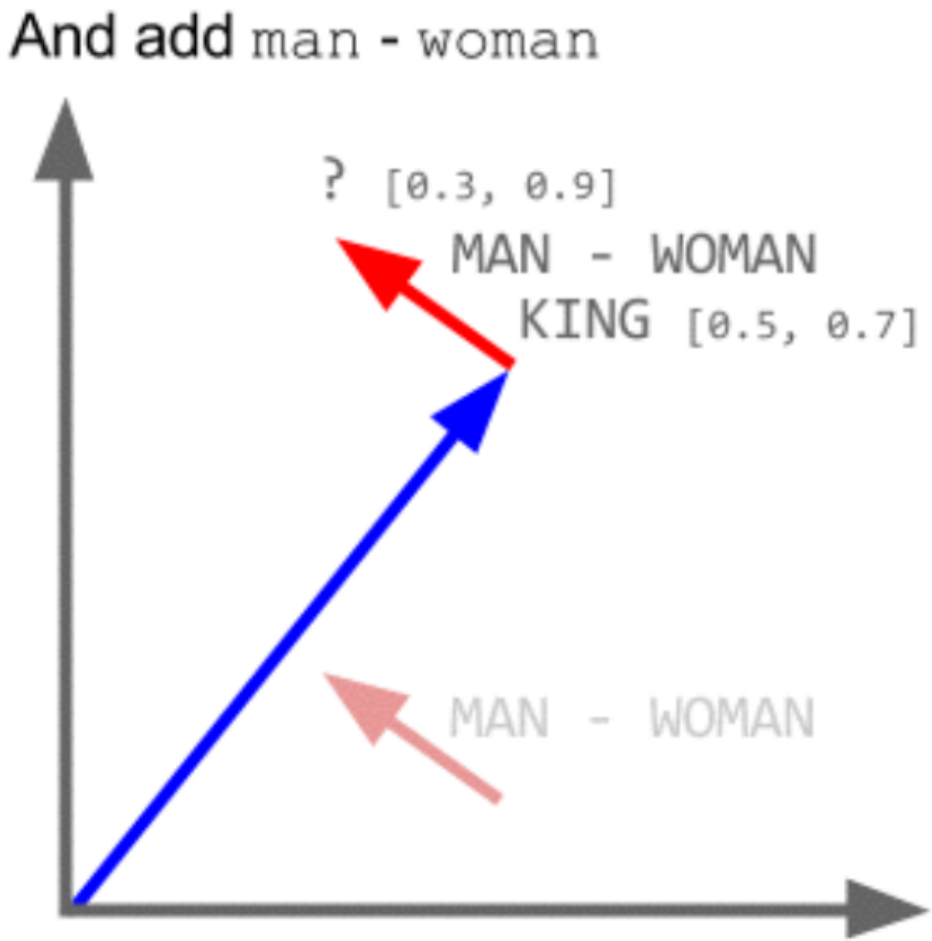
word2vec : nuts and bolts



word2vec : nuts and bolts

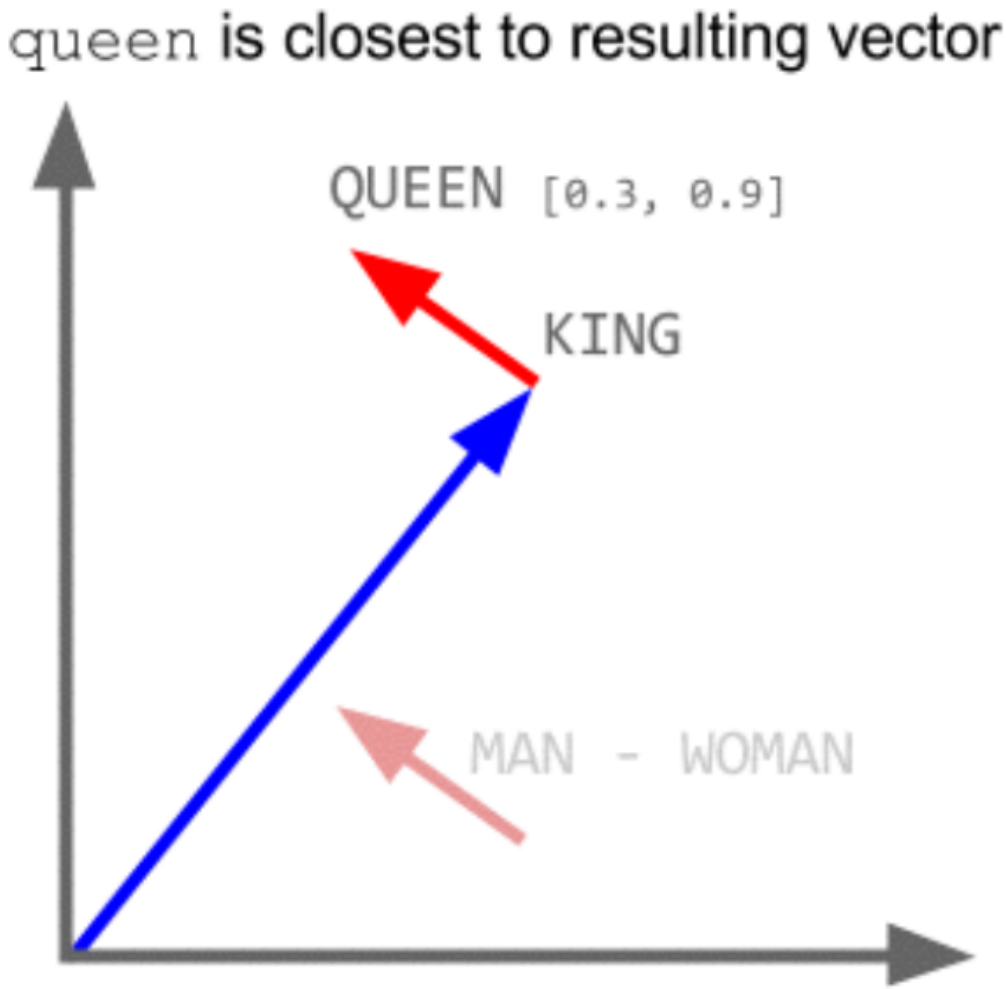


word2vec : nuts and bolts

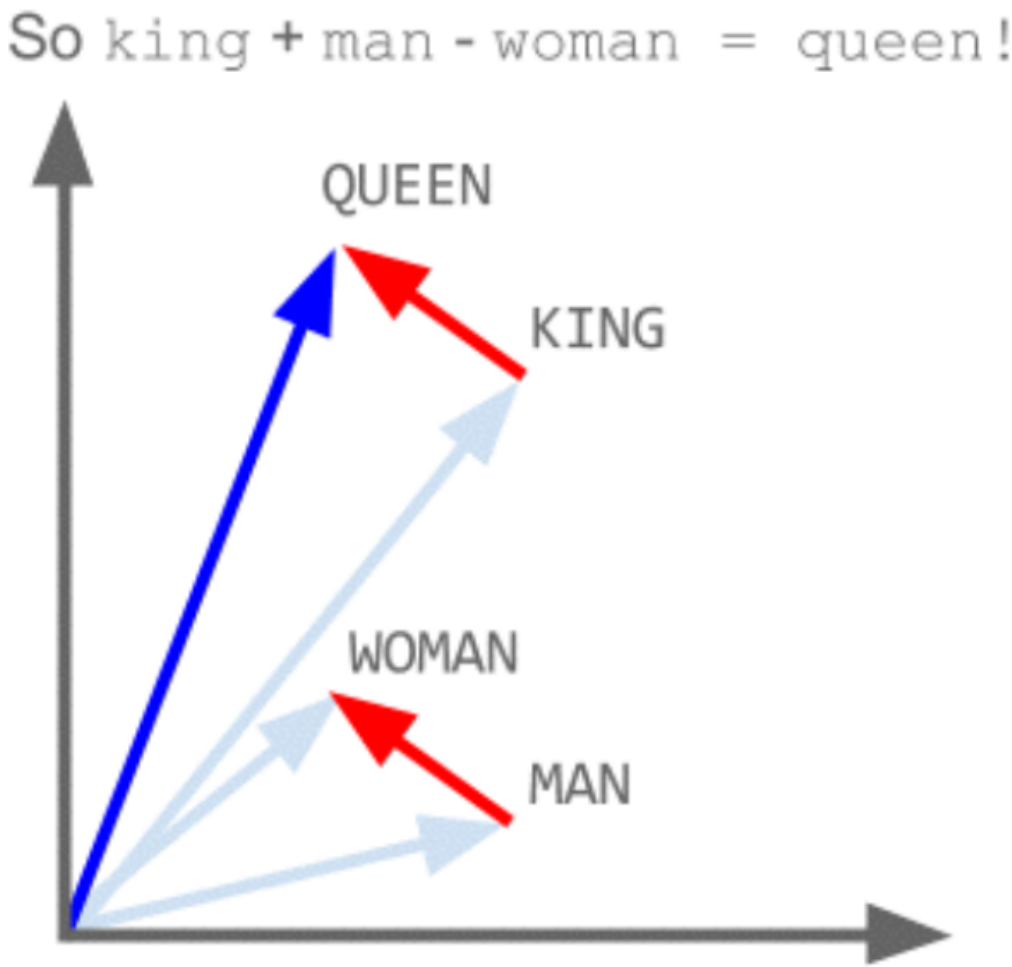


word2vec : nuts and bolts

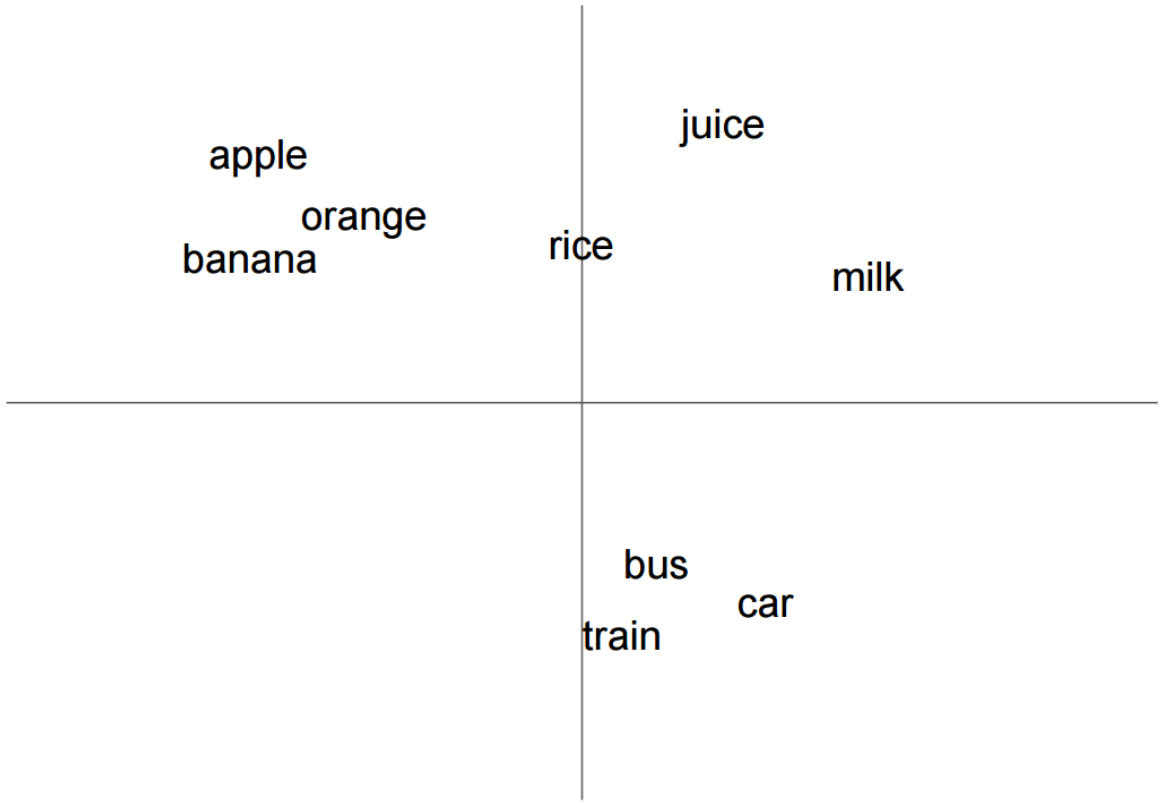
word2vec : nuts and bolts



word2vec : nuts and bolts



word2vec Visualization



word2vec in Nachos

word2vec:

- is a language Modeling ~~Deep Learning~~ Shallow Learning Algorithm
- predicts words locally - given one word it can predict the following word

Motivation
- Atomic Word Representation
- Distributed Representation
- The Data Sparseness Problem

word2vec
- Motivation
- Algorithm
- Analysis
- Summary
- Demo

Latent Dirichlet Allocation
- Motivation

Word Embedding Visualized

bit.ly/wevi-online

LDA - Motivation

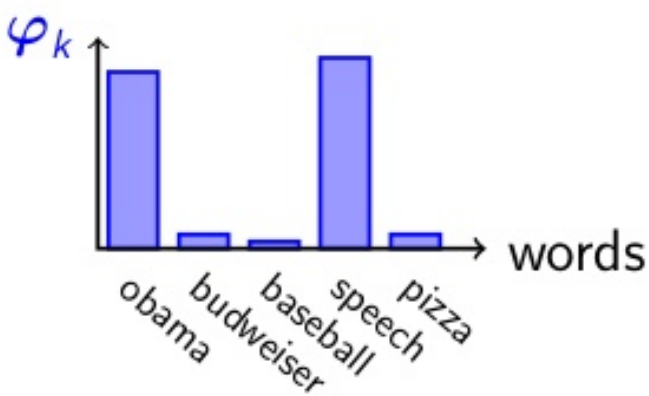
- LDA is able to create document representations a.k.a topics that are not so flexible but human-interpretable.
- LDA treats a set of documents as a set of documents, whereas word2vec works with a set of documents as with one very long text string

Latent Dirichlet Allocation

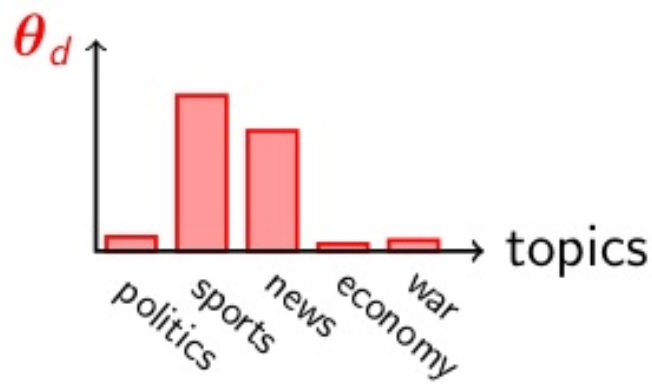
LDA discovers topics into a collection of documents.

LDA tags each document with topics.

Topic k



Document d



In a nutshell

- word2vec models word-to-word relationships
- LDA models document-to-word relationships

Credit

Large swathes of this talk are from previous presentations by

- Chris Moody
- Christopher Colah
- Xin Rong
- Chase Geigle

"I want to understand things clearly, and explain them well."

— Christopher Olah

Thank you!