Language Representation

Lecturer: Eilam Shapira

NLP Course - Unit 3 - Lecture

Based on Amir Feder's slides & Stanford's CS224n

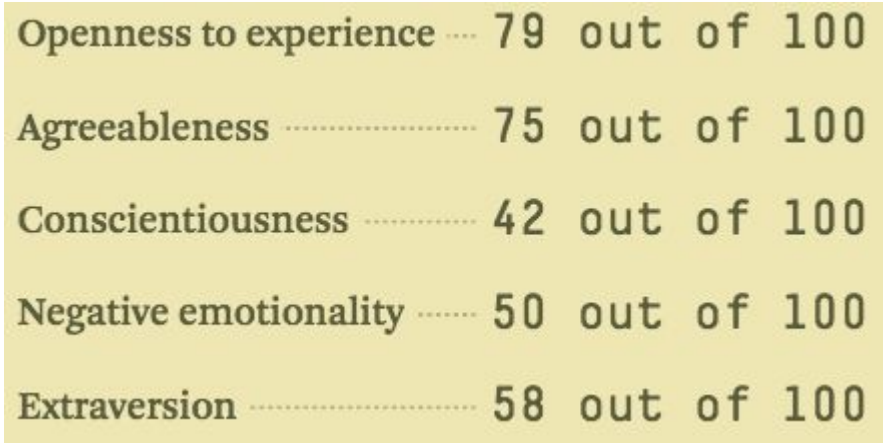
Today’s Agenda

- **Word2Vec**
- GloVe
- Sentence representations
- Applications

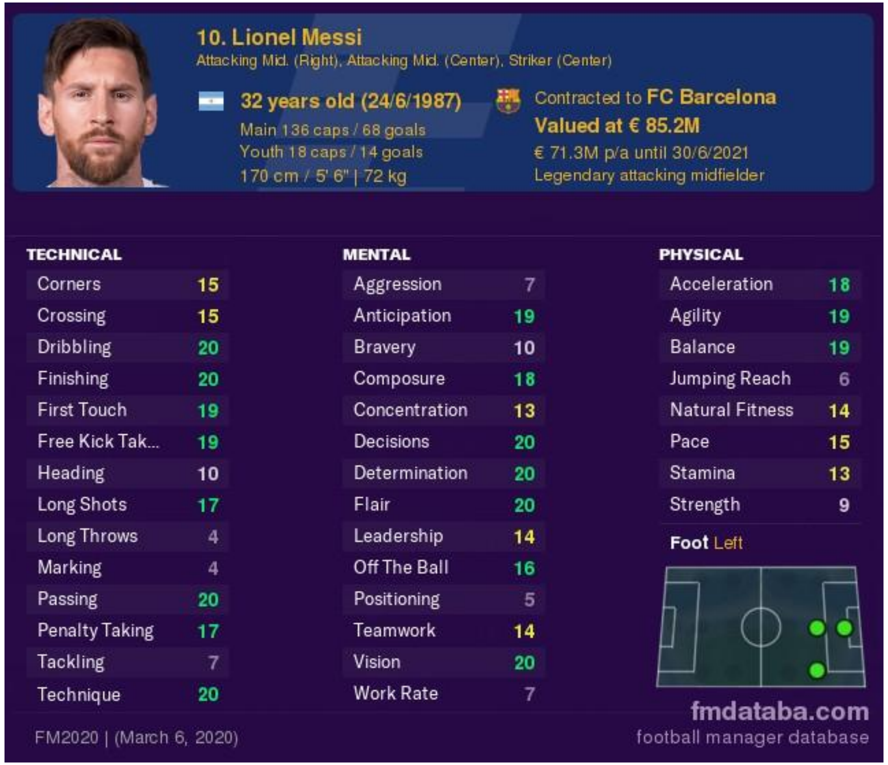
Before we start...

- Can you engineer features to represent:
  - Words?
  - Sentences?
  - Documents?
  - What about yourselves?
- What’s the dimension?
  - Remember PCA?

Embedding is a powerful way to represent data



Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100



10. Lionel Messi
Attacking Mid. (Right), Attacking Mid. (Center), Striker (Center)

32 years old (24/6/1987)
Main 136 caps / 68 goals
Youth 18 caps / 14 goals
170 cm / 5' 6" | 72 kg

Contracted to FC Barcelona
Valued at € 85.2M
€ 71.3M p/a until 30/6/2021
Legendary attacking midfielder

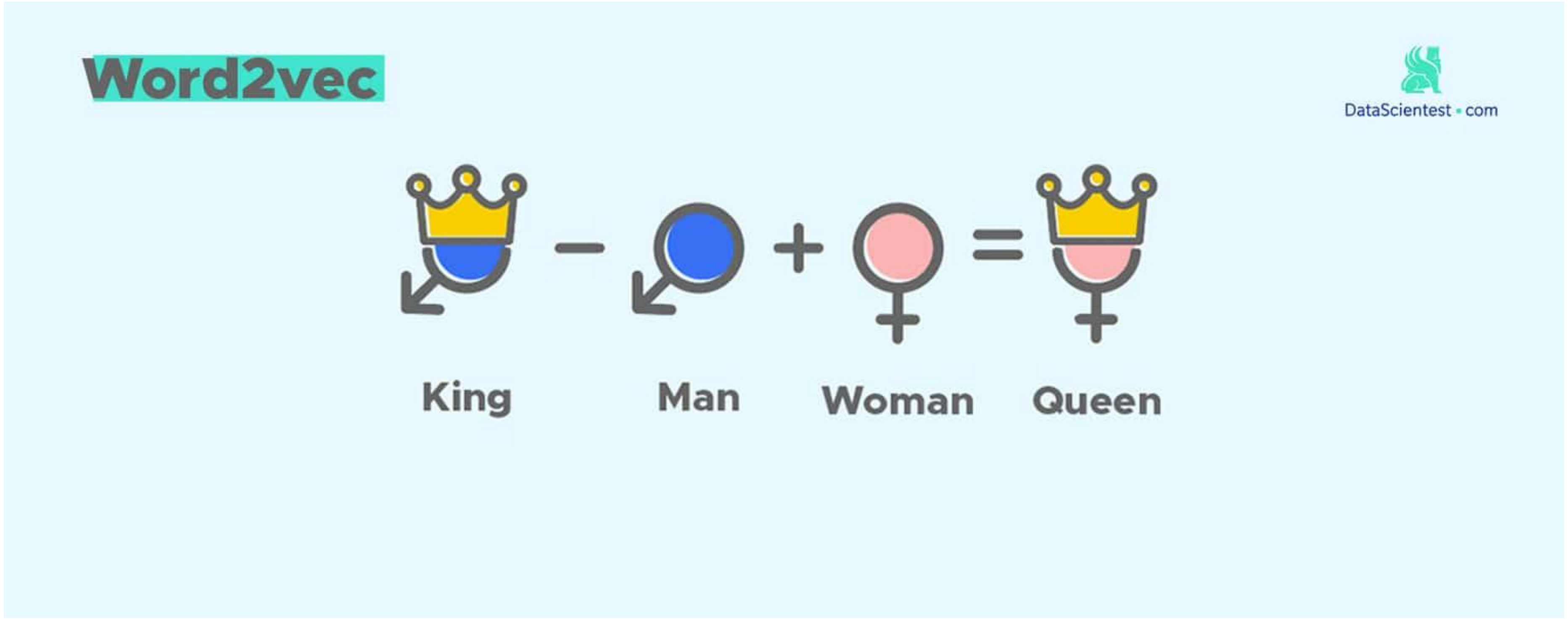
TECHNICAL		MENTAL		PHYSICAL	
Corners	15	Aggression	7	Acceleration	18
Crossing	15	Anticipation	19	Agility	19
Dribbling	20	Bravery	10	Balance	19
Finishing	20	Composure	18	Jumping Reach	6
First Touch	19	Concentration	13	Natural Fitness	14
Free Kick Tak...	19	Decisions	20	Pace	15
Heading	10	Determination	20	Stamina	13
Long Shots	17	Flair	20	Strength	9
Long Throws	4	Leadership	14		
Marking	4	Off The Ball	16		
Passing	20	Positioning	5		
Penalty Taking	17	Teamwork	14		
Tackling	7	Vision	20		
Technique	20	Work Rate	7		

Foot Left

FM2020 | (March 6, 2020)

fmdataba.com
football manager database

Similarity-preserving Representation



Word2Vec

Word2vec: Skip-Gram Task

- Word2vec provides a variety of options. Let's do
    - "skip-gram with negative sampling" (SGNS)
    - ~~Continuous Bag-of-words (CBOW)~~

- Word2vec actually uses a logistic regression initialized with random weights
    - But we’ll show it with a shallow NN

Skip-Gram Algorithm

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples
3. Use NN to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

Skip-Gram Goal

Given a tuple (t,c) = target, context

- (*hotel*, *haifa*)
- (*hotel*, *aardvark*)

Return probability that c is a real context word:

$P(+|t,c)$

$P(-|t,c) = 1 - P(+|t,c)$

How to compute p(+|t,c)?

Intuition:

- Words are likely to appear near similar words
- Model similarity with dot-product!
- Similarity(t,c) $\propto t \cdot c$

Problem:

- *Dot product is not a probability!*
  - *(Neither is cosine)*

Turning dot product into a probability

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$$P(-|t,c) = 1 - P(+|t,c)$$

$$= \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$

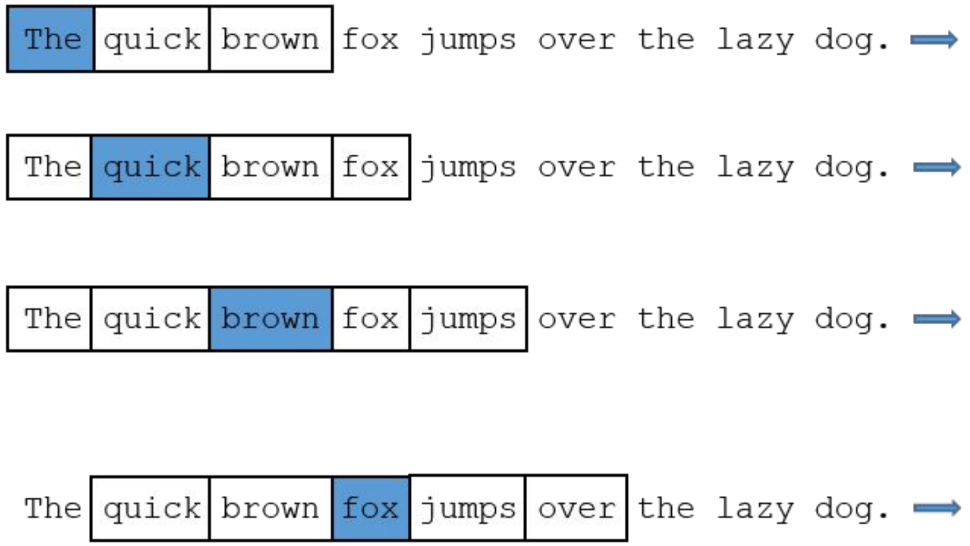
For all the context words

Assume they are independent

$$P(+|t, c_{1:k}) = \Pi_{i=1}^{K} \frac{1}{1+e^{-t \cdot c_i}}$$

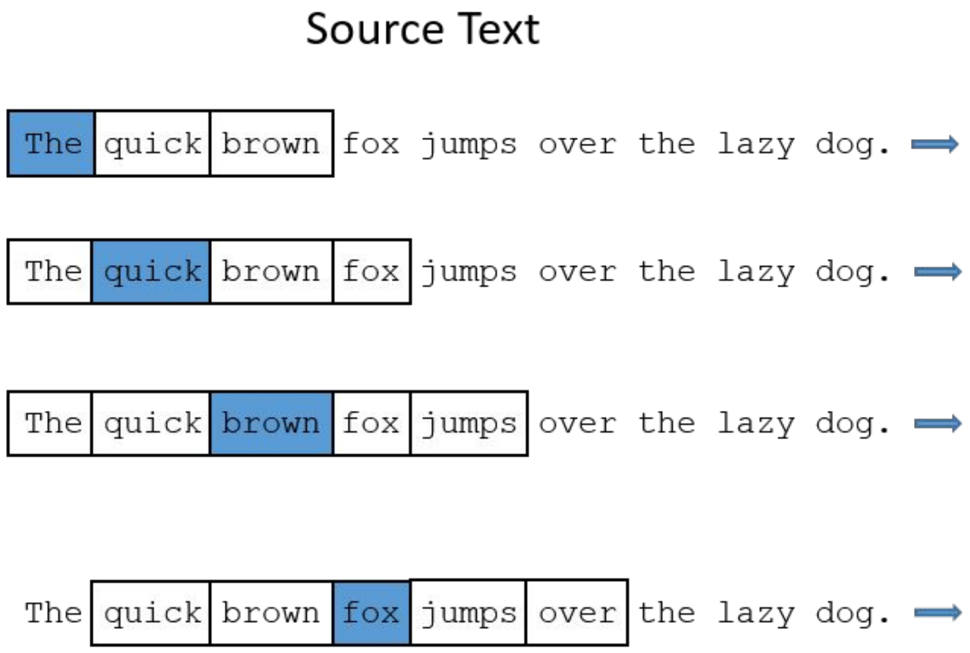
$$logP(+|t, c_{1:k}) = \sum_{i=1}^{K} log \frac{1}{1+e^{-t \cdot c_i}}$$

Positive Training Examples


The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Source Text	Training Samples
The quick brown fox jumps over the lazy dog.	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog.	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog.	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog.	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Negative Training Examples



Source Text
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Training Samples	Negative Samples
(the, quick) (the, brown)	(the, aardvark) … (the,zztop)
(quick, the) (quick, brown) (quick, fox)	
(brown, the) (brown, quick) (brown, fox) (brown, jumps)	(brown, aardvark) … (brown,zztop)
(fox, quick) (fox, brown) (fox, jumps) (fox, over)	….

Objective Criteria

We want to maximize…

$$\sum_{(t,c)\in +} logP(+|t,c) + \sum_{(t,c)\in -} logP(-|t,c)$$

Maximize the + label for the pairs from the positive training data, and the – label for the pairs sample from the negative data.

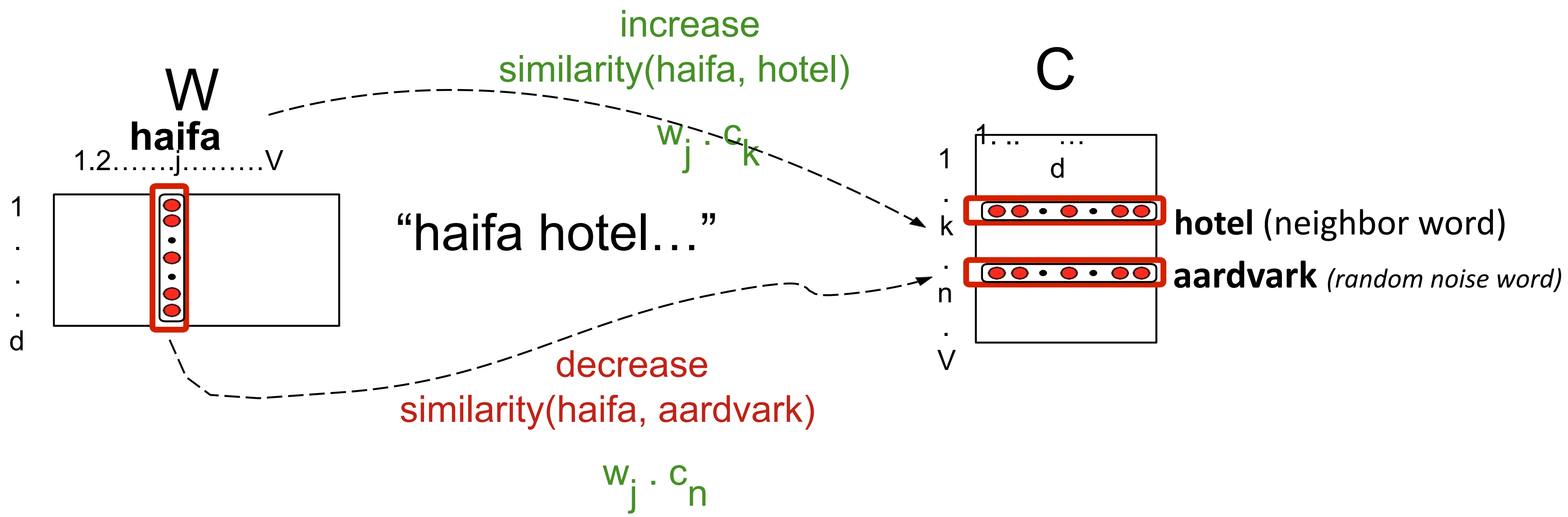
Focusing on one target word t

$$L(\theta) = logP(+|t,c) + \sum_{i=1}^{k} logP(-|t,n_i)$$

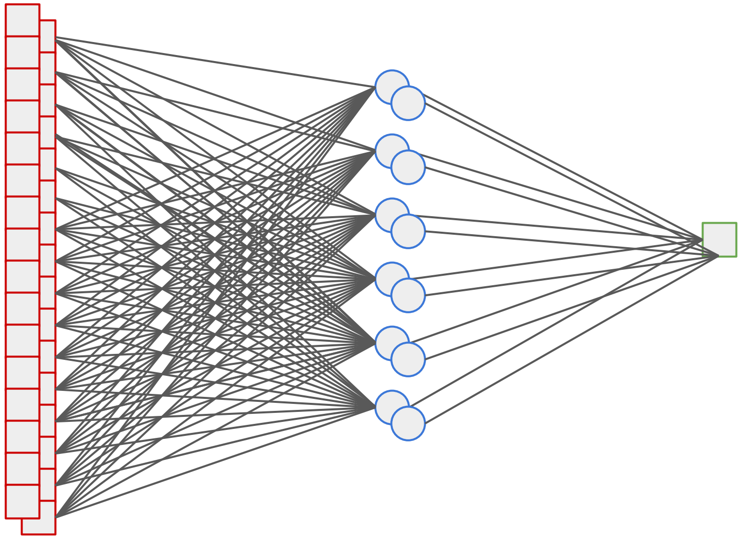
$$= log\sigma(c \cdot t) + \sum_{i=1}^{k} log\sigma(-n_i \cdot t)$$

$$= log\frac{1}{1+e^{-c \cdot t}} + \sum_{i=1}^{k} log\frac{1}{1+e^{n_i \cdot t}}$$

Training step (via GD)



Word2vec – Forward Step

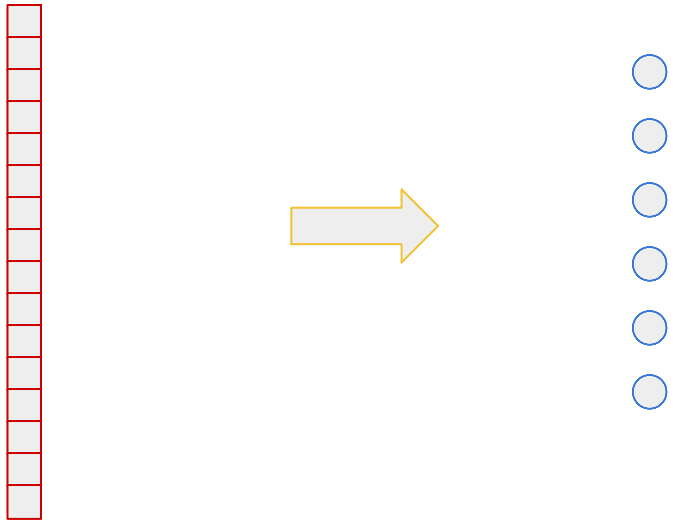


Vocabulary size
~450,000

Word vector
dimension
300

Binary
Prediction
+/-

Word2Vec – Result



Vocabulary size
~450,000

Word vector
dimension
300

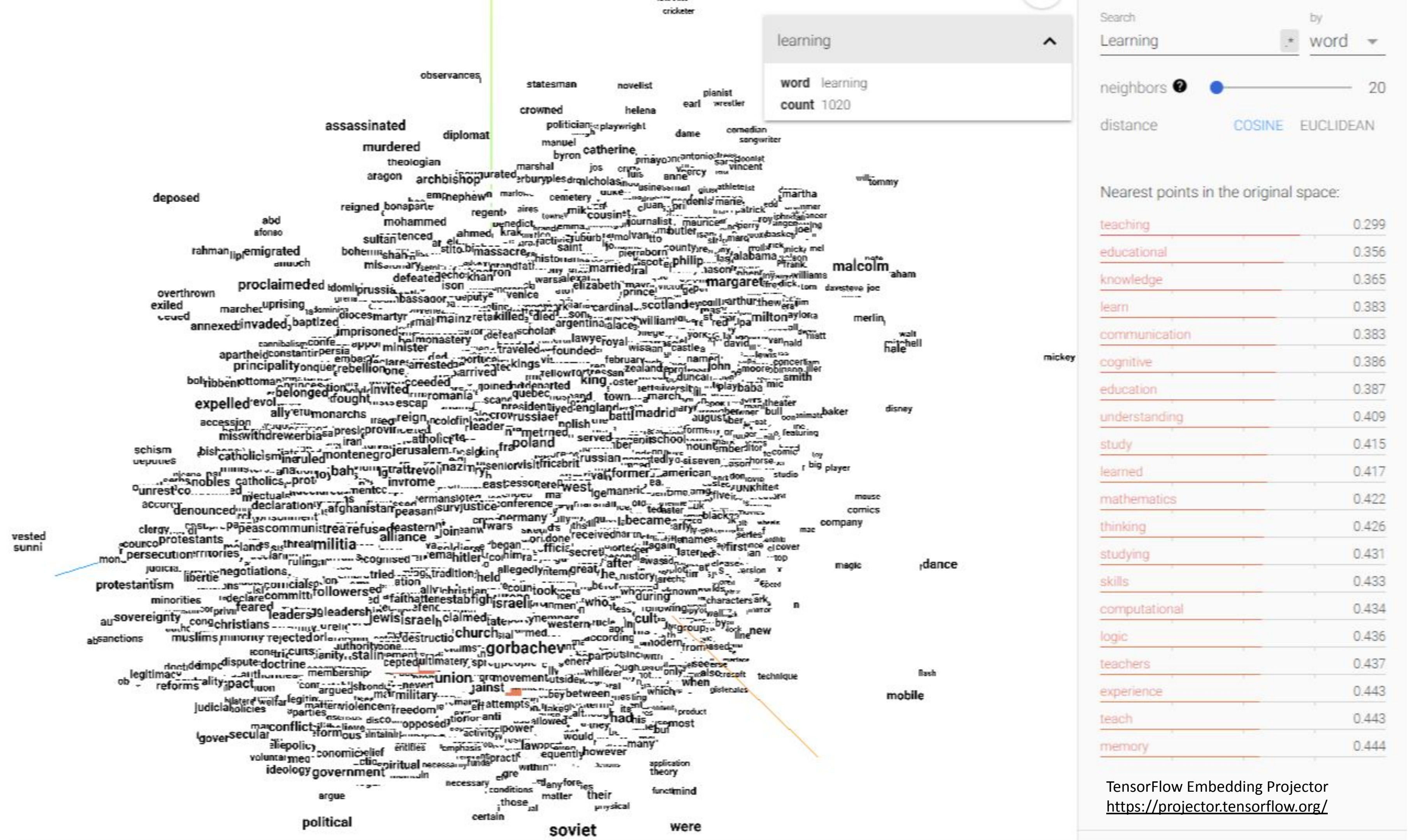
Summary: How to learn word2vec (skip-gram) embeddings

- Start with 2 random 300-dimensional layers (1 NN)
- In a corpus, take pairs of words that co-occur as positive examples
- Take pairs of words that don't as negative examples
- Train a classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

Evaluating embeddings

Compare to human scores on word similarity-type tasks:

- WordSim-353 (Finkelstein et al., 2002)
- **SimLex-999 (Hill et al., 2015)**
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*



learning

word learning

count 1020

Search Learning .* by word

neighbors 20

distance COSINE EUCLIDEAN

Nearest points in the original space:

teaching	0.299
educational	0.356
knowledge	0.365
learn	0.383
communication	0.383
cognitive	0.386
education	0.387
understanding	0.409
study	0.415
learned	0.417
mathematics	0.422
thinking	0.426
studying	0.431
skills	0.433
computational	0.434
logic	0.436
teachers	0.437
experience	0.443
teach	0.443
memory	0.444

TensorFlow Embedding Projector
https://projector.tensorflow.org/

Word Vector Space Models - Discussion

- The model is derived from context, but not context sensitive:
  One representation (one vector) for the words: ***play*** and ***bank***.
  Yet they have many different senses, which are context dependent.
  bank: border of the lake, financial institute, to pile up…
  play: to play a piano, to play with a ball, a play…

- Vectors represent words, but how can we represent phrases and sentences?
- What about combining different modalities such as visual and textual ?

tiger

rose

zebra

Today’s Agenda

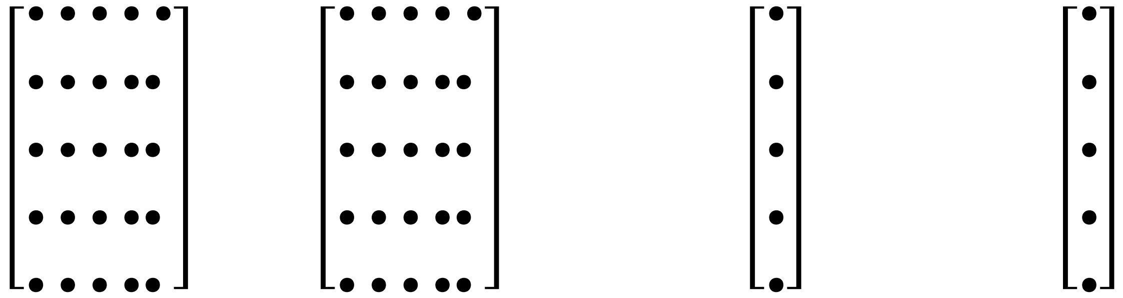
- Word2Vec
- **GloVe**
- Sentence representations
- Applications

GloVe

GloVe - TL;DR

- GloVe is a static word embedding algorithm
- Combines ideas from co-occurrence and language modeling
- Essentially closed the static word embedding lit.
    - Whenever we say we use *static word embedding*, we use GloVe

Word2vec parameters and computations



U outside

V center

$U.\,v_4{}^T$ dot product

$\mathrm{softmax}(U.\,v_4{}^T)$ probabilities

"Bag of words" model! $\rightarrow$ The model makes the same predictions at each position

We want a model that gives a reasonably high probability estimate to *all* words that occur in the context (at all often)

Gradient Descent

- Update equation (in matrix notation):

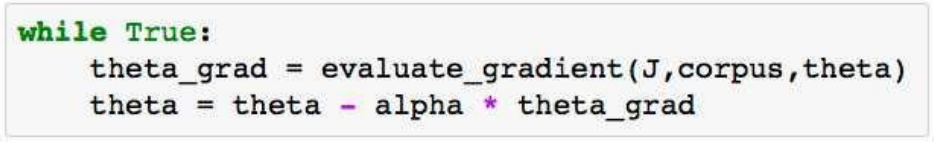
$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

α = *step size* or *learning rate*

- Update equation (for a single parameter):

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

- Algorithm:

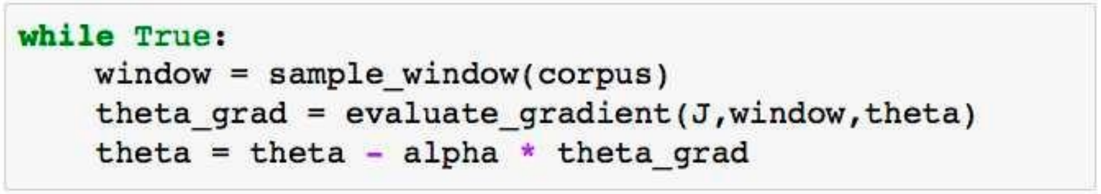


```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

Stochastic Gradient Descent

- **Problem**: $J(\theta)$ is a function of all windows in the corpus (often, billions!)
  - So $\nabla_\theta J(\theta)$ is very expensive to compute
- You would wait a very long time before making a single update!

- **Very** bad idea for pretty much all neural nets!
- **Solution: Stochastic gradient descent (SGD)**
  - Repeatedly sample windows, and update after each one, or each small batch
- Algorithm:



```
while True:
    window = sample_window(corpus)
    theta_grad = evaluate_gradient(J,window,theta)
    theta = theta - alpha * theta_grad
```

Stochastic gradients with word vectors! [Aside]

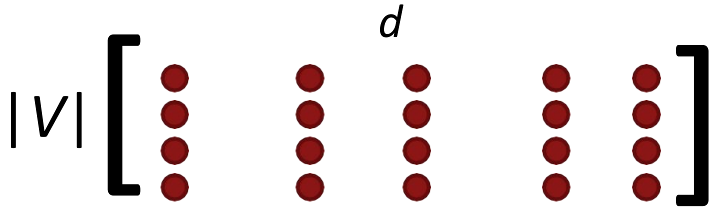
- Iteratively take gradients at each such window for SGD
- But in each window, we only have at most 2*m* + 1 words, so $\nabla_\theta J_t(\theta)$ is very sparse!

$$\nabla_\theta J_t(\theta) = \begin{bmatrix} 0 \\ \vdots \\ \nabla_{v_{like}} \\ \vdots \\ 0 \\ \nabla_{u_I} \\ \vdots \\ \nabla_{u_{learning}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2dV}$$

Stochastic gradients with word vectors!

- We might only update the word vectors that actually appear!

- Solution: either you need sparse matrix update operations to only update certain rows of full embedding matrices *U* and *V*, or you need to keep around a hash for word vectors



- If you have millions of word vectors and do distributed computing, it is important to not have to send gigantic updates around!

Why not capture co-occurrence counts directly?

Building a co-occurrence matrix *X*

- 2 options: windows vs. full document
- Window: Similar to word2vec, use window around each word
    - captures some syntactic and semantic information
- Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis”

Example: Window based co-occurrence 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

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

Co-occurrence vectors

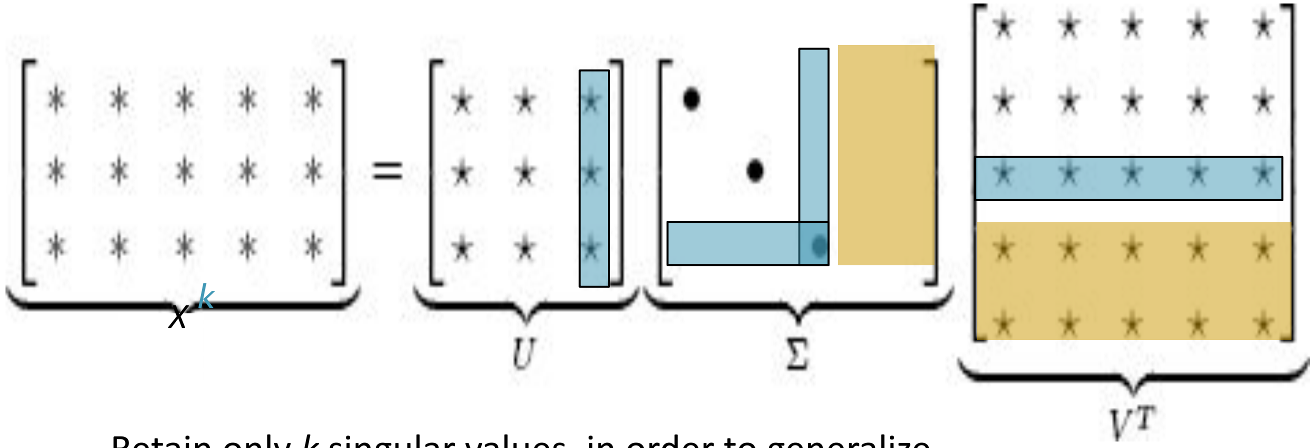
- Simple count co-occurrence vectors
  - Vectors increase in size with vocabulary
  - Very high dimensional: require a lot of storage (though sparse)
  - Subsequent classification models have sparsity issues -> Models are less robust

- Low-dimensional vectors
  - Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector
  - Usually 25–1000 dimensions, similar to word2vec
  - How to reduce the dimensionality?

Classic Method: Dimensionality Reduction on X

Singular Value Decomposition of co-occurrence matrix X

Factorizes X into UΣVT, where U and V are orthonormal



Retain only *k* singular values, in order to generalize.

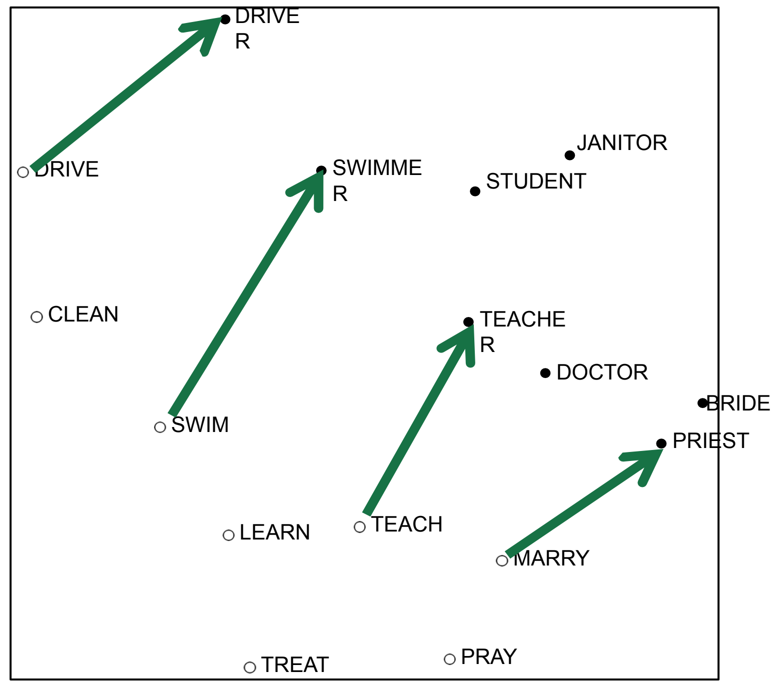
X is the best rank *k* approximation to *X* , in terms of least squares.
Classic linear algebra result. Expensive to compute for large matrices.

Hacks to X (several used in Rohde et al. 2005 in COALS)

- Running an SVD on raw counts doesn’t work well

- Scaling the counts in the cells can help ***a lot***
  - Problem: function words (*the, he, has*) are too frequent syntax has too much impact. Some fixes:
    - log the frequencies
    - min(X,t), with t ≈ 100
    - Ignore the function words
- Ramped windows that count closer words more than further away words
- Use Pearson correlations instead of counts, then set negative values to 0
- Etc.

Interesting semantic patterns emerge in the scaled vectors



COALS model from
Rohde et al. ms., 2005. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

Towards GloVe: Count based vs. direct prediction

- LSA, HAL
- COALS, Hellinger-PCA

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- Skip-gram/CBOW
- NNLM, HLBL, RNN

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Encoding meaning components in vector differences

Crucial insight: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = random
$P(x \mid \text{ice})$	**large**	small	**large**	small
$P(x \mid \text{steam})$	small	**large**	**large**	small
$\frac{P(x \mid \text{ice})}{P(x \mid \text{steam})}$	**large**	small	~1	~1

Encoding meaning in vector differences

Crucial insight: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	**1.9 x 10^{-4}**	6.6 x 10^{-5}	**3.0 x 10^{-3}**	1.7 x 10^{-5}
$P(x\|\text{steam})$	2.2 x 10^{-5}	**7.8 x 10^{-4}**	**2.2 x 10^{-3}**	1.8 x 10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	**8.9**	8.5 x 10^{-2}	1.36	0.96

Encoding meaning in vector differences

Q: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

A: Log-bilinear model: with vector differences

$$w_i \cdot w_j = \log P(i|j)$$

$$w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$$

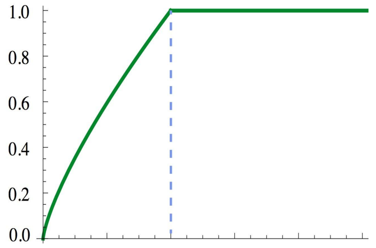
Combining the best of both worlds - GloVe

$$w_i \cdot w_j = \log P(i|j)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors

$f \sim$



GloVe results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

How to evaluate word vectors?

- Related to general evaluation in NLP: Intrinsic vs. extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy

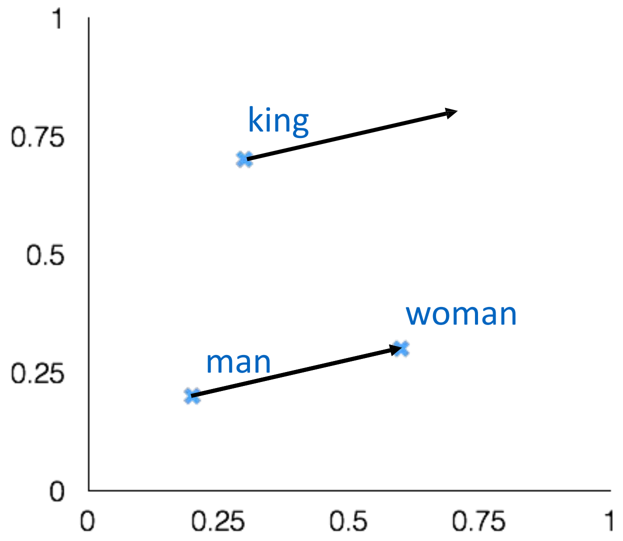
Intrinsic word vector evaluation

- Word Vector Analogies

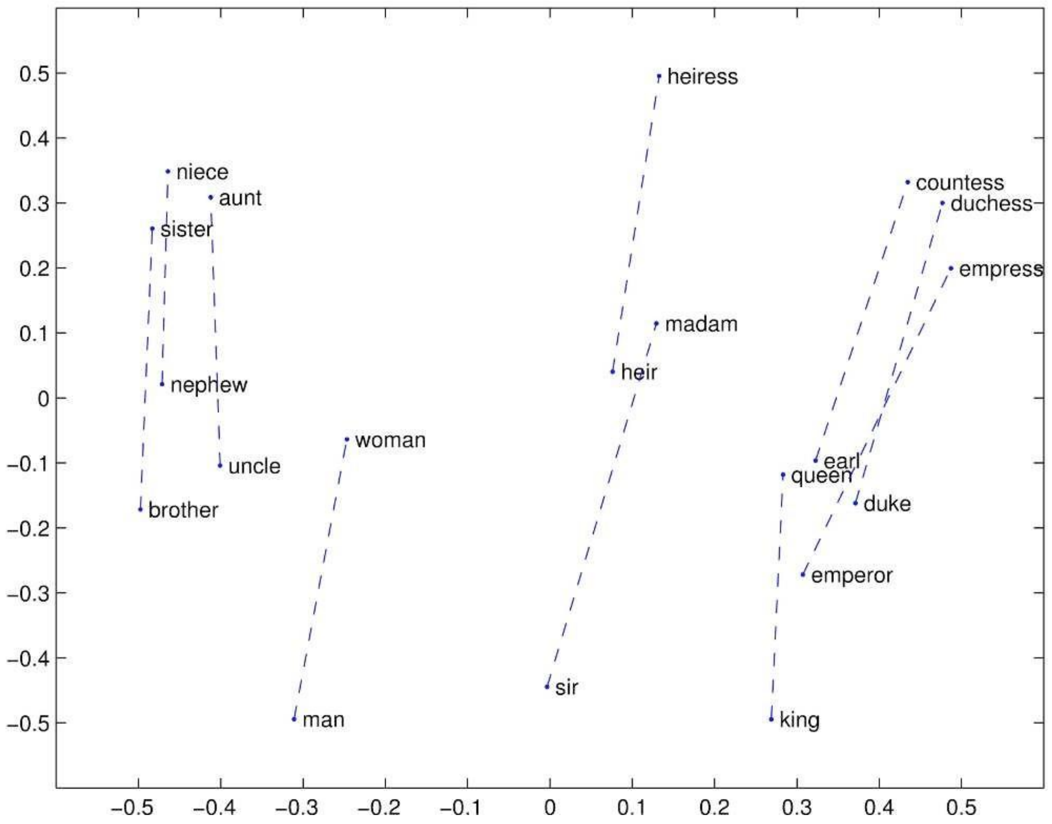
$$\boxed{\text{a:b :: c:?}} \longrightarrow \boxed{d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}}$$

man:woman :: king:?

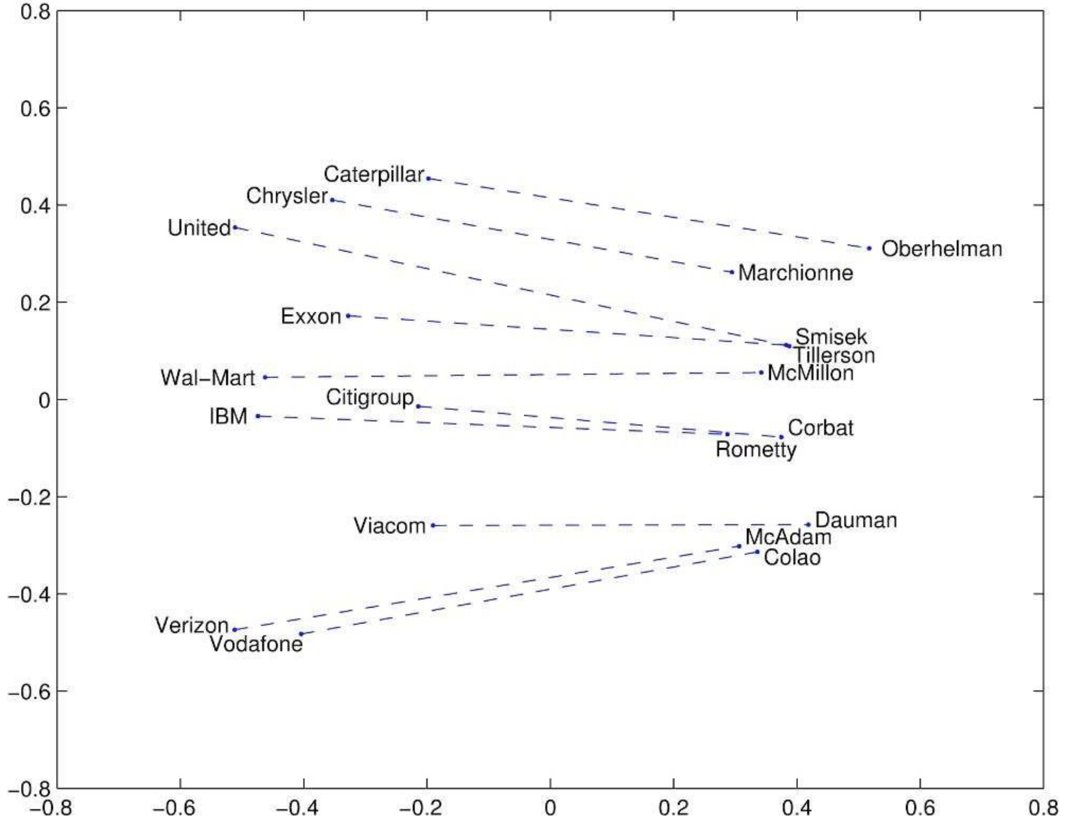
- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



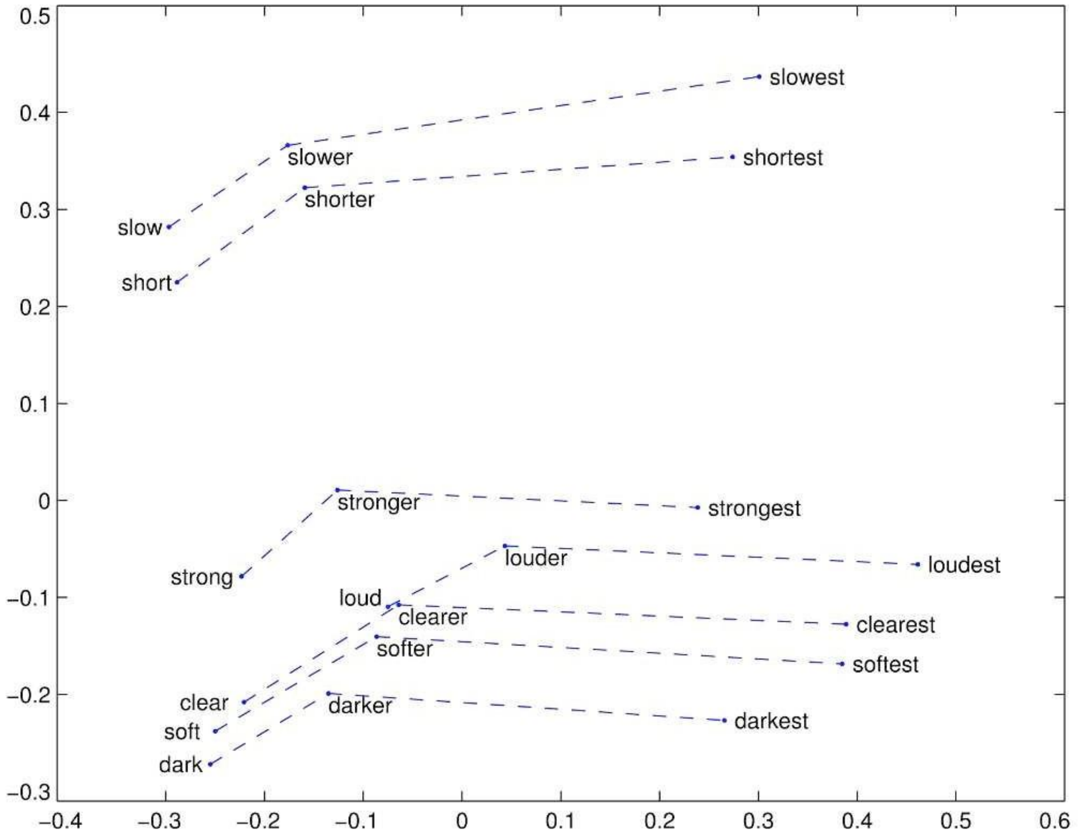
Glove Visualizations



Glove Visualizations: Company - CEO

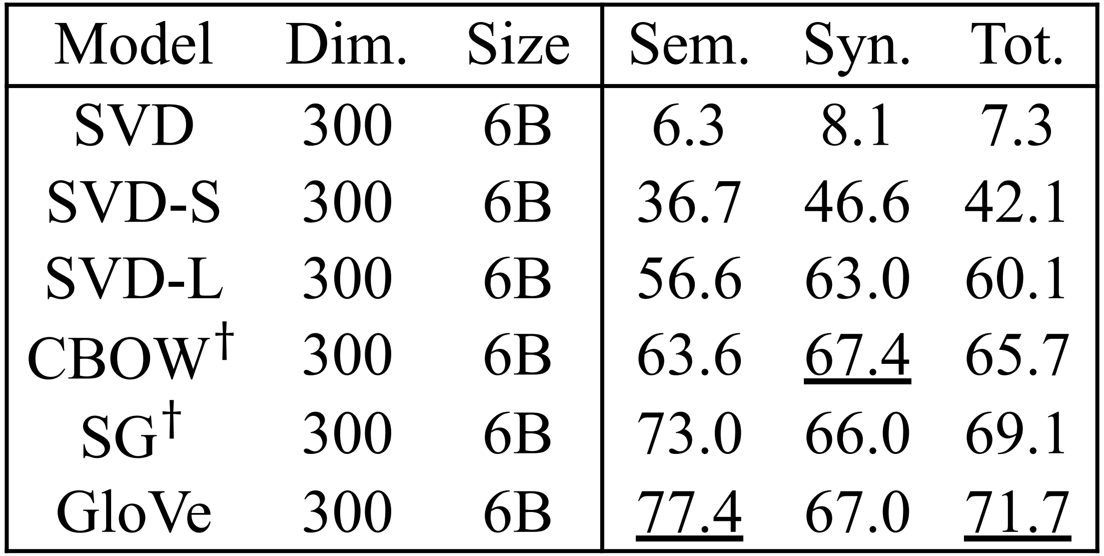


Glove Visualizations: Comparatives and Superlatives



Analogy evaluation and hyperparameters

Glove word vectors evaluation

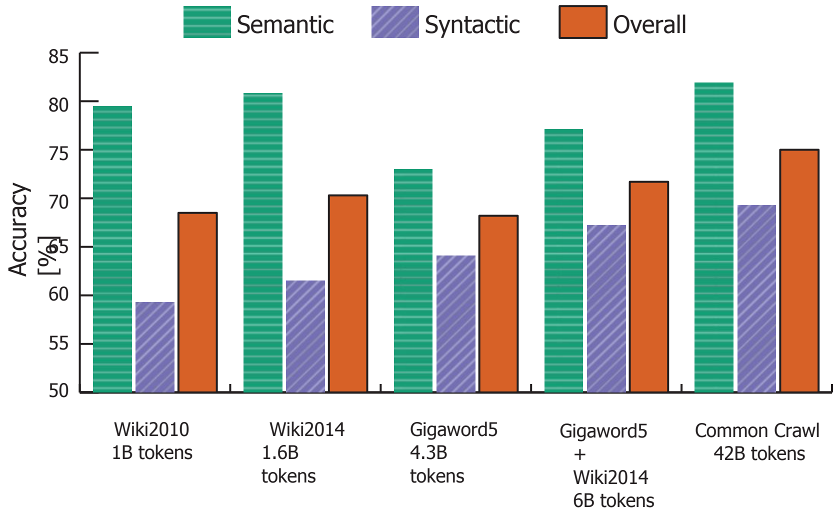


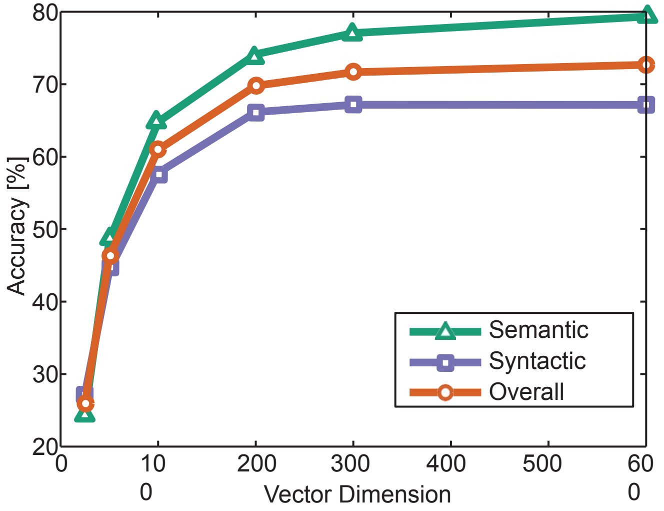
Model	Dim.	Size	Sem.	Syn.	Tot.
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	$\underline{67.4}$	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	$\underline{77.4}$	67.0	$\underline{71.7}$

Analogy evaluation and hyperparameters

- More data helps
- Wikipedia is better than news text!

- Dimensionality
- Good dimension is ~300





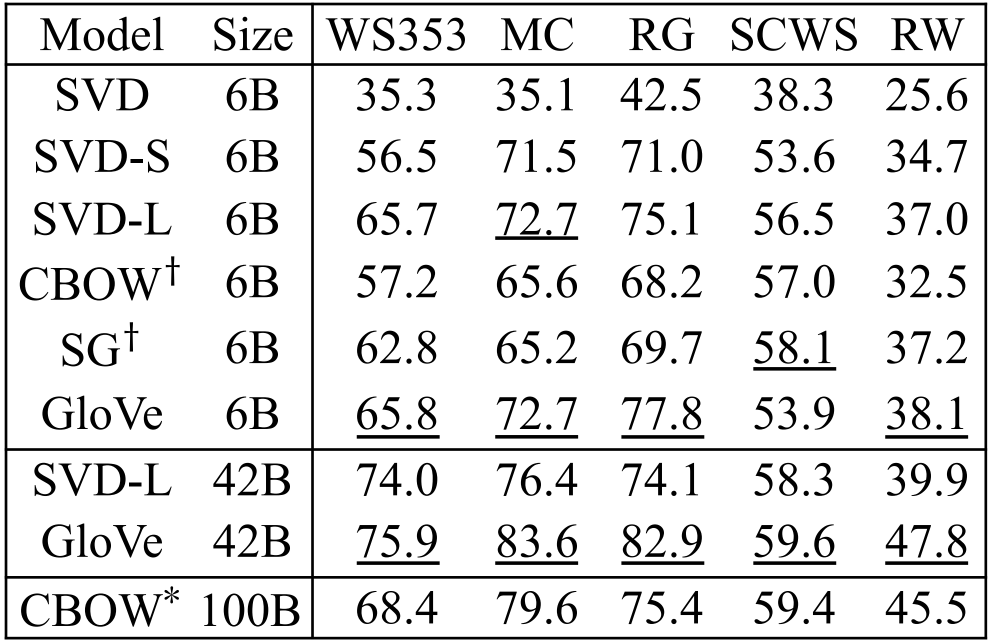
Another intrinsic word vector evaluation

- Word vector distances and their correlation with human judgments
- Example dataset: SimLex999 http://www.cl.cam.ac.uk/~fh295/simlex.html

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Correlation evaluation

- Word vector distances and their correlation with human judgments



Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	$\underline{72.7}$	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	$\underline{58.1}$	37.2
GloVe	6B	$\underline{65.8}$	$\underline{72.7}$	$\underline{77.8}$	53.9	$\underline{38.1}$
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	$\underline{75.9}$	$\underline{83.6}$	$\underline{82.9}$	$\underline{59.6}$	$\underline{47.8}$
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Extrinsic word vector evaluation

- Extrinsic evaluation of word vectors: All subsequent NLP tasks in this class. More examples soon.
- One example where good word vectors should help directly: **named entity recognition**: identifying references to a person, organization or location

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	88.7	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	93.2	88.3	82.9	82.2

Today’s Agenda

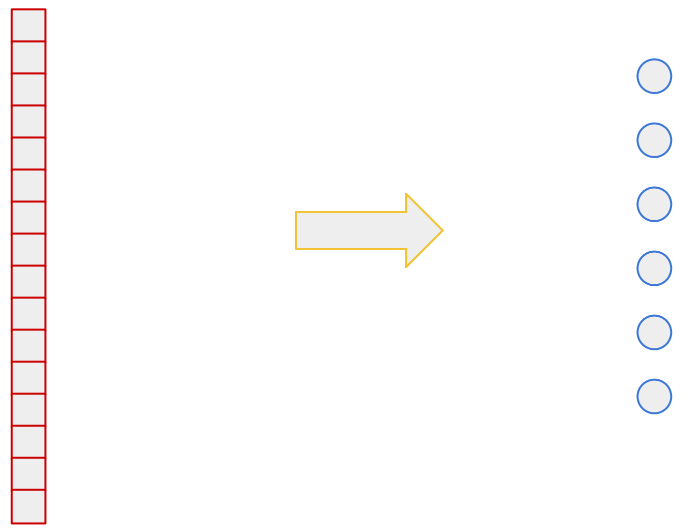
- Word2Vec
- GloVe
- **Sentence representations**
- Applications

Sentence representations

A sentence is a bunch of words

- We learned how to represent words
- We can just concatenate
- But what will we lose?

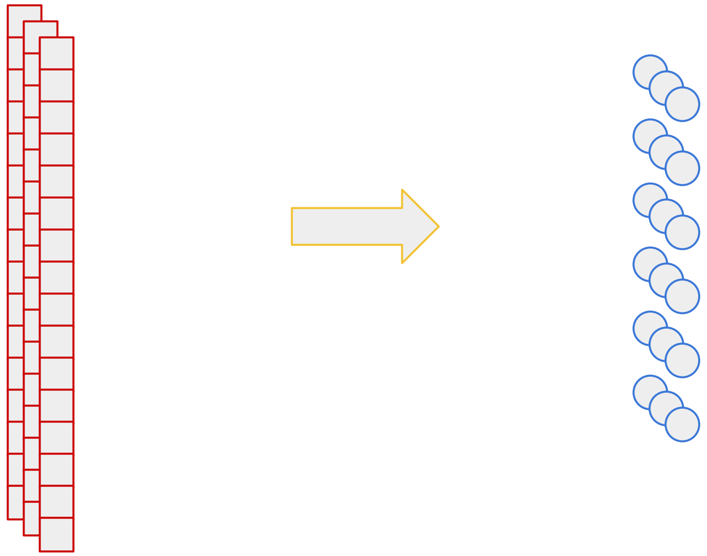
Words now represent meaning



Vocabulary size
~450,000

Word vector
dimension
300

Sentence = Concatenated Word Rep.

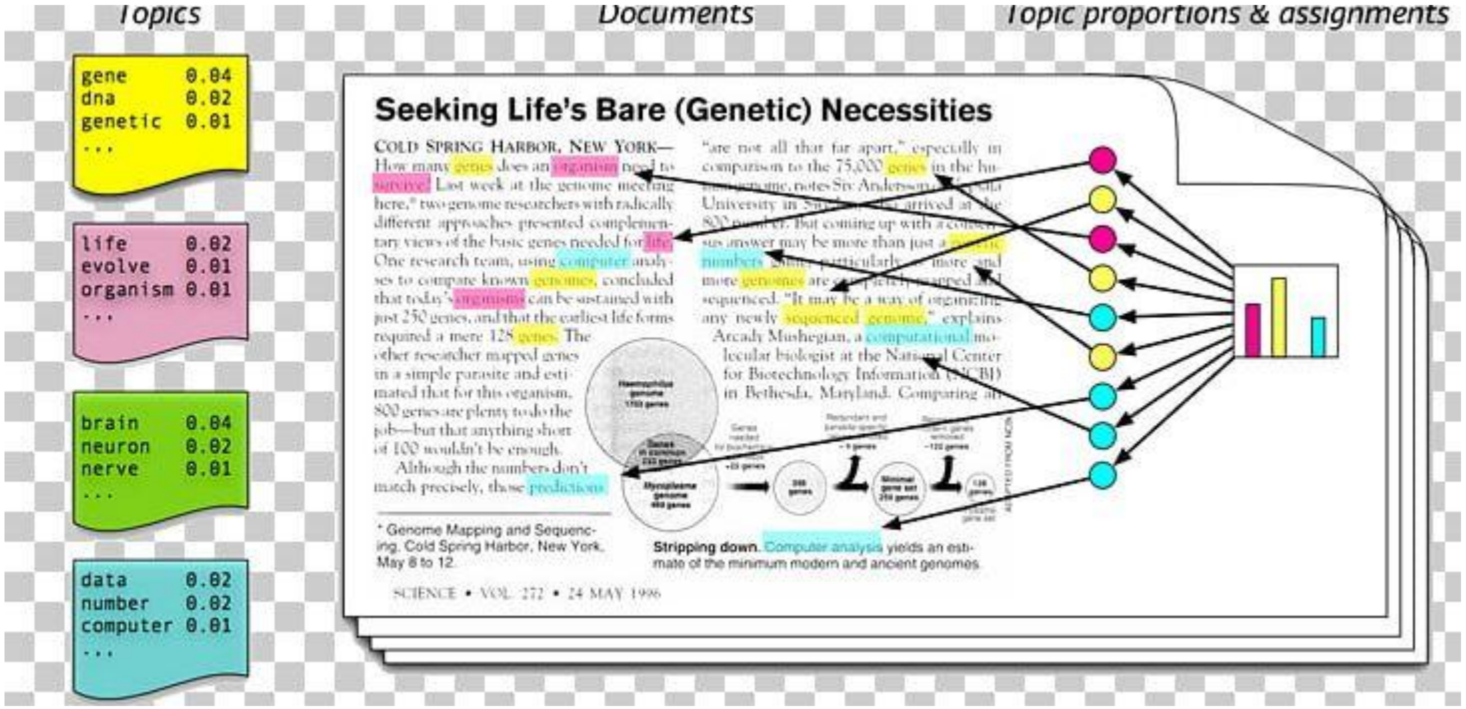


Concatenated Word Rep.

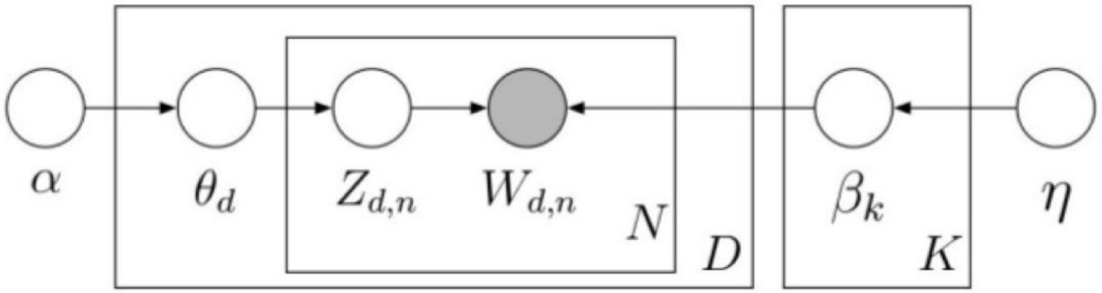
Count-based Representation

- We also learned to represent windows with multiple words
- But this is high-dimensional and sparse
- We can use only some of the dimensions
- We can cluster, but how?

Latent Dirichlet Allocation


Topics
gene 0.04
dna 0.02
genetic 0.01
...
life 0.02
evolve 0.01
organism 0.01
...
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
...
Documents
Seeking Life's Bare (Genetic) Necessities
SCIENCE • VOL. 272 • 24 MAY 1996
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.
Topic proportions & assignments

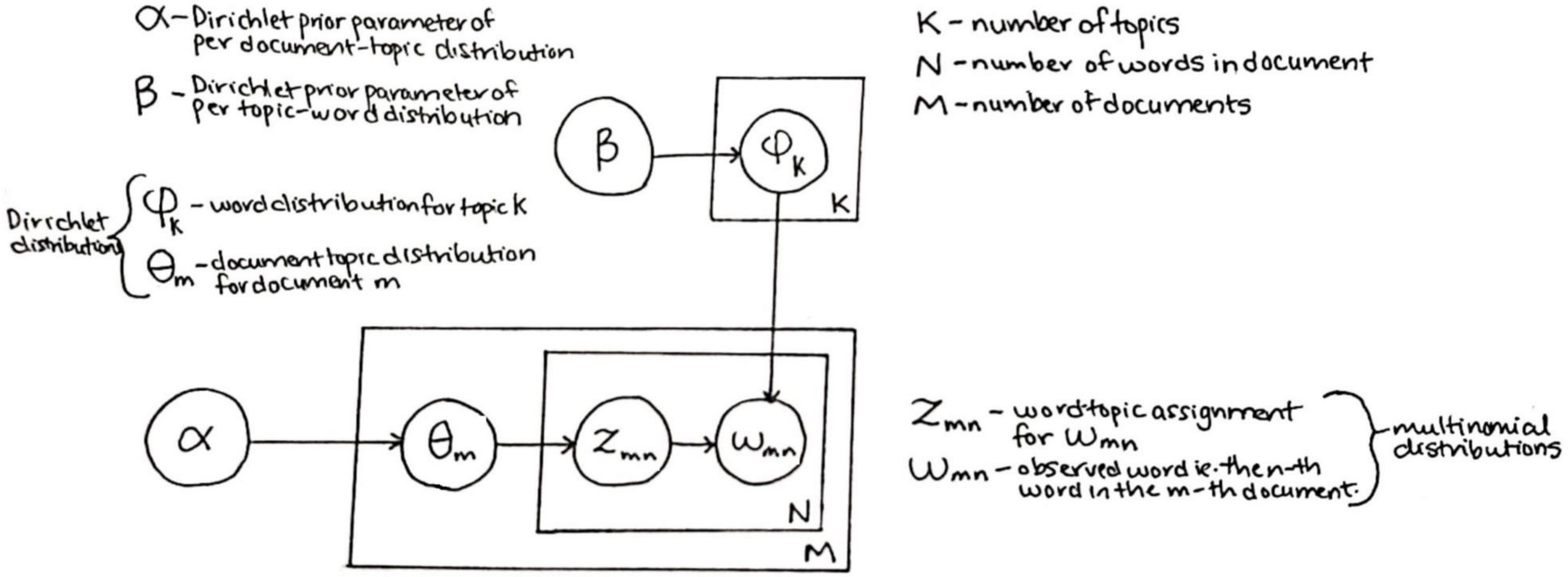
Latent Dirichlet Allocation



K – total number of topics
β_k – topic, a distribution over the vocabulary
D – total number of documents
Θ_d – per-document topic proportions
N – total number of words in a document (it fact, it should be N_d)
$Z_{d,n}$ – per-word topic assignment
$W_{d,n}$ – observed word
α, η – Dirichlet parameters

- Several **inference algorithms** are available (e.g. sampling based)
- A few **extensions** to LDA were created:
  - Bigram Topic Model

Latent Dirichlet Allocation



α – Dirichlet prior parameter of per document-topic distribution

β – Dirichlet prior parameter of per topic-word distribution

Dirichlet distribution {
φ_k – word distribution for topic k
θ_m – document topic distribution for document m

K – number of topics
N – number of words in document
M – number of documents

Z_{mn} – word topic assignment for W_{mn}
W_{mn} – observed word i.e. the n-th word in the m-th document.
} multinomial distributions

Today’s Agenda

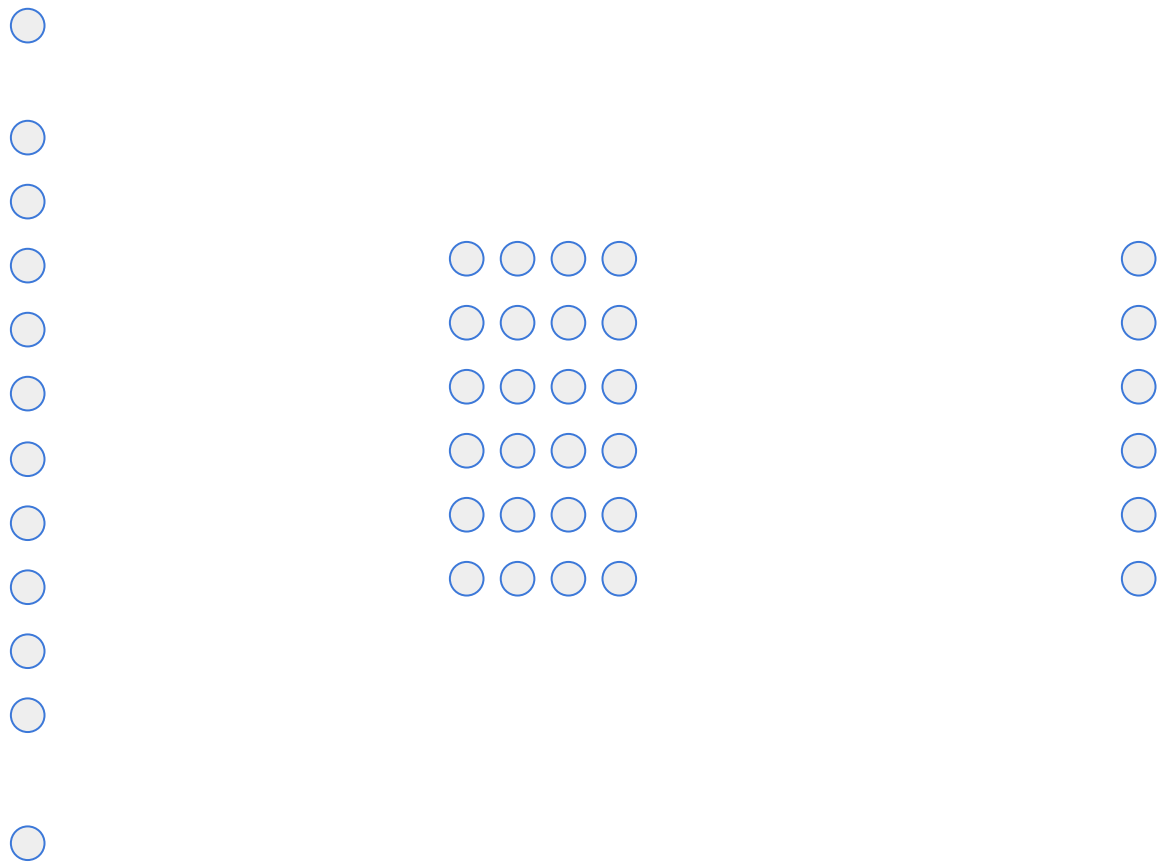
- Word2Vec
- GloVe
- Sentence representations
- **Applications**

Applications

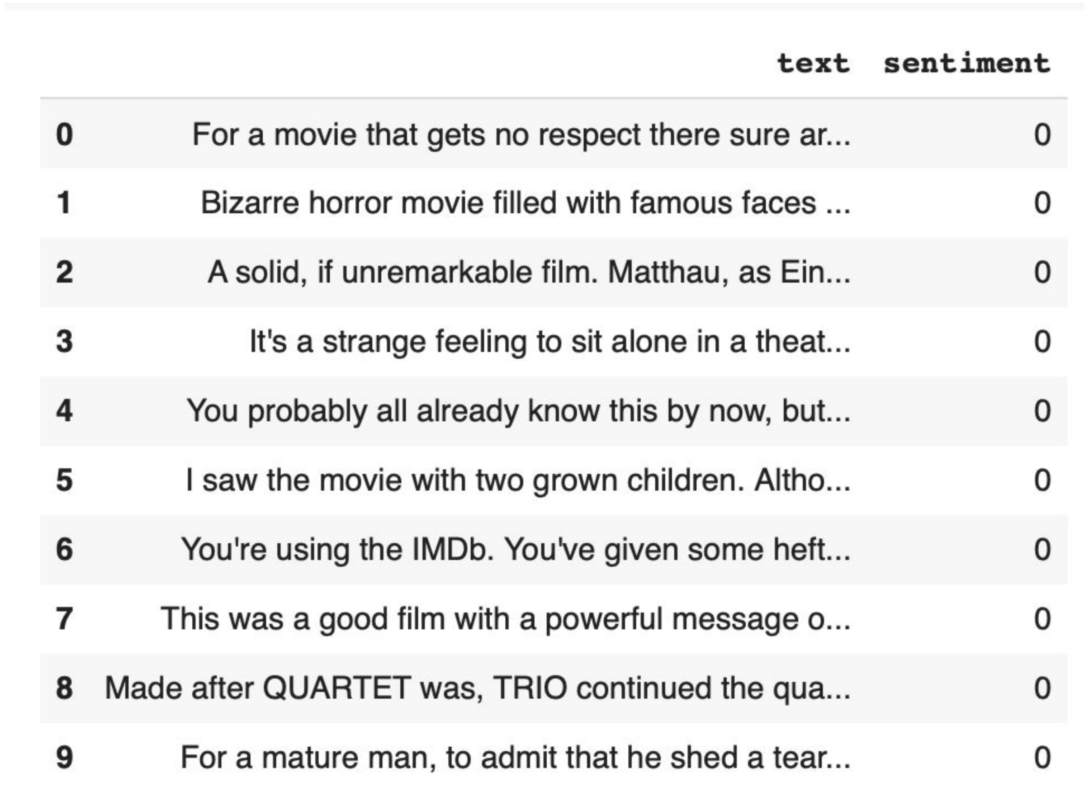
Let’s build some models together (on board)

- Sentiment Classifier
- Named Entity Recognizer
- Part-of-Speech Tagger

Sentence Representation

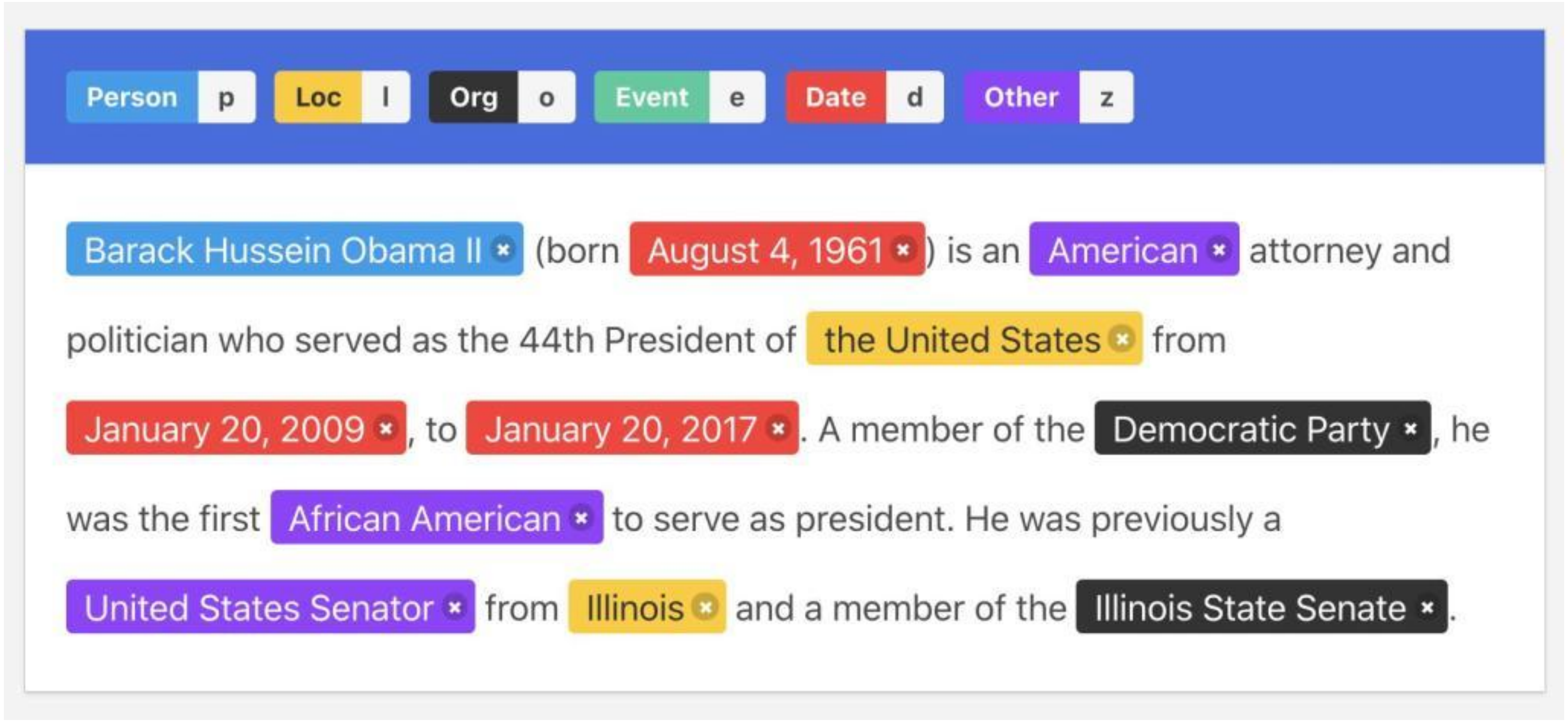


Sentiment Classification



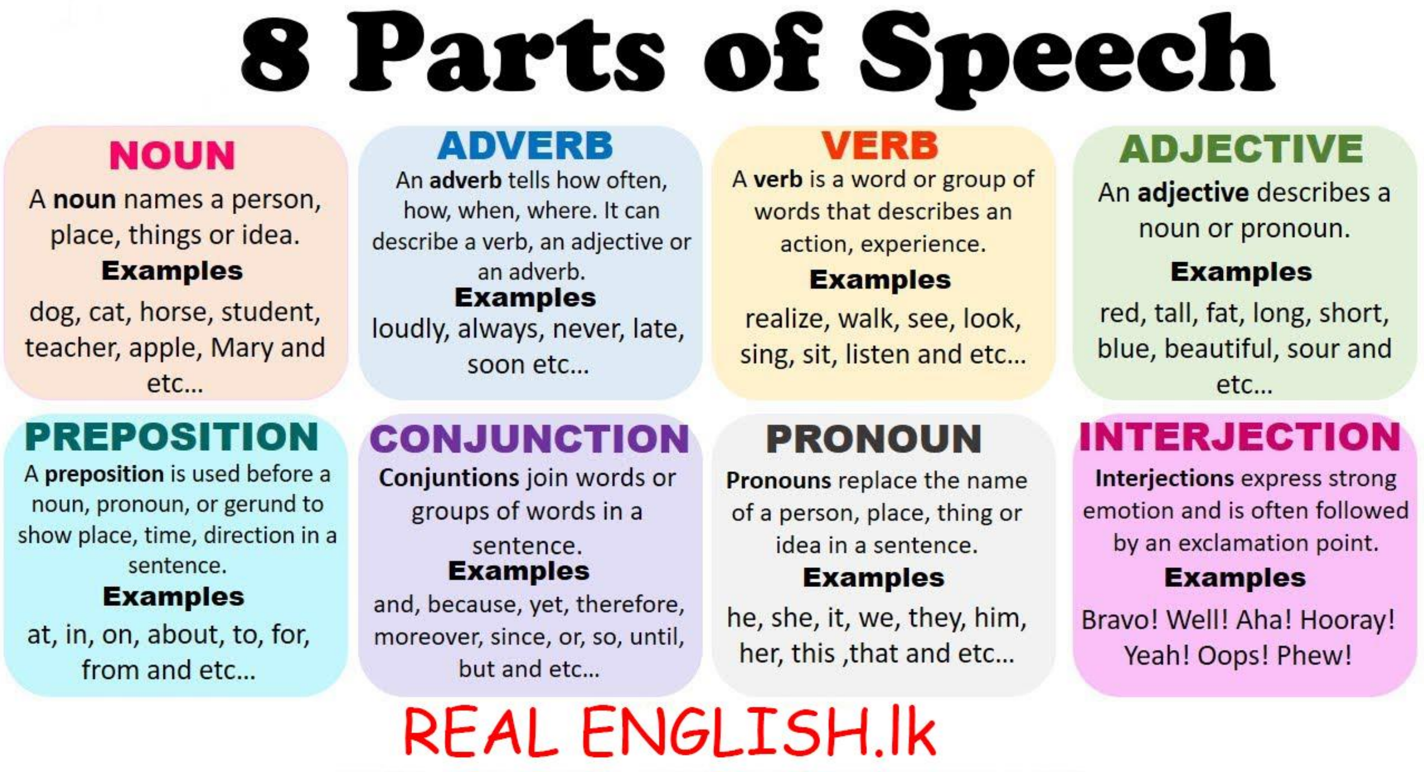
	text	sentiment
0	For a movie that gets no respect there sure ar...	0
1	Bizarre horror movie filled with famous faces ...	0
2	A solid, if unremarkable film. Matthau, as Ein...	0
3	It's a strange feeling to sit alone in a theat...	0
4	You probably all already know this by now, but...	0
5	I saw the movie with two grown children. Altho...	0
6	You're using the IMDb. You've given some heft...	0
7	This was a good film with a powerful message o...	0
8	Made after QUARTET was, TRIO continued the qua...	0
9	For a mature man, to admit that he shed a tear...	0

Named Entity Recognition



Barack Hussein Obama II (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate.

Parts of Speech



8 Parts of Speech

NOUN
A **noun** names a person, place, things or idea.
Examples
dog, cat, horse, student, teacher, apple, Mary and etc...

ADVERB
An **adverb** tells how often, how, when, where. It can describe a verb, an adjective or an adverb.
Examples
loudly, always, never, late, soon etc...

VERB
A **verb** is a word or group of words that describes an action, experience.
Examples
realize, walk, see, look, sing, sit, listen and etc...

ADJECTIVE
An **adjective** describes a noun or pronoun.
Examples
red, tall, fat, long, short, blue, beautiful, sour and etc...

PREPOSITION
A **preposition** is used before a noun, pronoun, or gerund to show place, time, direction in a sentence.
Examples
at, in, on, about, to, for, from and etc...

CONJUNCTION
Conjuntions join words or groups of words in a sentence.
Examples
and, because, yet, therefore, moreover, since, or, so, until, but and etc...

PRONOUN
Pronouns replace the name of a person, place, thing or idea in a sentence.
Examples
he, she, it, we, they, him, her, this ,that and etc...

INTERJECTION
Interjections express strong emotion and is often followed by an exclamation point.
Examples
Bravo! Well! Aha! Hooray! Yeah! Oops! Phew!

REAL ENGLISH.lk