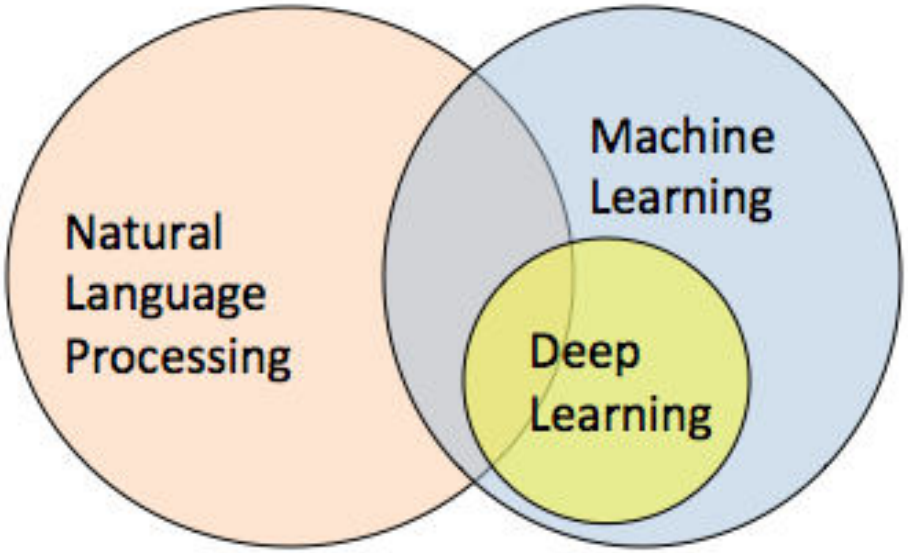
FCIS | Machine Learning 2017

Language Models & Word Vectors

Ahmed Hani | Abdelrahman Hamdy | Ibrahim Sharaf

What is NLP?


Natural
Language
Processing
Machine
Learning
Deep
Learning

How do we represent the meaning of a word?

Definition: **Meaning** (Webster dictionary)

- the idea that is represented by a word, phrase, etc.
- the idea that a person wants to express by using words, signs, etc.
- the idea that is expressed in a work of writing, art, etc.

How to represent meaning in a computer?

Common answer: Use a taxonomy like WordNet that has hypernyms (is-a) relationships and synonym sets (good):

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

```
S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced,
proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
…
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good
```

Problems with this discrete representation

- Great as resource but missing nuances, e.g. **synonyms**:
  adept, expert, good, practiced, proficient, skillful?

- Missing new words (impossible to keep up to date):
  wicked, badass, nifty, crack, ace, wizard, genius, ninjia
- Subjective
- Requires human labor to create and adapt
- Hard to compute accurate word similarity à

Problems with this discrete representation

The vast majority of rule-based **and** statistical NLP work regards words as atomic symbols: hotel, conference, walk

In vector space terms, this is a vector with one 1 and a lot of zeroes

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

We call this a "one-hot" representation. Its problem:

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Distributional similarity based representations

You can get a lot of value by representing a word by means of its neighbors

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

One of the most successful ideas of modern statistical NLP

government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

ë These words will represent *banking ì*

How to make neighbors represent words?

Answer: With a cooccurrence matrix X

- 2 options: full document vs windows
- Word - document cooccurrence matrix will give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis”

- Instead: Window around each word à captures both syntactic (POS) and semantic information

Window based cooccurence matrix

- Window length 1 (more common: 5 - 10)
- Symmetric (irrelevant whether left or right context)
- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

Window based cooccurence matrix

- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

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

Problems with simple cooccurrence vectors

Increase in size with vocabulary

Very high dimensional: require a lot of storage

Subsequent classification models have sparsity issues

à Models are less robust

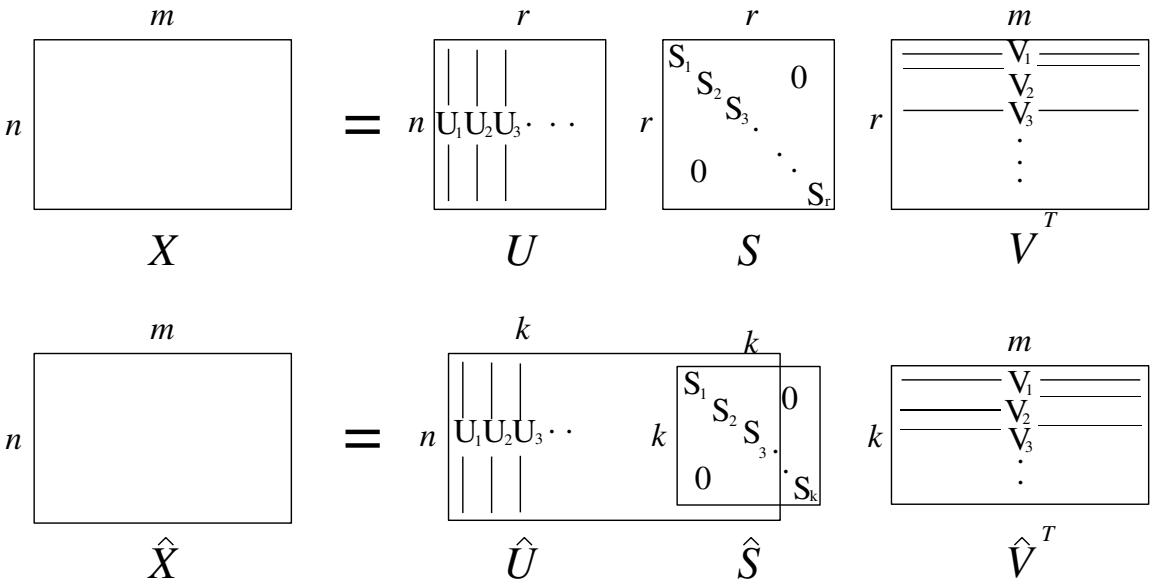
Solution: Low dimensional vectors

- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector
- Usually around 25 – 1000 dimensions

- How to reduce the dimensionality?

Method 1: Dimensionality Reduction on X

Singular Value Decomposition of cooccurrence matrix *X*.



$\hat{X}$ is the best rank *k* approximation to *X* , in terms of least squares.

Simple SVD word vectors in Python

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

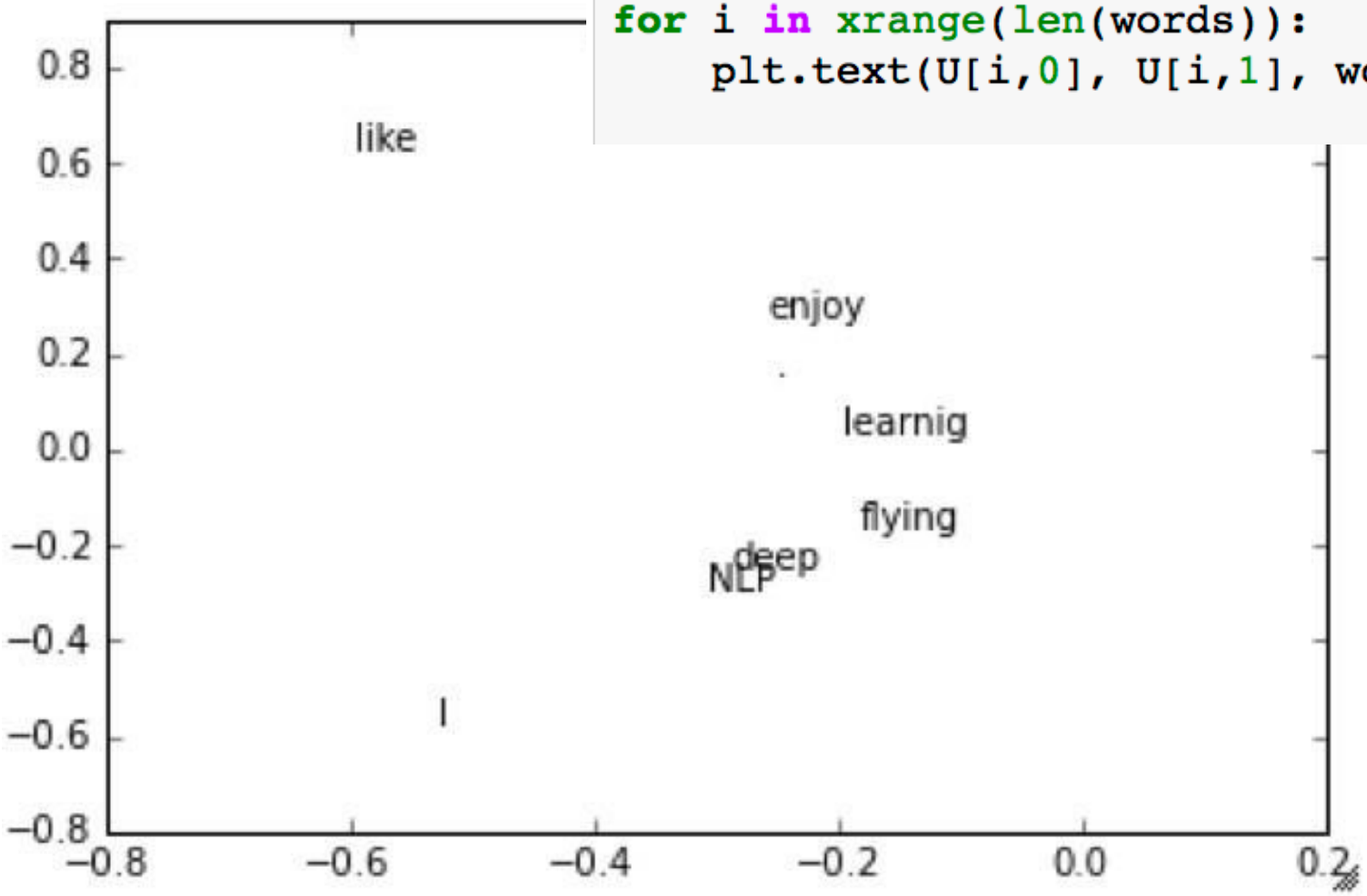
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

Simple SVD word vectors in Python

Corpus: I like deep learning. I like NLP. I enjoy flying.
Printing first two columns of U corresponding to the 2 biggest singular values

```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



Word meaning is defined in terms of vectors

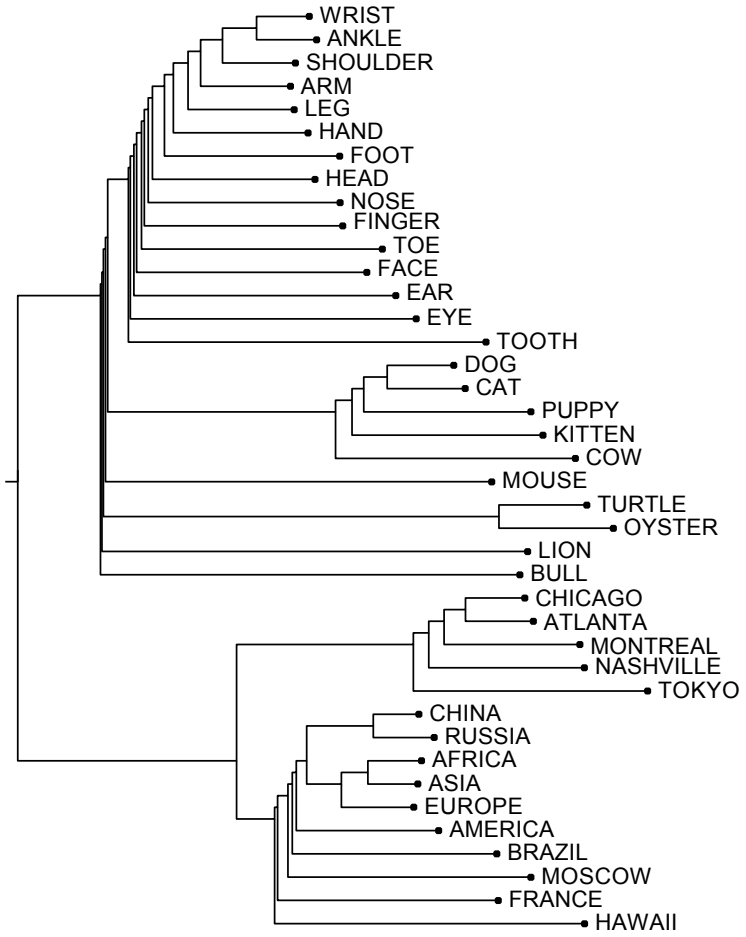
- In all subsequent models, including deep learning models, a word is represented as a dense vector

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Hacks to X

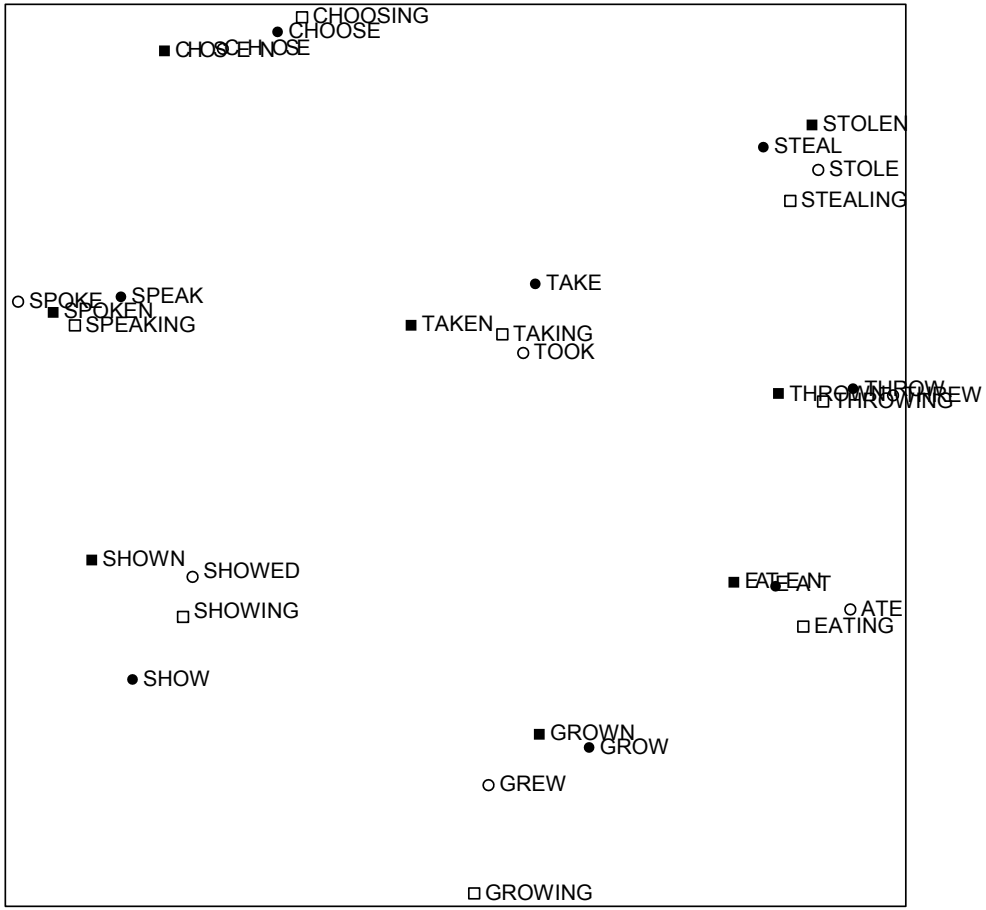
- Problem: function words (the, he, has) are too frequent à syntax has too much impact. Some fixes:
  - min(X,t), with t~100
  - Ignore them all
- Ramped windows that count closer words more
- Use Pearson correlations instead of counts, then set negative values to 0
- +++

Interesting semantic paPers emerge in the vectors



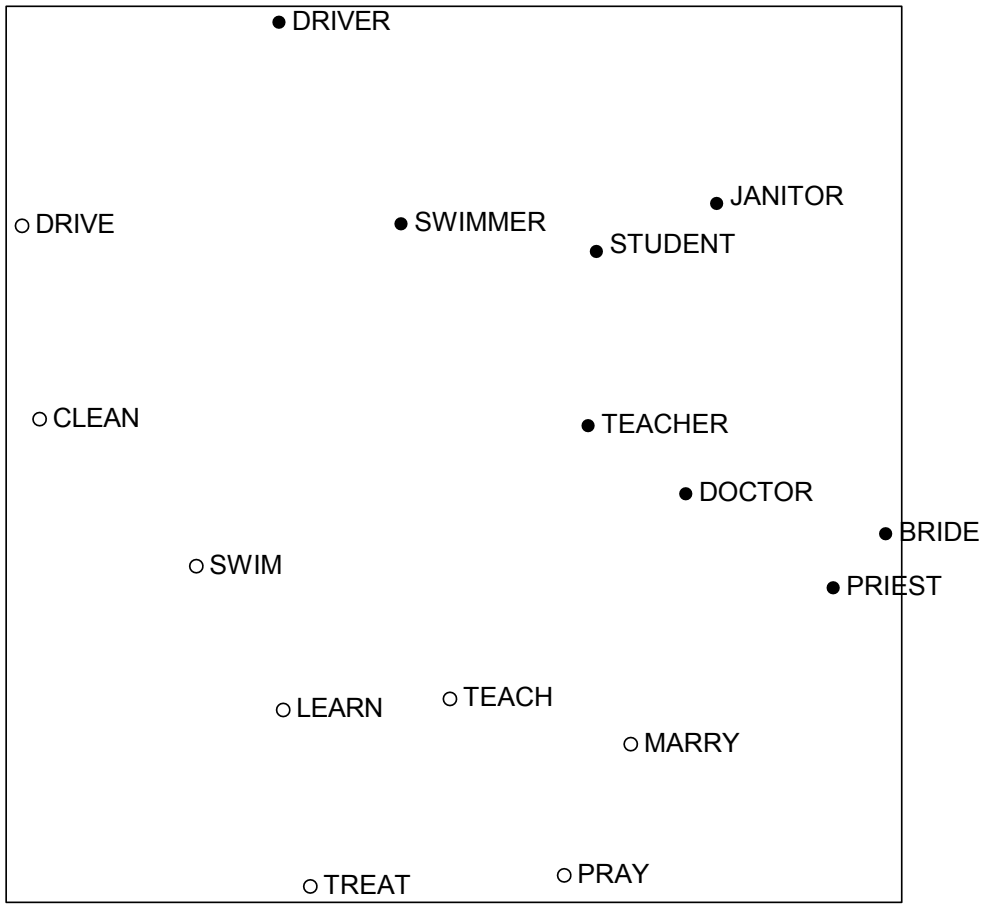
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

Interesting syntactic paPers emerge in the vectors



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

Interesting semantic paPers emerge in the vectors



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

Problems with SVD

Computational cost scales quadratically for n x m matrix:

O(mn^2) flops (when n<m)

à Bad for millions of words or documents

Hard to incorporate new words or documents

Different learning regime than other DL models

Idea: Directly learn low-dimensional word vectors

- Old idea. Relevant for this lecture & deep learning:
  - Learning representations by back-propagating errors. (Rumelhart et al., 1986)
  - A neural probabilistic language model (Bengio et al., 2003)
  - NLP (almost) from Scratch (Collobert & Weston, 2008)
  - A recent, even simpler and faster model: word2vec (Mikolov et al. 2013) à intro now

Main Idea of word2vec

- Instead of capturing cooccurrence counts directly,
- Predict surrounding words of every word
- Both are quite similar, see "*Glove: Global Vectors for Word Representation*" by Pennington et al. (2014) and Levy and Goldberg (2014) … more later

- Faster and can easily incorporate a new sentence/ document or add a word to the vocabulary

Details of Word2Vec

- Predict surrounding words in a window of length m of every word.
- Objective function: Maximize the log probability of any context word given the current center word:
- $$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$
- Where μ represents all variables we optimize

Details of Word2Vec

- Predict surrounding words in a window of length m of every word
- For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

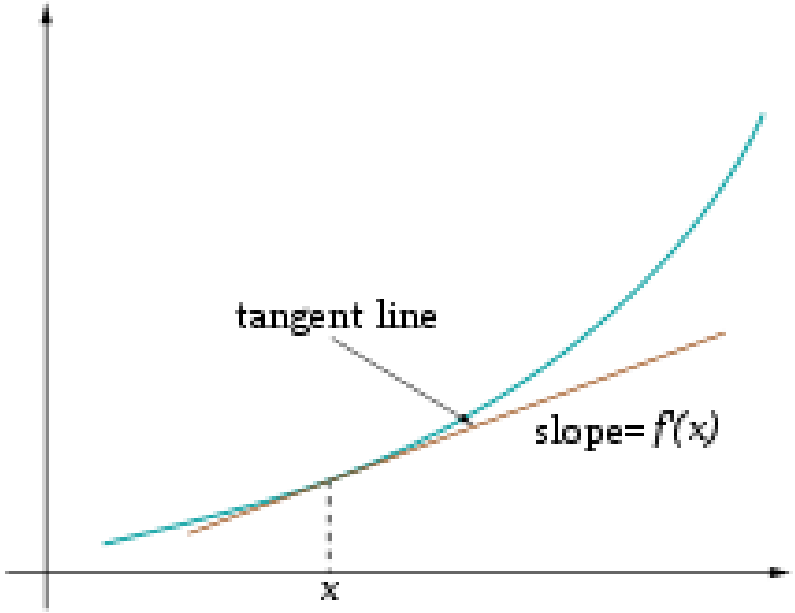
- where o is the outside (or output) word id, c is the center word id, u and v are “center” and “outside” vectors of o and c
- Every word has two vectors!
- This is essentially “dynamic” logistic regression

Cost/Objective functions

We will optimize (maximize or minimize) our objective/cost functions

For now: minimize à gradient descent

Refresher with trivial example: (from Wikipedia)
Find a local minimum of the function
$f(x)=x^4-3x^3+2$, with derivative $f'(x)=4x^3-9x^2$.




```
x_old = 0
x_new = 6 # The algorithm starts at x=6
eps = 0.01 # step size
precision = 0.00001

def f_derivative(x):
    return 4 * x**3 - 9 * x**2

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new = x_old - eps * f_derivative(x_old)

print("Local minimum occurs at", x_new)
```

Two architectures

Skip--gram CBOW

Neither has a hidden layer, only a projection layer:

- Predict a missing word given a window of context words (Skip--gram)
- Predict context words given a word (CBOW: Continuous Bag Of Words)

Two architectures

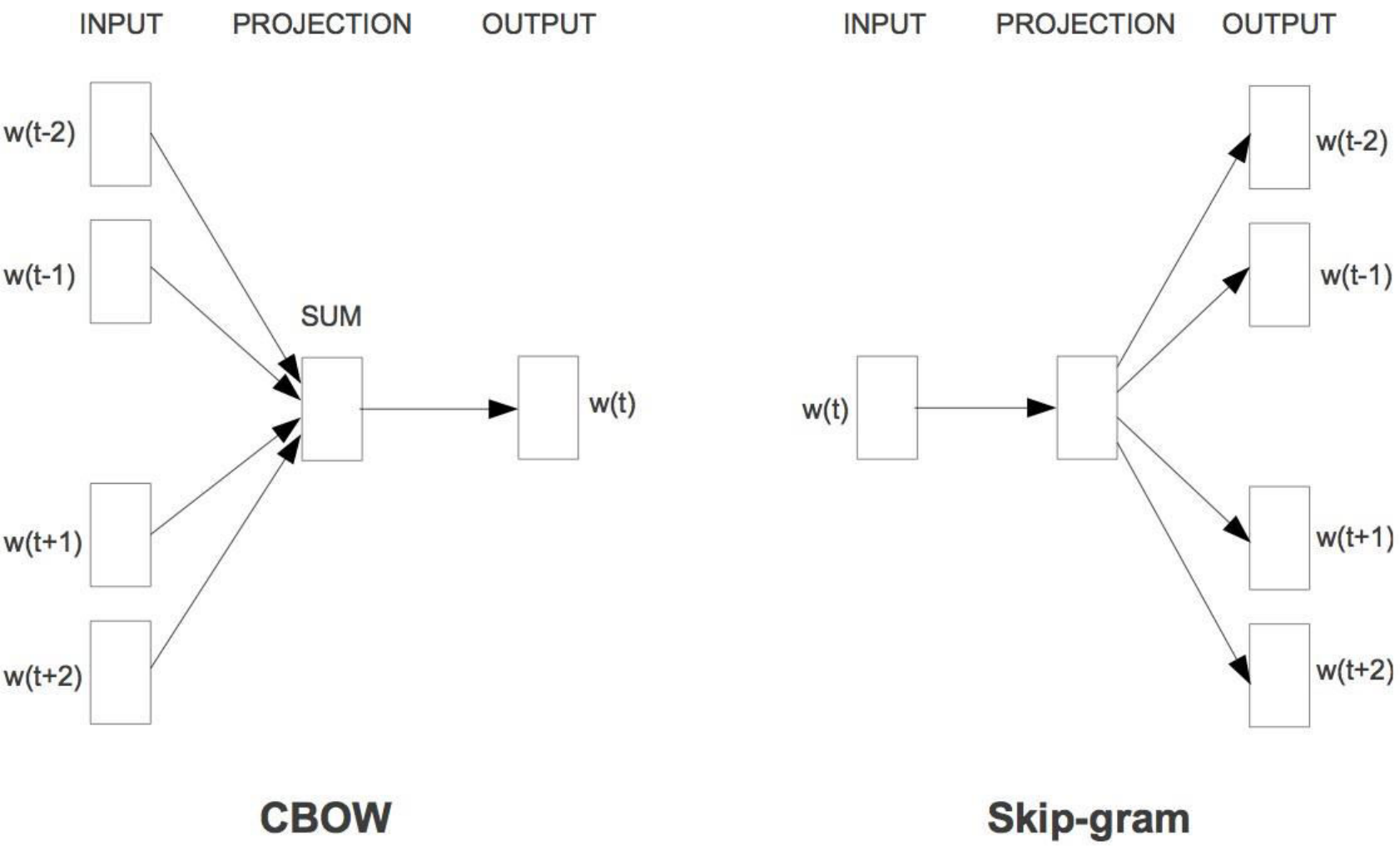


Figure from [4]

Linear Relationships in word2vec

These representations are *very good* at encoding dimensions of similarity!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space

  Syntactically

  - $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$

  - Similarly for verb and adjective morphological forms

  Semantically (Semeval 2012 task 2)

  - $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$
  - $x_{king} - x_{man} \approx x_{queen} - x_{woman}$

Word Analogies

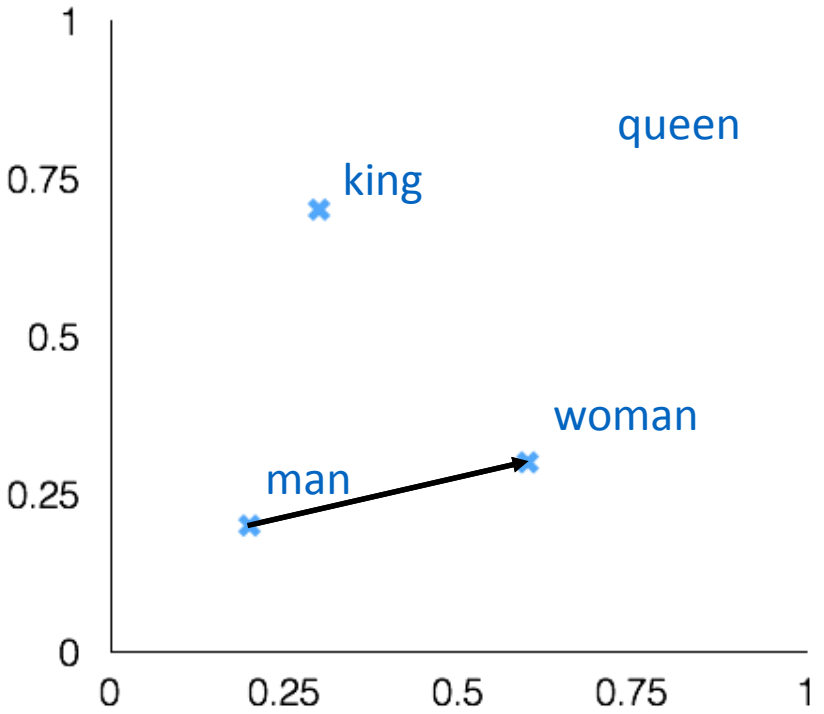
Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



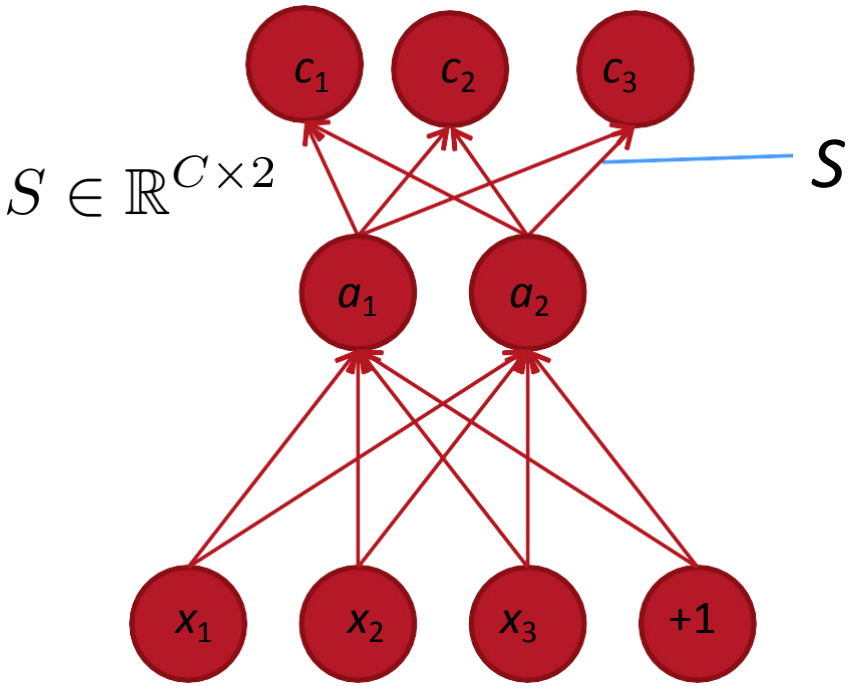
Advantages of low dimensional word vectors

What is the major benefit of deep learned word vectors?

Ability to also propagate **any** information into them via neural networks.

$$P(c \mid d, \lambda) = \frac{e^{\lambda^{\mathsf{T}} f(c,d)}}{\sum_{c!} e^{\lambda^{\mathsf{T}} f(c!,d)}}$$

$$p(c|x) = \frac{\exp(S_c.a)}{\sum_{c'} \exp(S_{c'}.a)}$$



What can you do with it?

- Sentence completion: Microsoft Sentence Completion Challenge
- Selecting out of list words

  Example: which word does not belong in [*monkey, lion, dog, truck]*
- Bilingual Word Embeddings for Phrase--Based Machine Translation. *EMNLP*. 2013.
- Synonym detection

  Choose among 4 candidates which one is the best synonym for a given word.

What can you do with it?

- Concept categorization

  Group *helicopter* and *motorcycle* together, and *dog* an *elephant*
- Selectional preference

  Predict typical verb noun pairs. Like *people* go well with to *eat* as a subject (but not as an object).

Dealing with phrases

- Deal with this during pre--processing step

  Find collocations the usual way, and feed them as single ‘words’ to the NN

- Just add/average/concatenate the vectors

  This is better than the previous approach if you happen to be dealing with entities like *Bengal tiger* versus *Sumatran Tiger*

- Paragraph2vec

  Make fixed--length representation of *variable*--length input

- Build a convolutional NN on top of word2vec

  This is done in [2] for sentence classification

(By no means extensive) list of papers

Overview and evaluation of many different tasks

[1] M. Baroni, G. Dinu, and G. Kruszewski. Dont count, predict! a systematic comparison of context–counting vs. context--predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, volume 1, 2014.

Sentence classification

[2] Y. Kim. Convolutional neural networks for sentence classification.

This is where the 'gay' and 'cell' example came from

[3] Y. Kim, Y.–I.Chiu, K. Hanaki, D. Hedge, and S. Petrov. Temporal analysis of language through neural language models, 2014.

For more elaborate discussion on the architectures used

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

More on Skip–gram, negative sampling, hierarchical soMmax, and dealing with phrases

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119, 2013.

Resources

- Stanford NLP (Lectures)
- Stanford CS224n: Natural Language Processing with Deep Learning (Class)
- Word2Vec curated resources (Link)
- The amazing power of word vectors (Link)
- Understanding Word2Vec for word embeddings (Link)