TTIC 31190: Natural Language Processing

Lecture 3: Word Representations

Fall 2023

Announcements

- TA (Jiamin Yang) Tutorial Sessions & Office Hours
  - Fridays 3 pm – 4 pm; TTIC Room 530
  - This week and next: tutorials on Python programming (numpy, PyTorch, etc.)
  - Office hour 4 pm – 5 pm
- Assignment 1 to be released today; due in two weeks

Recap

- Linguistic Morphology
- Lexical Semantics
- Word Tokenization

Linguistic Morphology

- **morphology**: study of how words are built from morphemes
- **morphemes**: meaning-bearing units in a language, often classified into **stems** and **affixes**
- type/token ratio correlated with morphological richness of a language
- types of word formation: **inflection**, **derivation**, **compounding**
- morphological decomposition is sometimes hierarchical (`unlockable`)

Linguistic Morphology

- **lemmatization**: convert wordform to lemma (may depend on context)
- **stemming:** removing affixes from words to get stems (simple, rule-based)

Lexical Semantics

- **word sense**: discrete representation of an aspect of a word's meaning

- most common words have multiple senses
  - though some sense distinctions are subtle

- semantic relationships among senses:
  - **synonymy**: senses have same meanings, can be used interchangeably
  - **antonymy**: senses are opposites in some dimension of meaning, otherwise are similar
  - **hyponymy** (and **hypernymy**): subclass (or superclass) relationship

Lexical Semantics

- **word sense disambiguation (WSD)**: NLP task of determining intended sense of a word based on its context
  - methods use words from context of the ambiguous word
  - unclear if useful for downstream tasks
  - today often done implicitly as part of another task

Word Tokenization

- to do NLP on some text, we need to preprocess it:
  - tokenize documents into sentences
  - tokenize sentences into tokens

- rule-based tokenizers exist for many languages

- for writing systems without whitespace, tokenization becomes complex (often treated as an NLP problem)

Word Tokenization

- useful terms: **type**, **token**, **type/token ratio**
- when adding data, number of types keeps increasing
- most types are extremely rare (**Zipf's law**)
- Data-driven tokenizers: **Byte Pair Encoding (BPE)**
  - splits words based on data, very common in deep learning

Question

How does ChatGPT (GPT-2 etc.) tokenize texts from different languages, with a unified tokenizer and fixed vocabulary size?

Byte-level BPE (BBPE)

GPT-2 vocabulary size: 50257

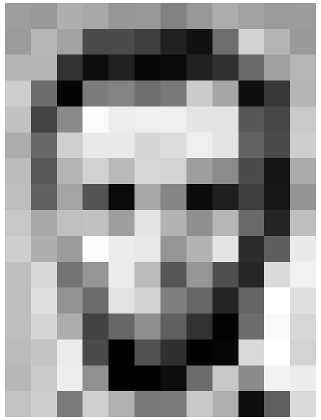
“That’s great 👍”

```
T     54
h     68
a     61
t     74
’   2019
s     73
      20
g     67
r     72
e     65
a     61
t     74
      20
👍  1F44D
```

```
T     54
h     68
a     61
t     74
�     e2
�     80
�     99
s     73
      20
g     67
r     72
e     65
a     61
t     74
      20
�     f0
�     9f
�     91
�     8d
```

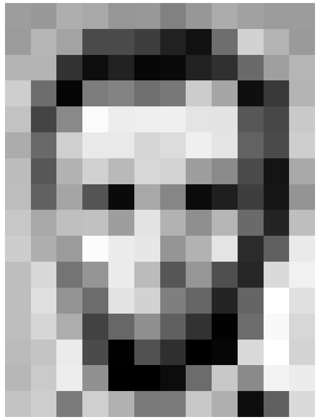
Digital Representations

How does a computer see?

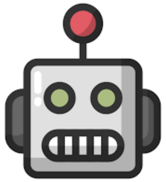


Digital Representations

How does a computer see?

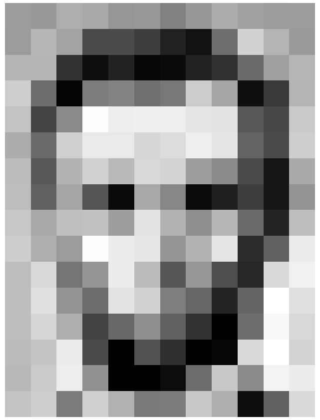


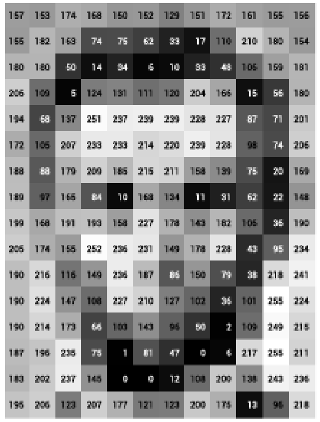
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



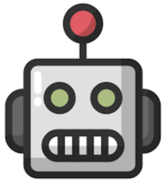
Digital Representations

How does a computer see?





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



Digital Representations

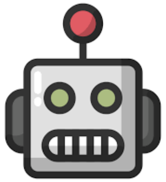
How does a computer read?

`Birds  are  n’t  real  .`

Digital Representations

How does a computer read?

Birds are n’t real .

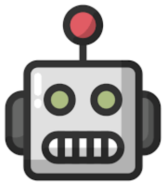


515 834 45 3435 9

Digital Representations

How does a computer read?

`Birds   are   n’t   real   .`



`515     834   45    3435   9`

“Raw” input is often uninteresting/unwieldy to work with.

Word Representations

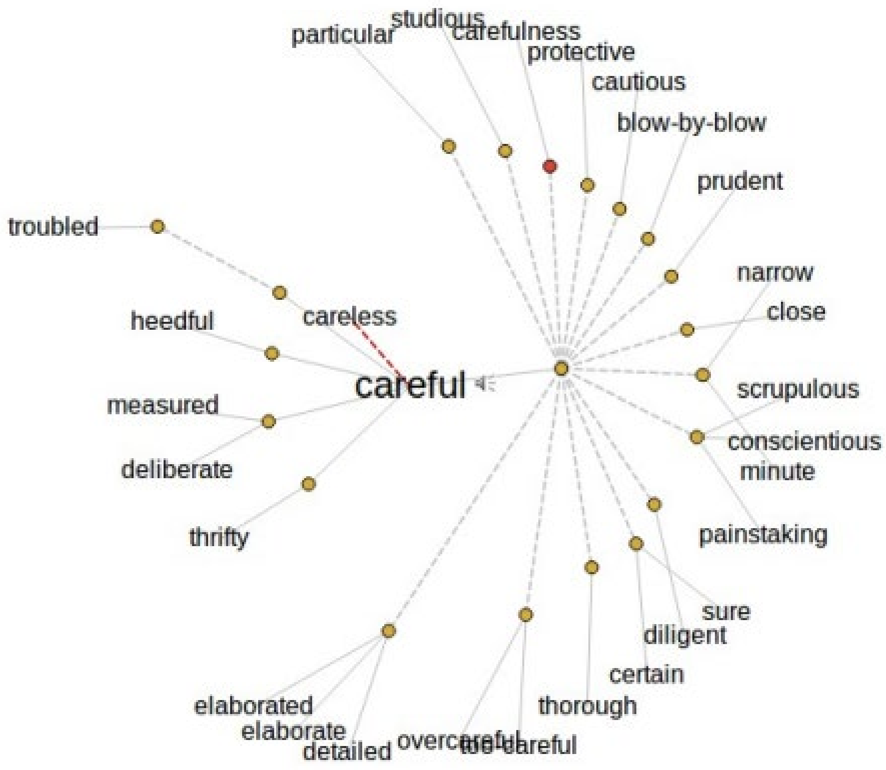
Representing words in **vector** space that captures meaningful structure

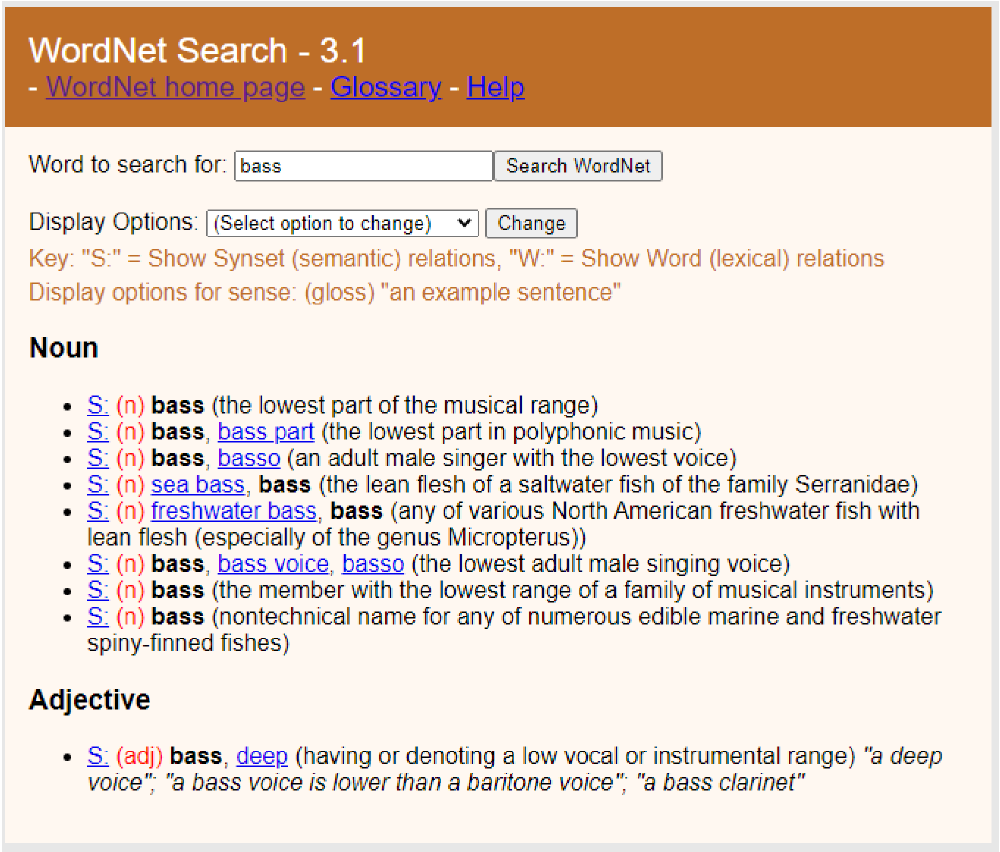
How to represent a word

- Stems and affixes
- Dictionary definition
- Lemma and wordforms
- Senses
- Relationships between words and senses

Annotated Database for Lexical Semantics

- WordNet (Fellbaum, 1998): https://wordnet.princeton.edu/





All-Words WSD

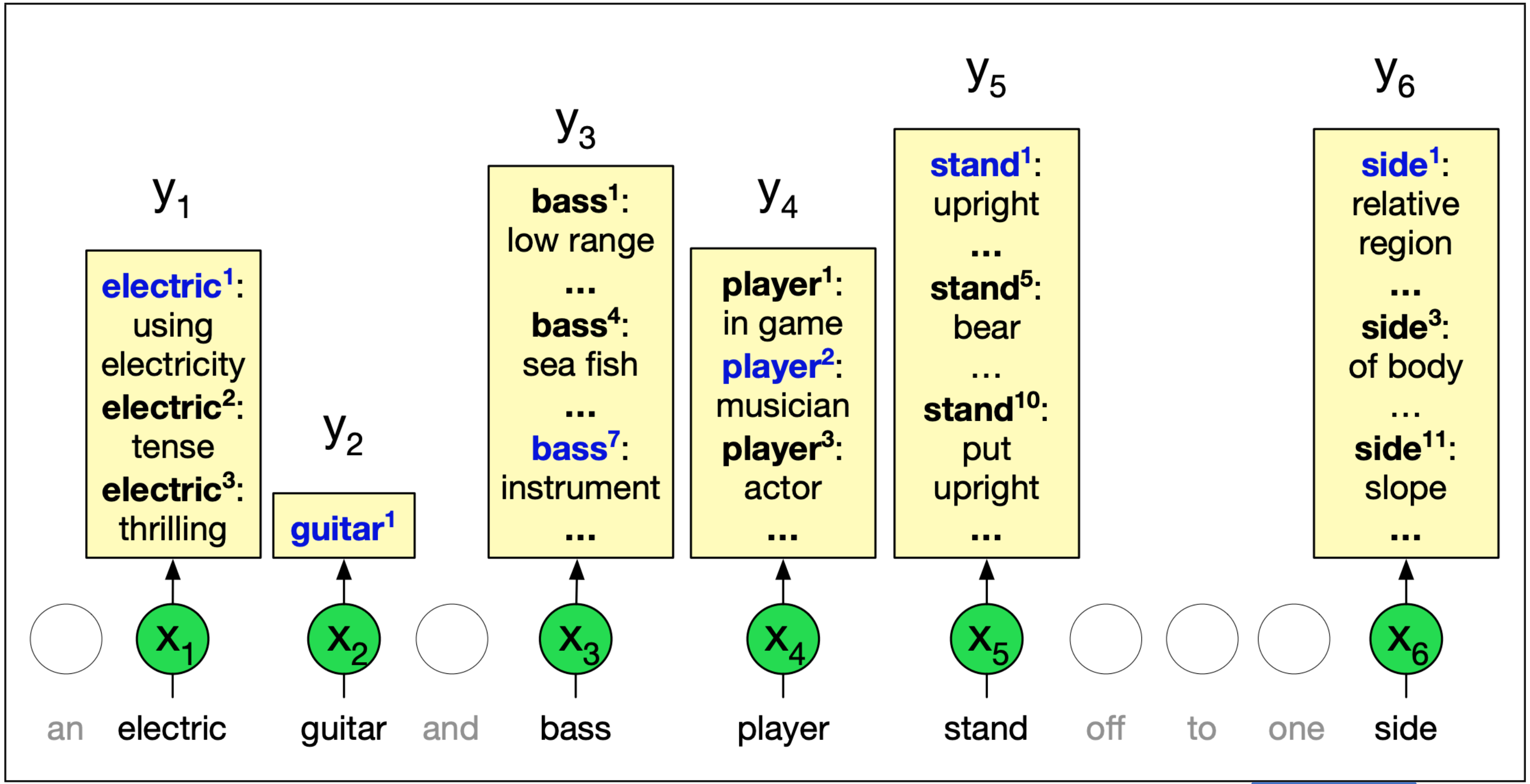


Figure 19.8 The all-words WSD task, mapping from input words (x) to WordNet senses (y). Only nouns, verbs, adjectives, and adverbs are mapped, and note that some words (like *guitar* in the example) only have one sense in WordNet. Figure inspired by Chaplot and Salakhutdinov (2018).

WordNet

- hierarchically organized lexical database
- fine-grained sense inventories, relationships among senses
- originally developed for English; other languages now available
- English WordNet version 3.0 contains:

Category	Unique Strings
Noun	117,798
Verb	11,529
Adjective	22,479
Adverb	4,481

How is "sense" defined in WordNet?

- **synset (synonym set):**
  - set of near-synonyms, instantiates a sense or concept
  - has a **gloss** (roughly, a definition)
- example: $\texttt{chump}_1$ has gloss "*a person who is gullible and easy to take advantage of*"
- $\texttt{chump}_1$ belongs to a synset with 8 other senses:

  $\texttt{fool}_2$, $\texttt{gull}_1$, $\texttt{mark}_9$, $\texttt{patsy}_1$, $\texttt{fall guy}_1$, $\texttt{sucker}_1$, $\texttt{soft touch}_1$, $\texttt{mug}_2$

- each of **these** senses has this same gloss
  - not **every** sense of these words; $\texttt{gull}_2$ is the aquatic bird

WordNet has three synsets for the noun `fool`:

- S: (n) **fool**, sap, saphead, muggins, tomfool (a person who lacks good judgment)
- S: (n) chump, **fool**, gull, mark, patsy, fall guy, sucker, soft touch, mug (a person who is gullible and easy to take advantage of)
- S: (n) jester, **fool**, motley fool (a professional clown employed to entertain a king or nobleman in the Middle Ages)

Ambiguity

one form, multiple meanings → split form

- the three senses of `fool` belong to different synsets

Variability

multiple forms, one meaning → merge forms

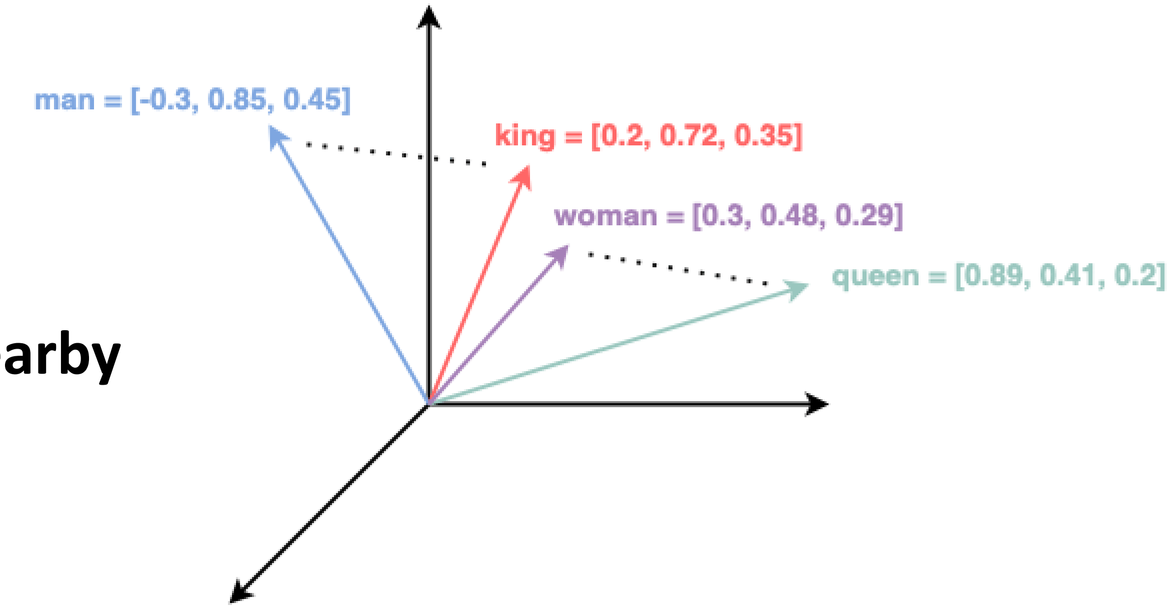
- each synset contains senses of several different words

Hypernyms in WordNet

1. (n) {chump, fool, gull, mark, patsy, fall guy, sucker, soft touch, mug} (a person who is gullible and easy to take advantage of)
    2. (n) {victim, dupe} (a person who is tricked or swindled)
        3. (n) {person, individual, someone, somebody, mortal, soul} (a human being)
            4. (n) {organism, being} (a living thing that has (or can develop) the ability to act or function independently)
                5. (n) {living thing, animate thing} (a living (or once living) entity)
                    6. (n) {whole, unit} (an assemblage of parts that is regarded as a single entity)
                        7. (n) {object, physical object} (a tangible and visible entity; an entity that can cast a shadow)
                            8. (n) {physical entity} (an entity that has physical existence)
                                9. (n) {entity} (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

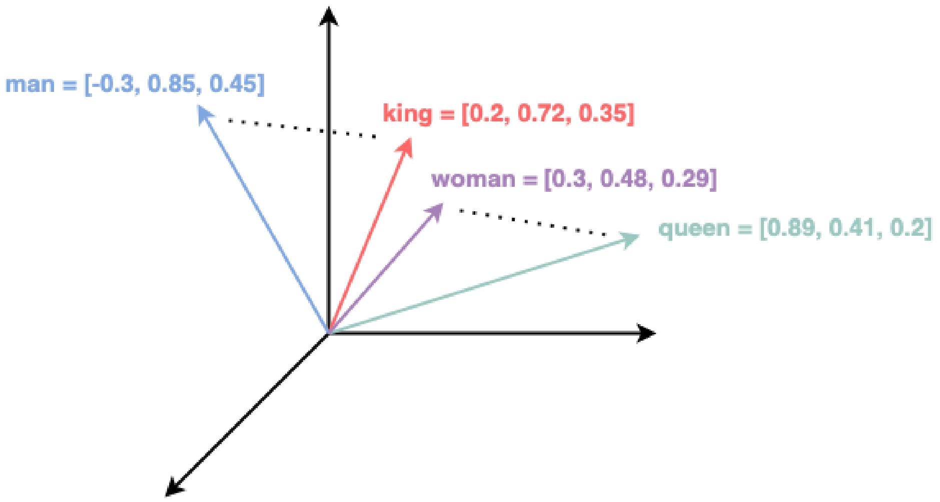
How to represent a word

- Until the ~2010s, in NLP, words == atomic symbols
- Nowadays, **vector representations**, **word == vectors**



Similar words are “**nearby in the vector space**”

How to represent a word



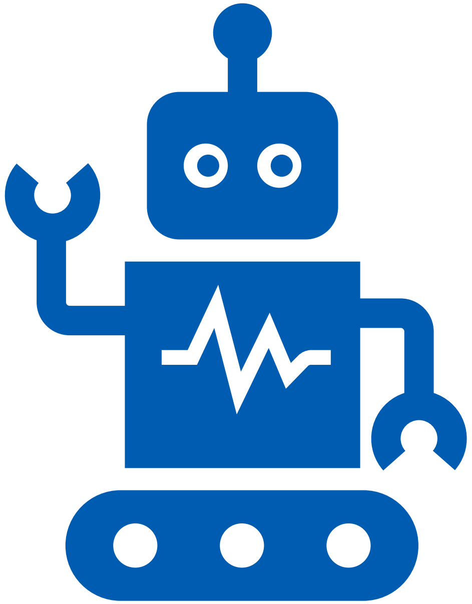
CHAPTER

6

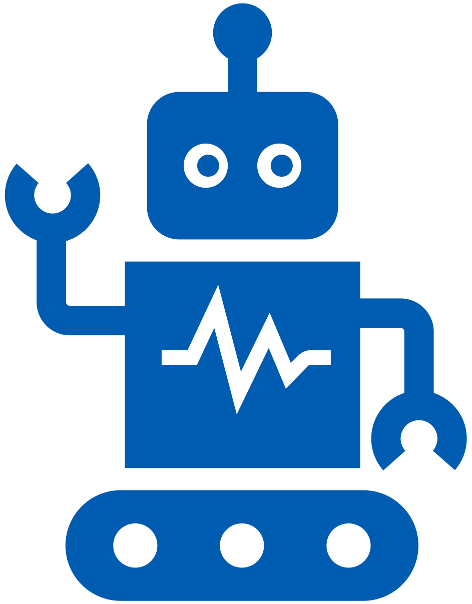
Vector Semantics and Embeddings

荃者所以在鱼，得鱼而忘荃 Nets are for fish;
Once you get the fish, you can forget the net.
言者所以在意，得意而忘言 Words are for meaning;
Once you get the meaning, you can forget the words
庄子(Zhuangzi), Chapter 26

[SLP3, Chapter 6]



cat	chef	chicken	civic	cooked	council …
17	91	253	104	5	6001 …



cat	chef	chicken	civic	cooked	council …
0.1	-0.1	-0.4	0.1	-0.5	0.6
7.9	2.1	2.4	0	-1.1	-1.3
2.4	3.8	9.7	-1.5	7.6	0
-1.3	-0.1	-1.0	2.4	-3.1	3.4
0.5	5.3	3.2	0.2	4.2	-0.6

"embeddings"

Motivations

Variability

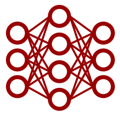
multiple forms,
similar meaning

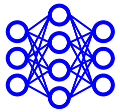
really	reallly	realllly
$\begin{pmatrix} 2.1 \\ -7.9 \\ 8.4 \\ -1.3 \end{pmatrix}$	$\begin{pmatrix} 2.3 \\ -6.1 \\ 7.8 \\ -0.8 \end{pmatrix}$	$\begin{pmatrix} 1.9 \\ -6.8 \\ 7.7 \\ -1.0 \end{pmatrix}$

Representation Learning for Engineering

- Engineering: these representations are often useful for downstream tasks!

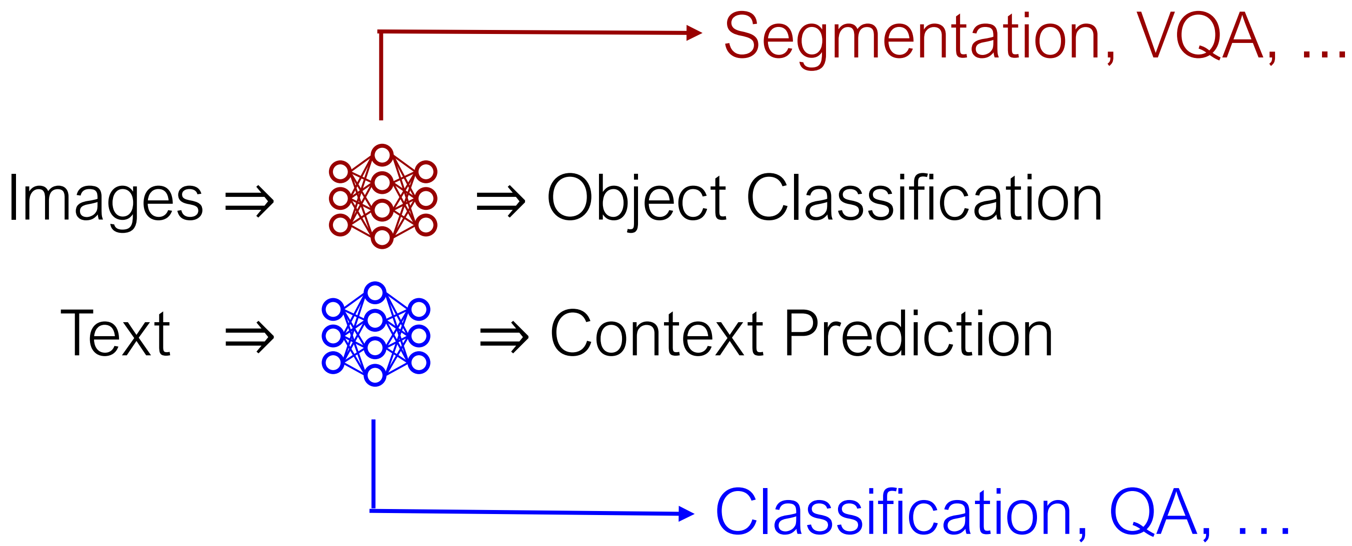
- Transfer learning:

Images ⇒  ⇒ Object Classification

Text ⇒  ⇒ Context Prediction

Representation Learning for Engineering

- Engineering: these representations are often useful for downstream tasks!
- Transfer learning:

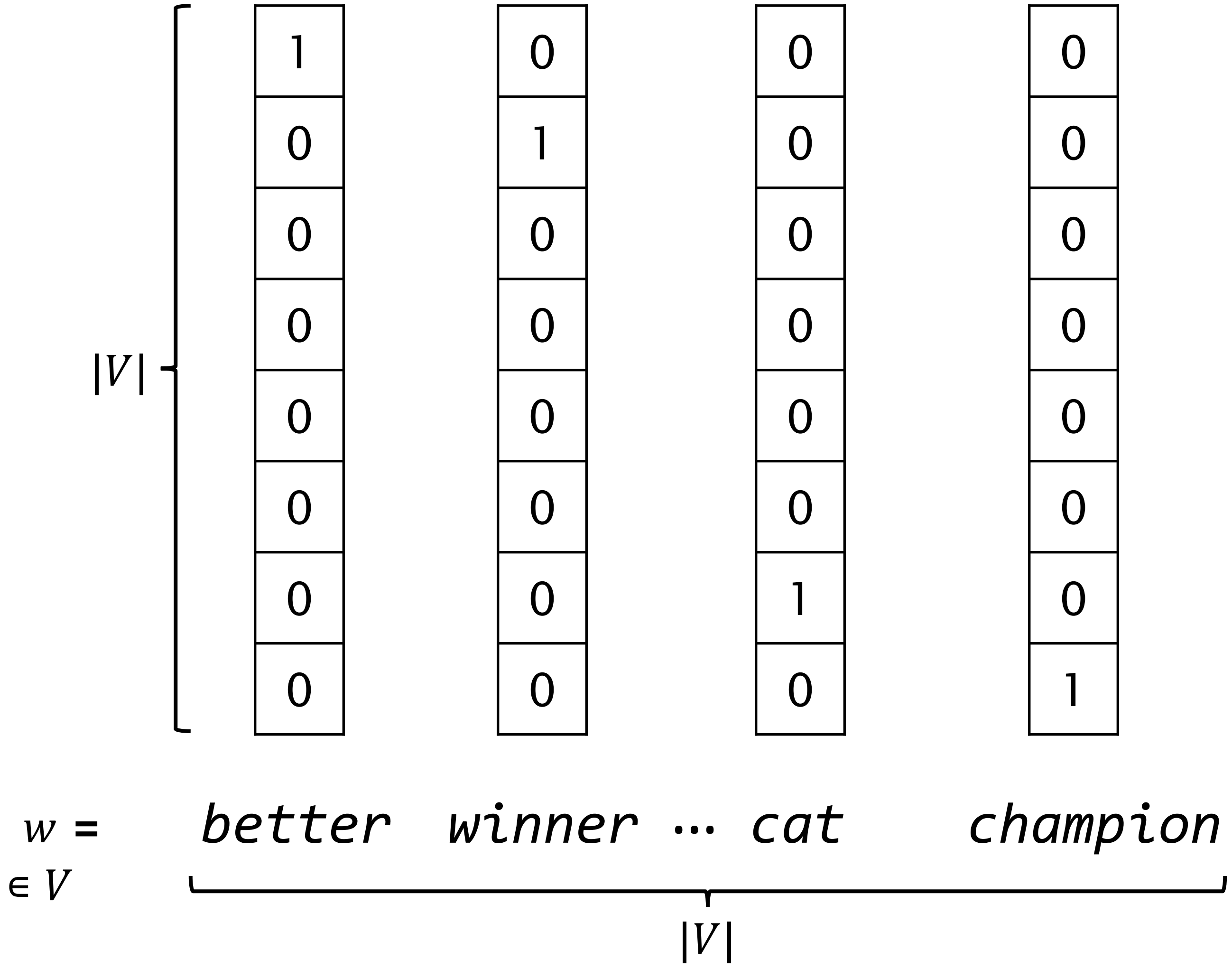


How to represent a word

$$|V|\left\{\begin{array}{cccc}
1 & 0 & 0 & 0\\
0 & 1 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 1 & 0\\
0 & 0 & 0 & 1
\end{array}\right.$$

$$\underset{\in V}{w} = \underbrace{\mathit{better} \quad \mathit{winner} \cdots \mathit{cat} \quad \mathit{champion}}_{|V|}$$

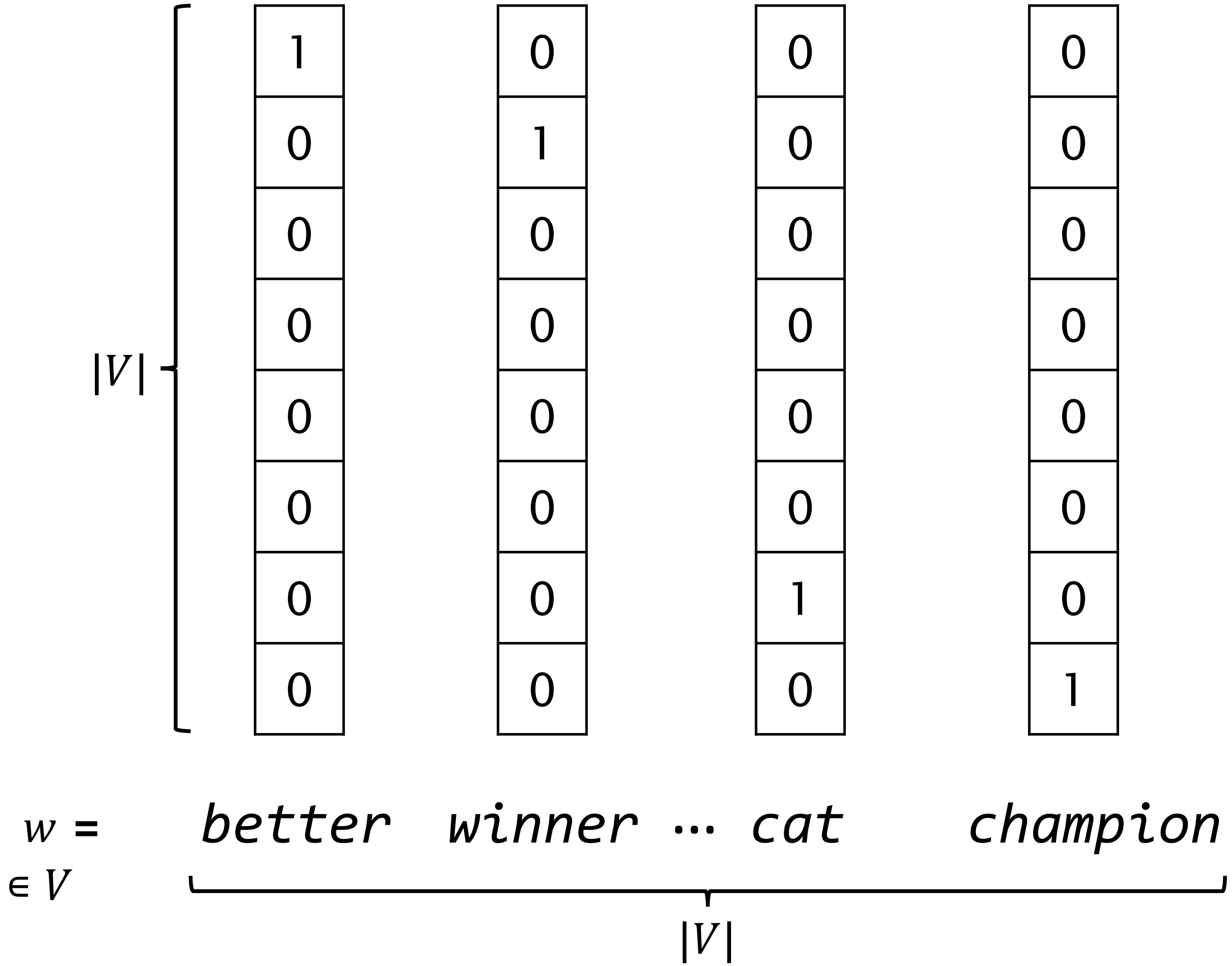
How to represent a word



- "One-hot" representation of words

  $\text{Rep}(w) \in \{0, 1\}^{|V|}$

How to represent a word



- "One-hot" representation of words

  $\text{Rep}(w) \in \{0,1\}^{|V|}$

$|V|$ could be very large!
(e.g. 50K)

How to represent a word

$$
\underbrace{
\begin{array}{cccccc}
 & \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} & & \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix} \\
w = & \mathit{better} & \mathit{winner} & \cdots & \mathit{cat} & \mathit{champion}
\end{array}}_{|V|}
$$

$|V|$

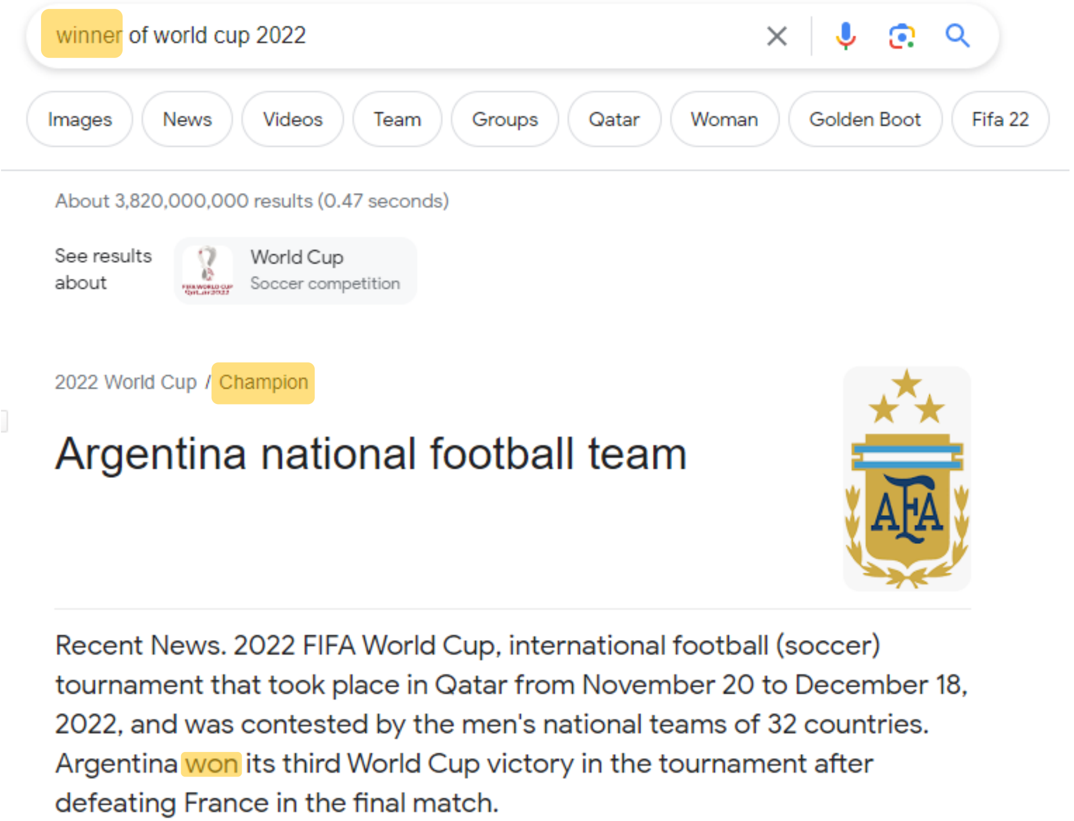
$\in V$

How to represent a word

$$|V| \left[\begin{array}{cccccc} \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} & \boxed{\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix}} & & \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix} & & \boxed{\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}} \end{array} \right.$$

$$w = \underbrace{\mathtt{better} \quad \boxed{\mathtt{winner}} \quad \cdots \quad \mathtt{cat} \quad \boxed{\mathtt{champion}}}_{|V|}$$

$\in V$



Vectors are orthogonal!

Word Representation

- What is an ideal word representation?

- It should probably capture information about usage and meaning:
  - Part of speech tags (noun, verb, adj., adv., etc.)
  - The intended sense
  - Semantic similarities (winner vs. champion)
  - Semantic relationships (antonyms, hypernyms, etc.)

Features?

	$better$	$winner$	…	cat	$champion$
Is noun?	0	1		1	1
Is verb?	0	0		0	1
Is adj.?	1	0		0	0
Is animal?	0	0		1	0
…	0	0		0	0
	0	0		0	0
	0	0		1	0
	0	0		0	1

?

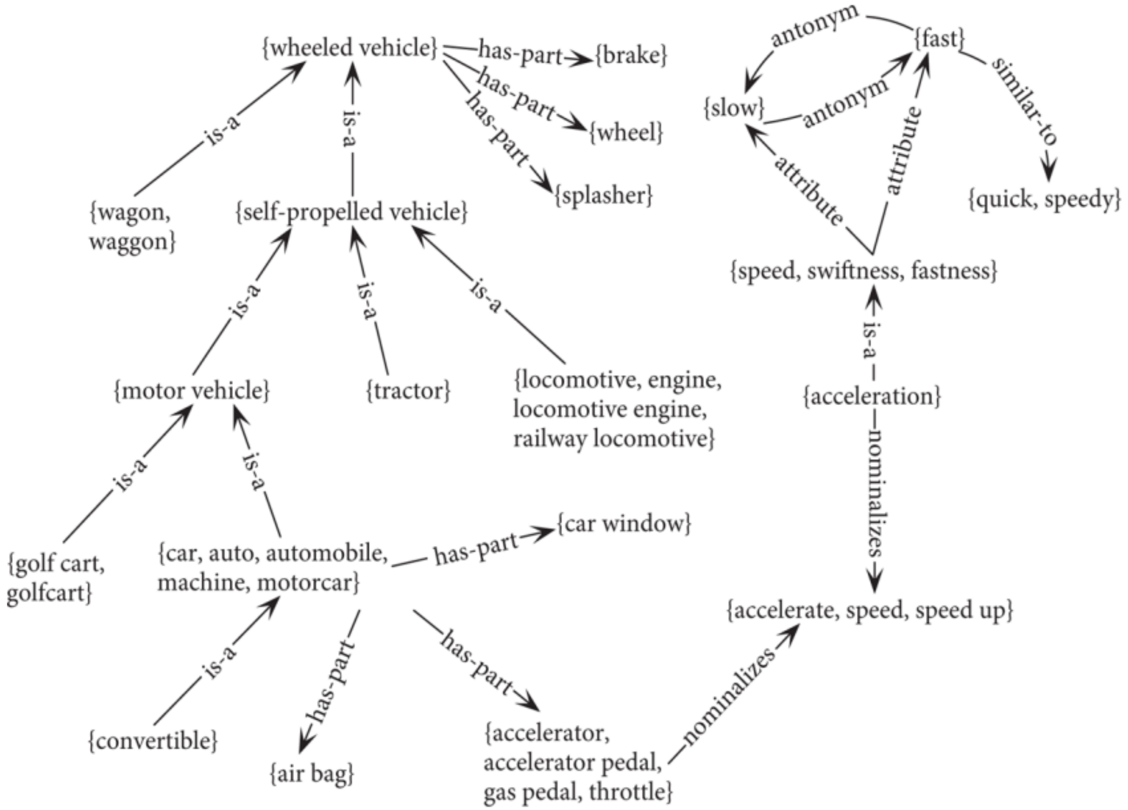
$|V|$

Features?

Is noun?	0	1		1	1	
Is verb?	0	0		0	1	
Is adj.?	1	0		0	0	
Is animal?	0	0		1	0	
...	0	0		0	0	
	0	0		0	0	
	0	0		1	0	?
	0	0		0	1	
	1	1	...	1	1	
	0	0		0	0	
	0	0		0	0	

Features?

WordNet



Word Representation

- What is an ideal word representation?

- It should probably capture information about usage and meaning:
  - Part of speech tags (noun, verb, adj., adv., etc.)
  - The intended sense
  - Semantic similarities (winner vs. champion)
  - Semantic relationships (antonyms, hypernyms, etc.)

Distributional Semantics:
How much of this can we capture from context/data alone?

Main Idea

“The meaning of a word is its use in the language.”

[Ludwig Wittgenstein 1943]

“Usage”:

Words are defined by their environments

(the words around them)

Main Idea

Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don’t have tezgüino before you drive.
4. We make tezgüino out of corn.

What do you think the tezgüino is?

loud
motor oil
tortillas
choices
wine

Main Idea

Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don't have tezgüino before you drive.
4. We make tezgüino out of corn.

What do you think the tezgüino is?

context

term	1	2	3	4
tezgüino	1	1	1	1
loud	0	0	0	0
motor oil	1	0	0	1
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

Main Idea

Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don't have tezgüino before you drive.
4. We make tezgüino out of corn.

What do you think the tezgüino is?

term	context 1	2	3	4	similarity?
tezgüino	1	1	1	1	
loud	0	0	0	0	0
motor oil	1	0	0	1	2
tortillas	0	1	0	1	2
choices	0	1	0	0	1
wine	1	1	1	0	3

Distributional Hypothesis

- These representations encode distributional properties of each word
- The distributional hypothesis: words with similar meaning are used in similar contexts.

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

J.R. Firth, *A Synopsis of Linguistic Theory*, 1957

"The meaning of a word is its use in the language."

[Ludwig Wittgenstein 1943]

"If A and B have almost identical environments we say that they are synonyms."

[Harris 1954]

Distributional Hypothesis

- How can we automate the process of constructing representations of word meaning from its “company”?

- First solution: word-word co-occurrence counts

words we are computing vectors for:

context words:	cat	chef	chicken	civic	cooked	council
the						
cat						
chicken						
city						
cook						

… , the club may also employ a **chef** to prepare and cook food items .

… is up to remy , linguini , and the **chef** colette to cook for many people …

… cooking program the cook and the **chef** with simon bryant , who is …

$$
\begin{array}{r c}
 & \text{chef} \\
\begin{array}{r} \text{the} \\ \text{cat} \\ \text{chicken} \\ \text{city} \\ \text{cook} \end{array} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\end{array}
$$

… , the club may also employ a **chef** to prepare and cook food items .

… is up to remy , linguini , and the **chef** colette to cook for many people …

… cooking program the cook and the **chef** with simon bryant , who is …

window size: $w = 1$

$$\begin{array}{r c} & \text{chef} \\ \begin{array}{r} \text{the} \\ \text{cat} \\ \text{chicken} \\ \text{city} \\ \text{cook} \end{array} & \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array}$$

… , the club may also employ a **chef** to prepare and cook food items .

… is up to remy , linguini , and the **chef** colette to cook for many people …

… cooking program the cook and the **chef** with simon bryant , who is …

window size: $w = 4$

$$
\begin{array}{r c}
 & \text{chef} \\
\begin{array}{r} \text{the} \\ \text{cat} \\ \text{chicken} \\ \text{city} \\ \text{cook} \end{array} & \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 3 \end{pmatrix}
\end{array}
$$

words we are computing vectors for:

context words:

	cat	chef	chicken	civic	cooked	council
the	24708	7410	7853	16486	3463	316380
cat	2336	14	23	0	1	36
chicken	23	21	1640	1	181	7
city	116	89	62	943	7	27033
cook	12	113	34	6	34	51

- once we have word vectors, we can compute similarities!
- many ways to define similarity of two vectors
- a simple way: **dot product** (also called inner product):

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^\top \mathbf{v} = \sum_i u_i v_i$$

$\mathbf{u} =$ a vector

$u_i =$ entry i in the vector

- dot product is large when the vectors have very large (or very negative) values in the same dimensions with dot product as similarity function, let's find the most similar words ("nearest neighbors") to each word:

nearest neighbors

cat	chef	chicken	civic	cooked	council
council	council	council	council	council	council
cat	cat	cat	cat	cat	cat
civic	civic	civic	civic	civic	civic
chicken	chicken	chicken	chicken	chicken	chicken
chef	chef	chef	chef	chef	chef
cooked	cooked	cooked	cooked	cooked	cooked

with dot product as similarity function, let's find the most similar words ("nearest neighbors") to each word:

	cat	chef	chicken	civic	cooked	council
nearest neighbors	council	council	council	council	council	council
	cat	cat	cat	cat	cat	cat
	civic	civic	civic	civic	civic	civic
	chicken	chicken	chicken	chicken	chicken	chicken
	chef	chef	chef	chef	chef	chef
	cooked	cooked	cooked	cooked	cooked	cooked

with dot product as similarity function, let’s find the
rds (“nearest neighbors”) to each word:

$$
\begin{array}{r}
 \\
\text{the} \\
\text{cat} \\
\text{chicken} \\
\text{city} \\
\text{cook}
\end{array}
\begin{array}{c}
\text{council} \\
\begin{pmatrix} 316380 \\ 36 \\ 7 \\ 27033 \\ 51 \end{pmatrix}
\end{array}
$$

	chef	chicken	civic	cooked	council
l	council	council	council	council	council
	cat	cat	cat	cat	cat
	civic	civic	civic	civic	civic
n	chicken	chicken	chicken	chicken	chicken
chef	chef	chef	chef	chef	chef
cooked	cooked	cooked	cooked	cooked	cooked

- dot product is large when vectors have large values in same dimensions, doesn’t control for vector length

- vector length:

$$||\mathbf{u}|| = \sqrt{\sum_i u_i^2}$$

- **cosine similarity**:

$$\frac{\mathbf{u}^\top \mathbf{v}}{||\mathbf{u}||\,||\mathbf{v}||}$$

$\mathbf{u} =$ a vector

$u_i =$ entry i in the vector

this is the cosine of the angle between the two vectors!

now using cosine similarity:

	cat	chef	chicken	civic	cooked	council
nearest neighbors	cat	chef	chicken	civic	cooked	council
	chef	civic	cooked	council	chef	civic
	cooked	cooked	chef	chef	civic	chef
	civic	council	civic	cooked	council	cooked
	council	cat	council	cat	cat	cat
	chicken	chicken	cat	chicken	chicken	chicken

Any issue?

Raw frequency count is probably a bad representation!

Counts of common words are very large, but not very useful

- “the”, “it”, “they”
- Not very informative

Many ways proposed for improving raw counts

Any issue?

Raw frequency count is probably a bad representation!

Counts of common words are very large, but not very useful

- “the”, “it”, “they”
- Not very informative

Many ways proposed for improving raw counts

- **TF-IDF**
- **PMI**
- **word2vec**

TF-IDF

TF (Term Frequency) - IDF (Inverse Document Frequency)

TF-IDF

TF (Term Frequency) - IDF (Inverse Document Frequency)

- Information Retrieval (IR) workhorse!
- A common baseline model
- Sparse vectors
- Words are represented by (a simple function of) the counts of nearby words

TF-IDF Term-Document Matrix

- Consider a matrix of word counts across documents: term-document matrix

document

word

$$\texttt{tf}(\boxed{w}, \boxed{d}) = \#\ \text{of times word } w \text{ appears in document } d \qquad \mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

$$\texttt{idf}(w) = \log \left(\frac{\#\ \text{of documents}}{\#\ \text{of documents in which word } w \text{ occurs}} \right) \qquad \mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

$$\texttt{tf-idf}(w, d) = \texttt{tf}(w, d) \cdot \texttt{idf}(w) \qquad w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

TF-IDF Term-Document Matrix

- Consider a matrix of word counts across documents: term-document matrix

$$\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

TF-IDF Term-Document Matrix

- Consider a matrix of word counts across documents: term-document matrix

$$\text{tf}_{t,d} = \text{count}(t,d)$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

word vector

bag-of-words
(document representation)

TF-IDF Term-Document Matrix

- Consider a matrix of word counts across documents: term-document matrix

document

$$\texttt{tf}(\boxed{w}, \boxed{d}) = \#\text{ of times word } w \text{ appears in document } d \qquad \mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

word

$$\texttt{idf}(w) = \log\left(\frac{\#\text{ of documents}}{\#\text{ of documents in which word } w \text{ occurs}}\right) \qquad \mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

$$\texttt{tf-idf}(w,d) = \texttt{tf}(w,d)\cdot\texttt{idf}(w) \qquad w_{t,d} = \mathrm{tf}_{t,d}\times\mathrm{idf}_t$$

≈ 0 for words like “the”

TF-IDF Term-Document Matrix

- IDF from 37 Shakespeare plays

$$\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

$$\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

TF-IDF Term-Document Matrix

- IDF from 37 Shakespeare plays

$$\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

TF-IDF Variations

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1 + n_t}\right) + 1$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

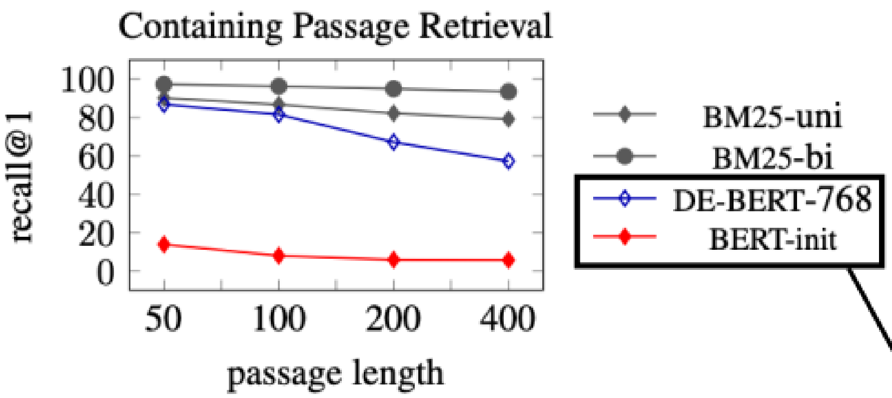
[Image: Wikipedia]

TF-IDF Usage

- TF-IDF was designed for and still excels at document retrieval
- The BM25 model (very similar to TF-IDF) is still a strong document retrieval baseline!

$$\text{BM25}(d, \mathbf{q}) = \sum_{q_i \in \mathbf{q}} \text{idf}_{q_i} \cdot \frac{tf_{q_i,d} \cdot (k_1 + 1)}{tf_{q_i,d} + k_1 (1 - b + b \frac{|D|}{avg|D|})}$$

Recent history in NLP might suggest that learned dense representations should always outperform sparse features, but this is not necessarily true: as shown in Figure 1, the BM25 model (Robertson et al., 2009) can outperform a dual encoder based on BERT, particularly on longer documents (See § 7).



Fancy, computationally expensive neural network models!

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. [Sparse, Dense, and Attentional Representations for Text Retrieval](). *Transactions of the Association for Computational Linguistics*, 9:329–345.

Pointwise Mutual Information (PMI)

Pointwise Mutual Information (PMI)

- consider two random variables, X and Y
- do two events $X = x$ and $Y = y$ occur together more often than if they were independent?

$$\mathrm{pmi}(x, y) = \log_2 \frac{p_{X,Y}(x, y)}{p_X(x)\, p_Y(y)}$$

- if they are independent, PMI = 0

PMI for Word Vectors

- for word vectors,
  X is the **center word**
  Y is the **context word**

$$\mathrm{pmi}(x, y) = \log_2 \frac{p_{X,Y}(x, y)}{p_X(x)\, p_Y(y)}$$

- each probability can be estimated using counts we already computed!

$$p_{X,Y}(x, y) = \frac{\#(x, y)}{N}$$

$$p_X(x) = \frac{\sum_y \#(x, y)}{N}$$

$$p_Y(y) = \frac{\sum_x \#(x, y)}{N}$$

$\#(x, y)$ = co-occurrence count of x and y
N= total count

Top co-occurrence counts with “chicken”

14464	,	1525	or	508	pork
7853	the	1225	for	500	meat
6276	and	1061	‘s	481	be
5931	.	940	fried	479	he
5213	a	906	on	452	such
3963	of	889	was	445	his
3282	in	869	that	417	at
2520	to	828	are	405	soup
2438	"	777	by	389	made
2339	is	746	from	384	rice
2127	with	710	it	375	but
1818	(	600	beef	350	has
1745	)	590	which	330	fish
1640	chicken	557	also	325	other
1594	as	531	an	318	this

Words with largest PMI with "chicken"

10.2	fried	7.0	robot	6.1	pig
9.7	chicken	6.9	burger	6.0	breeds
9.3	pork	6.8	recipe	6.0	vegetable
9.0	beef	6.6	vegetables	6.0	potato
8.7	soup	6.6	potatoes	5.9	goose
7.8	sauce	6.6	goat	5.9	dixie
7.7	curry	6.5	eggs	5.9	kung
7.6	cooked	6.4	cow	5.9	pie
7.5	lamb	6.4	pizza	5.8	menu
7.4	dish	6.4	rice	5.8	steamed
7.3	shrimp	6.3	ribs	5.8	tastes
7.3	egg	6.3	tomatoes	5.7	beans
7.2	sandwich	6.2	cheese	5.7	butter
7.2	dishes	6.2	duck	5.7	barn
7.2	meat	6.1	chili	5.7	breed

words we are computing vectors for:

context words:

	cat	chef	chicken	civic	cooked	council
the	24708	7410	7853	16486	3463	316380
cat	2336	14	23	0	1	36
chicken	23	21	1640	1	181	7
city	116	89	62	943	7	27033
cook	12	113	34	6	34	51

words we are computing vectors for:

context words:

	cat	chef	chicken	civic	cooked	council
the	0.1	-0.1	-0.4	0.1	-0.5	0.6
cat	7.9	2.1	2.4	0	-1.1	-1.3
chicken	2.4	3.8	9.7	-1.5	7.6	0
city	-1.3	-0.1	-1.0	2.4	-3.1	3.4
cook	0.5	5.3	3.2	0.2	4.2	-0.6

$$\mathrm{pmi}(\mathrm{cat}, \mathrm{the}) \quad \mathrm{pmi}(\mathrm{chef}, \mathrm{the}) \quad \ldots$$

using counts:

nearest neighbors

cat	chef	chicken	civic	cooked	council
cat	chef	chicken	civic	cooked	council
chef	civic	cooked	council	chef	civic
cooked	cooked	chef	chef	civic	chef
civic	council	civic	cooked	council	cooked
council	cat	council	cat	cat	cat
chicken	chicken	cat	chicken	chicken	chicken

using PMIs:

nearest neighbors

cat	chef	chicken	civic	cooked	council
cat	chef	chicken	civic	cooked	council
chicken	chicken	cooked	council	chicken	civic
chef	cooked	chef	chef	chef	chicken
cooked	cat	cat	cat	cat	chef
civic	council	council	chicken	council	cooked
council	civic	civic	cooked	civic	cat

Positive PMI (PPMI)

- some have found benefit by truncating PMI at 0 (“positive PMI”)

$$\texttt{PPMI}(u, v) = \max\{0, \texttt{PMI}(u, v)\}$$

- negative PMI: words occur together less than we would expect, i.e., they are anticorrelated

- these anticorrelations may need more data to reliably estimate

- however, negative PMIs do seem reasonable!

Largest PMIs:		PMIs close to zero:		Smallest PMIs:	
10.2	fried	0.003	climbed	-4.6	users
9.7	chicken	0.003	detailing	-4.6	data
9.3	pork	0.002	turkish	-4.7	discussion
9.0	beef	0.002	oaks	-4.7	museum
8.7	soup	0.001	productivity	-4.7	below
7.8	sauce	0.000	swing	-4.8	editors
7.7	curry	-0.001	structures	-4.8	railway
7.6	cooked	-0.001	thirteenth	-4.8	committee
7.5	lamb	-0.001	commentators	-4.8	elected
7.4	dish	-0.001	palmer	-4.9	championship
7.3	shrimp	-0.002	obstacles	-5.0	archive
7.3	egg	-0.003	horns	-5.3	edits
7.2	sandwich	-0.003	burning	-6.1	deletion

words we are computing vectors for:

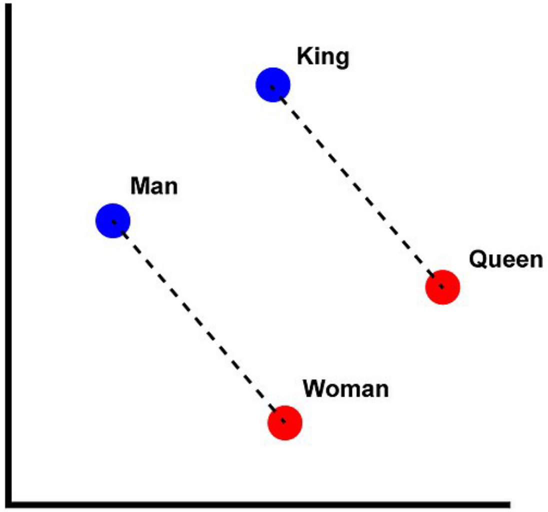
- downside: large context word vocabulary needed for good vectors (1,000 to 10,000)
- hard to work with high-dimensional vectors
- we can reduce dimensionality (SVD, etc.), but this is difficult to scale to large vocabularies

cooked	council
-0.5	0.6
-1.1	-1.3
7.6	0
-3.1	3.4
4.2	-0.6

word2vec

word2vec

- Learning representations with neural networks



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

word2vec

- Learning representations with neural networks

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

[Mikolov et al., 2013]

word2vec

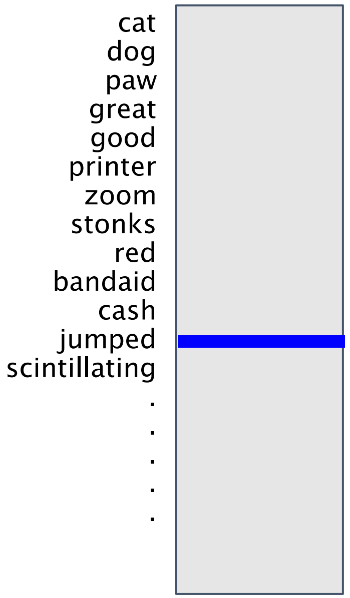
- Learning representations with neural networks

- Instead of counting, train a classifier (neural network) to **predict** context (e.g. neighboring words)

Count-based Distributional Semantics → *Neural Distributional Semantics*

- Training is **self-supervised**: no annotated data required, just raw text
- Word embeddings learned via backpropagation

Neural Word Embeddings



The excited dog jumped over the annoyed cat

The fluffy Samoyed jumped over the backyard fence

The quick brown fox jumped over the lazy dog

Intuition: word embedding for "jumped" should be learned (from random initialization) such that it can well-predict surrounding context.

word2vec

- CBOW (Continuous Bag-of-Words): learn representations that predict a word given context

$$P(w_t \mid w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k})$$

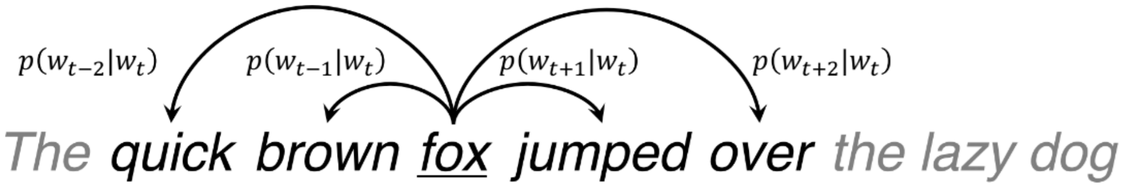
- Skipgram: learn representations that predict the context given a word

$$P(w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k} \mid w_t)$$

word2vec

- CBOW (Continuous Bag-of-Words): learn representations that predict a word given context

- Skipgram: learn representations that predict the context given a word



skipgram

Randomly initialized.
(To be learned via backprop)

$$\begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad U$$

$$\theta = \{W, U\}$$

$W \ : \ V \times d$ input embedding matrix

$U \ : \ V \times d$ output embedding matrix

skipgram

Randomly initialized.
(To be learned via backprop)

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\boxed{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}}$$

Just a (log) linear model!

softmax

$$\begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad\qquad U$$

$$\theta = \{W, U\}$$

$W \; : \; V \times d$ input embedding matrix

$U \; : \; V \times d$ output embedding matrix

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

Pick a window centered at a word and predict the context (window size is a hyperparameter)

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

$$L_t = -\log p_\theta(x_{t-2} \mid x_t) - \log p_\theta(x_{t-1} \mid x_t)$$
$$- \log p_\theta(x_{t+1} \mid x_t) - \log p_\theta(x_{t+2} \mid x_t)$$

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

$$L_t = -\log p_\theta(x_{t-2} \mid x_t) - \log p_\theta(x_{t-1} \mid x_t)$$
$$- \log p_\theta(x_{t+1} \mid x_t) - \log p_\theta(x_{t+2} \mid x_t)$$

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} \textit{a} \\ \textit{aardvark} \\ \textit{able} \\ \textit{are} \\ \vdots \\ \textit{zyzzyva} \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

$$L_t = -\log p_\theta(x_{t-2} \mid x_t) - \log p_\theta(x_{t-1} \mid x_t)$$
$$- \log p_\theta(x_{t+1} \mid x_t) - \log p_\theta(x_{t+2} \mid x_t)$$

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

$$L_t = -\log p_\theta(x_{t-2} \mid x_t) - \log p_\theta(x_{t-1} \mid x_t)$$
$$- \log p_\theta(x_{t+1} \mid x_t) - \log p_\theta(x_{t+2} \mid x_t)$$

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

W $\qquad\qquad$ U

it is a far , far better rest that I go to , than I have ever known

$$L_t = -\log p_\theta(x_{t-2} \mid x_t) - \log p_\theta(x_{t-1} \mid x_t) - \log p_\theta(x_{t+1} \mid x_t) - \log p_\theta(x_{t+2} \mid x_t)$$

skipgram

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad U$$

`it is a far , far better rest that I go to , than I have ever known`

skipgram

$$p_\theta(\mathsf{out} \mid \mathsf{input}) = \frac{\exp(u_{\mathsf{out}} \cdot w_{\mathsf{input}})}{\sum_{v \in V} \exp(u_v \cdot w_{\mathsf{input}})}$$

$$W = \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix}$$

$$U = \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

`it is a far , far better rest that I go to , than I have ever known`

CBOW (Continuous Bag-of-Words)

Use the context to predict the center word

$$
\begin{array}{r}
a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva
\end{array}
\begin{bmatrix}
1.2 & -0.1 & 0.3 & \dots & 0.1 \\
0.2 & 0.7 & -0.4 & \dots & 1.1 \\
-0.7 & 0.5 & 0.6 & \dots & -0.8 \\
0.1 & 0.9 & 0.8 & \dots & 0.7 \\
 & & \vdots & & \\
0.3 & -0.2 & 0.7 & \dots & 0.4
\end{bmatrix}
\qquad
\begin{array}{r}
a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva
\end{array}
\begin{bmatrix}
2.1 & -0.5 & 1.3 & \dots & 1.4 \\
-0.4 & -0.7 & 0.5 & \dots & 0.1 \\
0.2 & 0.1 & 0.4 & \dots & -0.7 \\
0.5 & 0.8 & 0.1 & \dots & 0.4 \\
 & & \vdots & & \\
-0.3 & 0.3 & 0.2 & \dots & 0.6
\end{bmatrix}
$$

$$p_\theta(x_t \mid x_{t-w}, \dots, x_{t+w}) \propto \exp\left(u_{x_t} \cdot \frac{1}{2w} \sum_{k \in \{-w, \dots, -1, 1, w\}} w_{x_{t+k}} \right)$$

W U

it is a far , far better rest that I go to , than I have ever known

$$L_t = -\log p_\theta(x_t \mid x_{t-2}, x_{t-1}, x_{t+1}, x_{t+2})$$

CBOW (Continuous Bag-of-Words)

Use the context to predict the center word

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix}$$

$$W$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

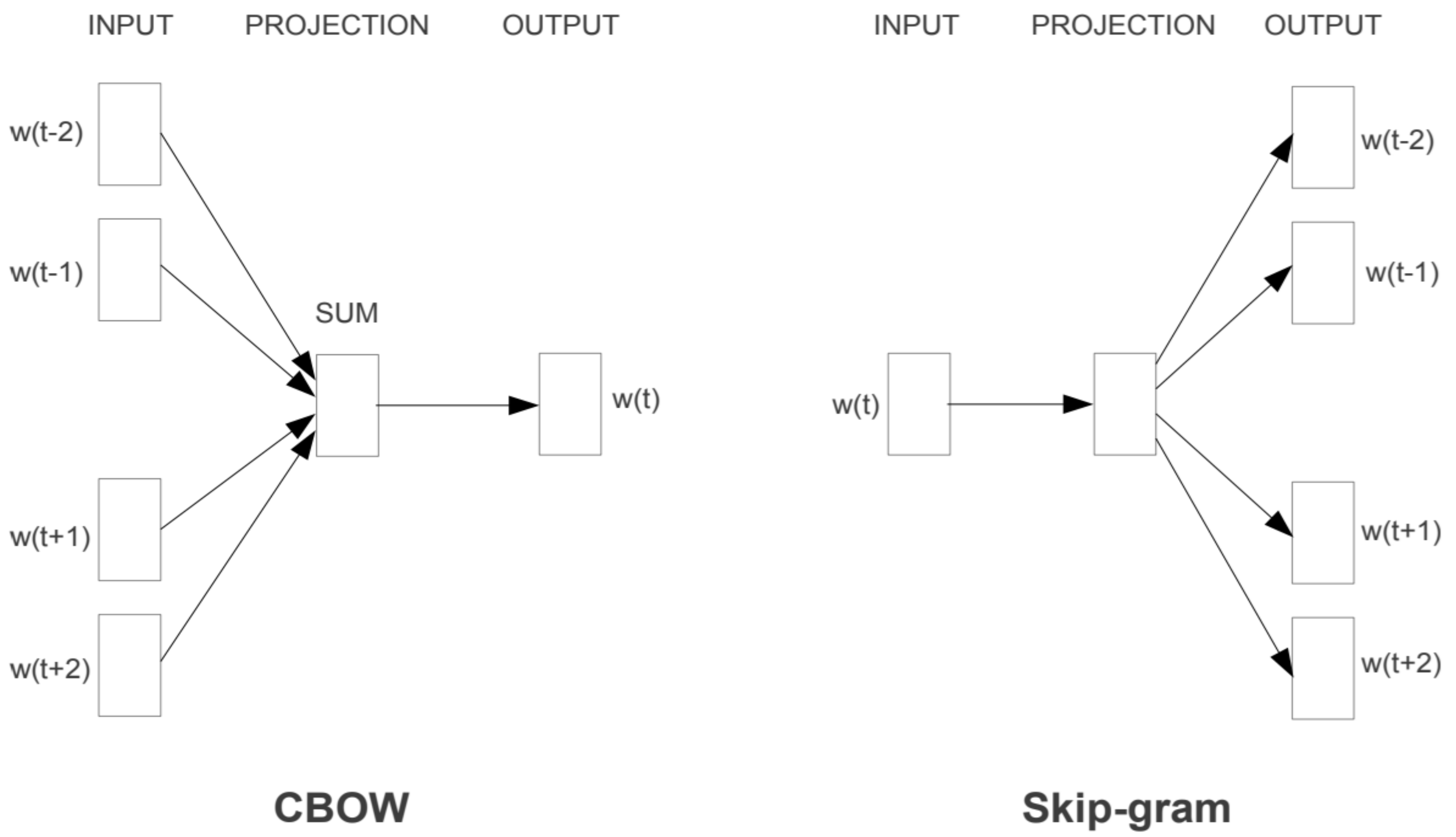
$$U$$

$$p_\theta(x_t \mid x_{t-w}, \dots, x_{t+w}) \propto \exp\left(u_{x_t} \cdot \frac{1}{2w} \sum_{k \in \{-w, \dots, -1, 1, w\}} w_{x_{t+k}} \right)$$

`it is a far , far better rest that I go to , than I have ever known`

$$L_t = -\log p_\theta(x_t \mid x_{t-2}, x_{t-1}, x_{t+1}, x_{t+2})$$

word2vec



[Mikolov et al. 2013]

skipgram w/ Negative Sampling

$$p_\theta(\text{out} \mid \text{input}) = \frac{\exp(u_{\text{out}} \cdot w_{\text{input}})}{\boxed{\sum_{v \in V} \exp(u_v \cdot w_{\text{input}})}}$$

$$\begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 1.2 & -0.1 & 0.3 & \dots & 0.1 \\ 0.2 & 0.7 & -0.4 & \dots & 1.1 \\ -0.7 & 0.5 & 0.6 & \dots & -0.8 \\ 0.1 & 0.9 & 0.8 & \dots & 0.7 \\ & & \vdots & & \\ 0.3 & -0.2 & 0.7 & \dots & 0.4 \end{bmatrix} \qquad \begin{array}{r} a \\ aardvark \\ able \\ are \\ \vdots \\ zyzzyva \end{array} \begin{bmatrix} 2.1 & -0.5 & 1.3 & \dots & 1.4 \\ -0.4 & -0.7 & 0.5 & \dots & 0.1 \\ 0.2 & 0.1 & 0.4 & \dots & -0.7 \\ 0.5 & 0.8 & 0.1 & \dots & 0.4 \\ & & \vdots & & \\ -0.3 & 0.3 & 0.2 & \dots & 0.6 \end{bmatrix}$$

$$W \qquad\qquad\qquad U$$

- Vocabulary size V: 50K – 30M
- Very expensive O(|V|)

skipgram w/ Negative Sampling

- Treat the target word and a neighboring context word as positive examples (x, y)
- Randomly sample other words outside of context to get negative samples (x, v)
- learn to distinguish between true pair (x, y) and negative samples (x, v) with a binary classifier
- New objective

$$\log p((x, y) \text{ is a true pair}) + \boxed{\sum_{k \in C}} \log p((x, k) \text{ is a negative pair})$$

C = Negative Samples

skipgram w/ Negative Sampling

- Treat the target word and a neighboring context word as positive examples (x, y)
- Randomly sample other words outside of context to get negative samples (x, v)
- learn to distinguish between true pair (x, y) and negative samples (x, v) with a binary classifier
- New objective

$$\log p((x, y) \text{ is a true pair}) + \sum_{k \in C} \log p((x, k) \text{ is a negative pair})$$

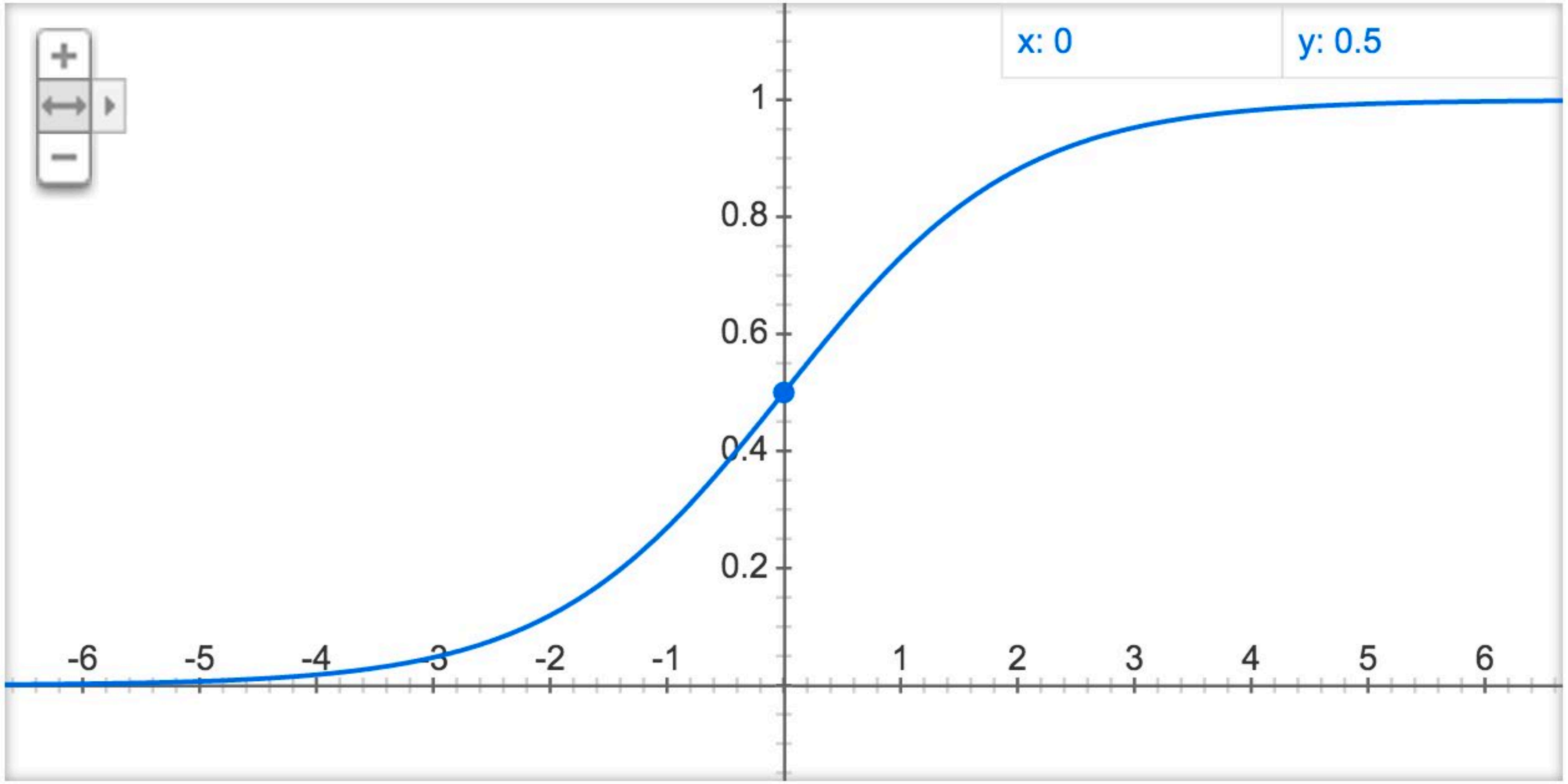
$$p((x, c) \text{ is a true pair}) = \sigma(u_c \cdot w_x) = \frac{1}{1 + \exp(-u_c \cdot w_x)}$$

skipgram w/ Negative Sampling

- Treat the target word and a neighboring context word as positive examples (x, y)
- Randomly sample other words outside of context to get negative samples (x, v)
- learn to distinguish between true pair (x, y) and negative samples (x, v) with a binary classifier
- New objective

$$\log p((x,y) \text{ is a true pair}) + \sum_{k \in C} \log p((x,k) \text{ is a negative pair})$$
$$= \log \sigma(u_y \cdot w_x) + \sum_{k \in C} \log(\sigma(-u_k \cdot w_x))$$

(logistic) sigmoid: $\sigma(x) = \dfrac{1}{1 + \exp\{-x\}}$



skipgram w/ Negative Sampling

- Treat the target word and a neighboring context word as positive examples (x, y)
- Randomly sample other words outside of context to get negative samples (x, v)
- learn to distinguish between true pair (x, y) and negative samples (x, v) with a binary classifier
- New objective

$$\log p((x,y) \text{ is a true pair}) + \sum_{k \in C} \log p((x,k) \text{ is a negative pair})$$

Much cheaper to compute: O(|C|)

Choosing negative samples

- According to unigram probabilities $P(w)$
- More common to choose from a flattened version

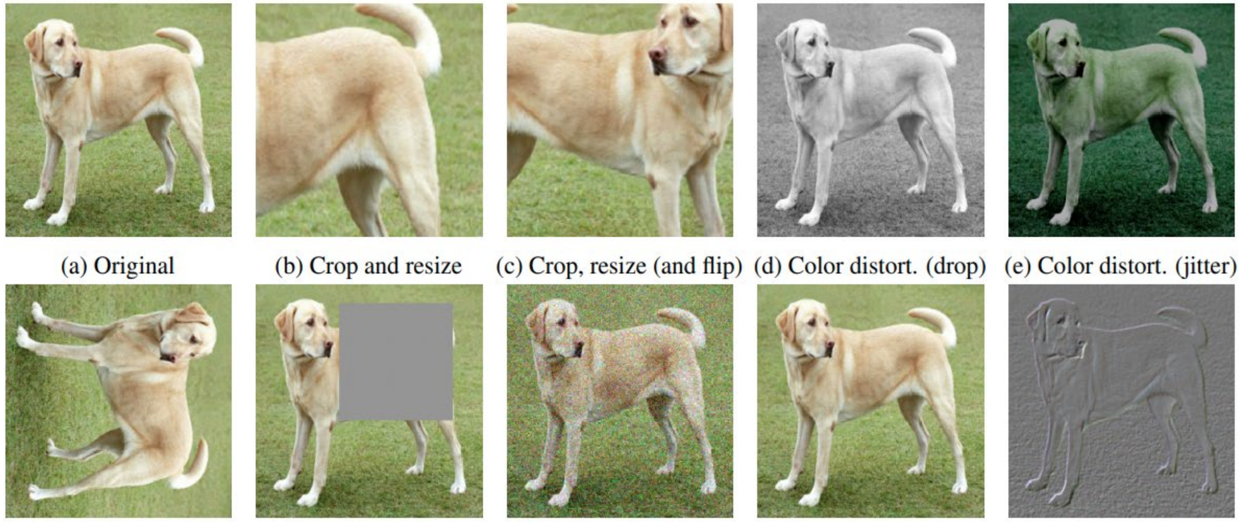
$$P_{\alpha}(w) = \frac{count(w)^{\alpha}}{\sum_{w} count(w)^{\alpha}}$$

- From Mikolov et al. (2013):
  - $\alpha = 0.75$ works well empirically (why?)
  - Usually 2-20 sampled negative words

Contrastive Learning

- Learning to contrast positive vs. negative samples is a very powerful idea!

  - Representation learning



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

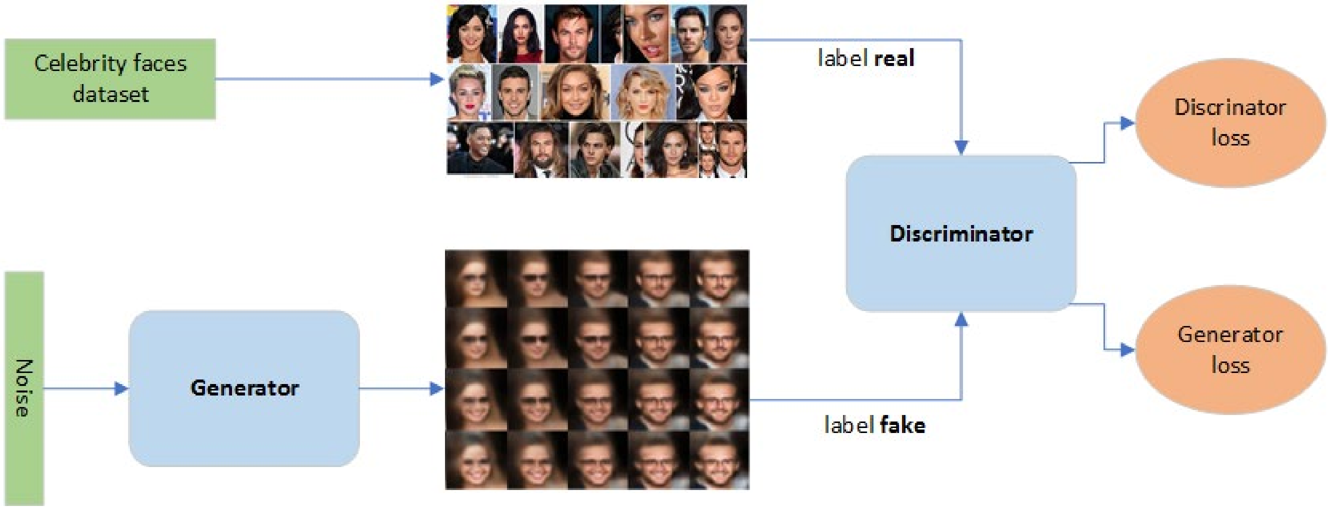
(f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

[SimCLR; Chen et al. 2020]

Contrastive Learning

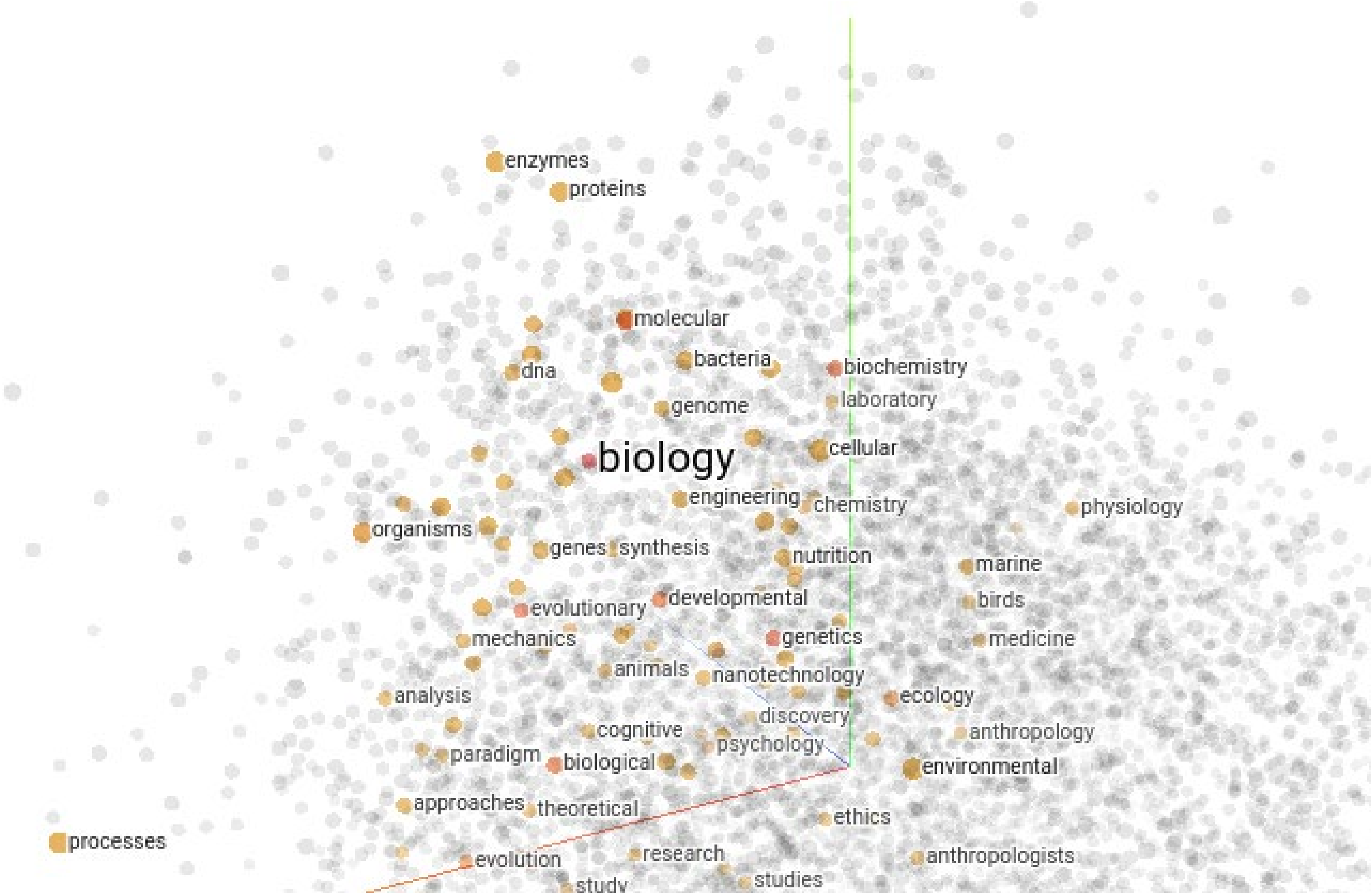
- Learning to contrast positive vs. negative samples is a very powerful idea!

  - Density estimation: Generative Adversarial Networks (GANs)



[Goodfellow et al. 2014]

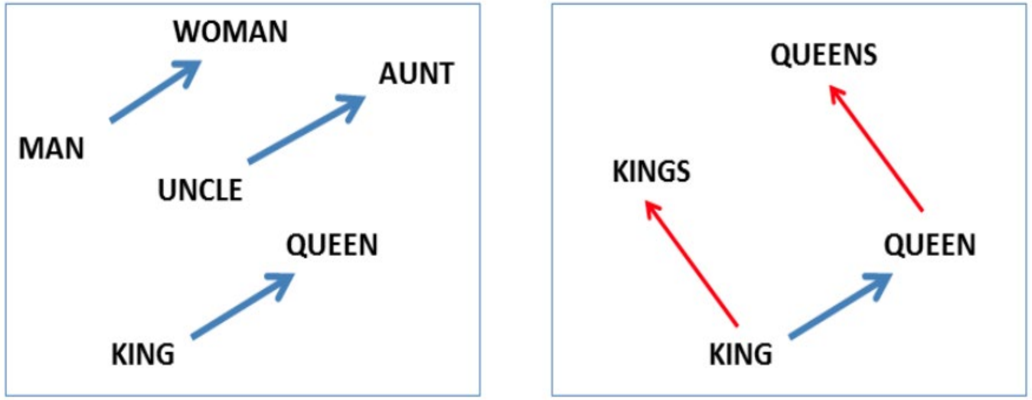
Word2vec Embeddings



[https://projector.tensorflow.org/]

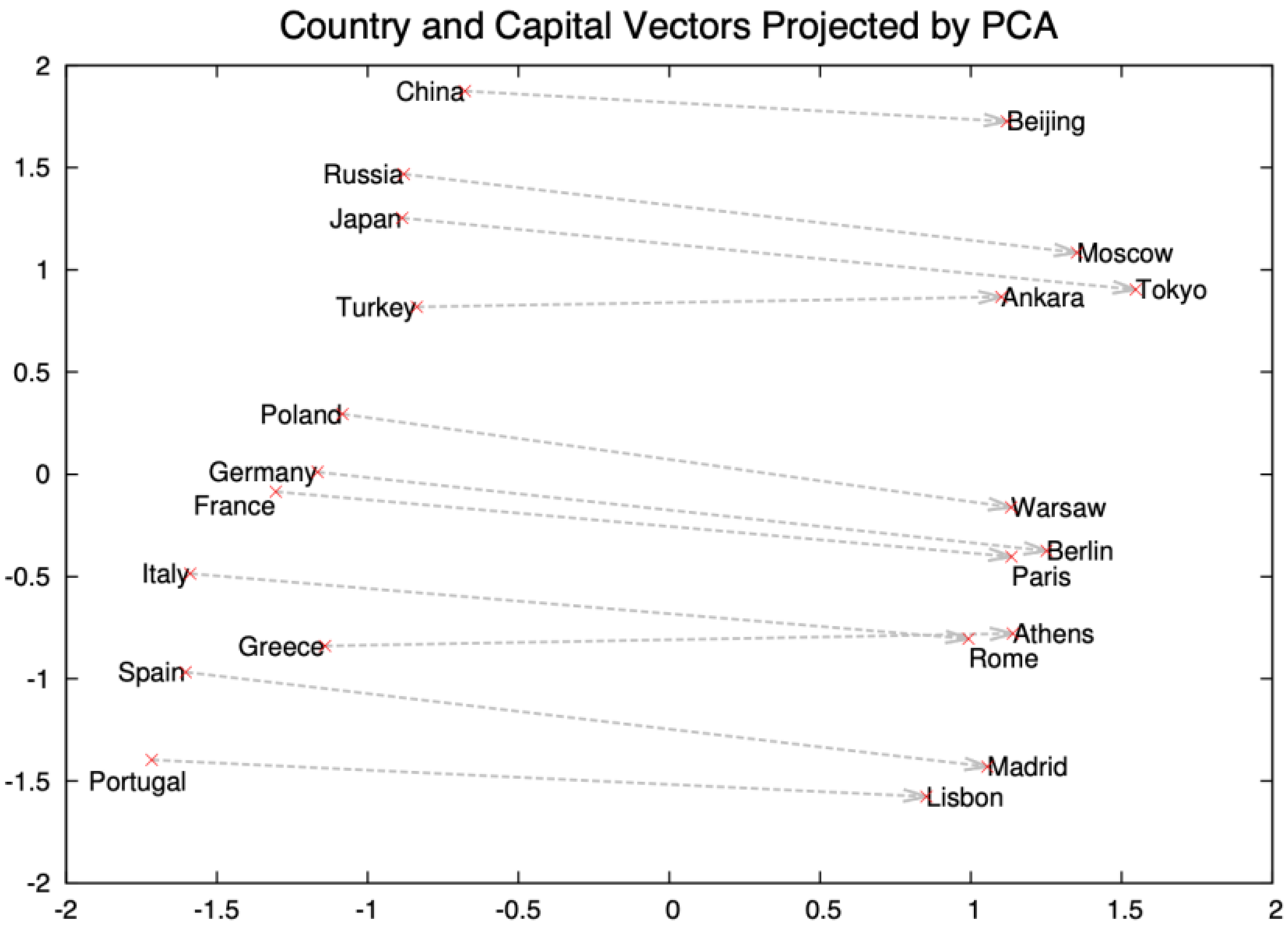
Word2vec Embeddings

- Regularities in the vector space correspond to regularities in language space!



$$w_{\text{man}} - w_{\text{woman}} \approx w_{\text{king}} - w_{\text{queen}}$$

$$w_{\text{apple}} - w_{\text{apples}} \approx w_{\text{car}} - w_{\text{cars}}$$



[Mikolov et al. 2013]

Word Embeddings You can Download

- word2vec [Mikolov et al. 2013]:
  https://code.google.com/archive/p/word2vec/

- GloVe [Pennington et al. 2014]:
  https://nlp.stanford.edu/projects/glove/

- fasttext [Bojanowsi et al. 2017]:
  https://fasttext.cc/

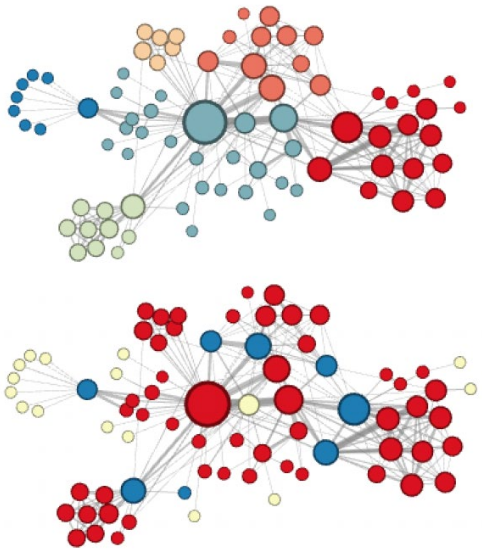
Extensions

- Neural word embeddings
  - Multilingual?
  - Social biases?
  - Contextualized?

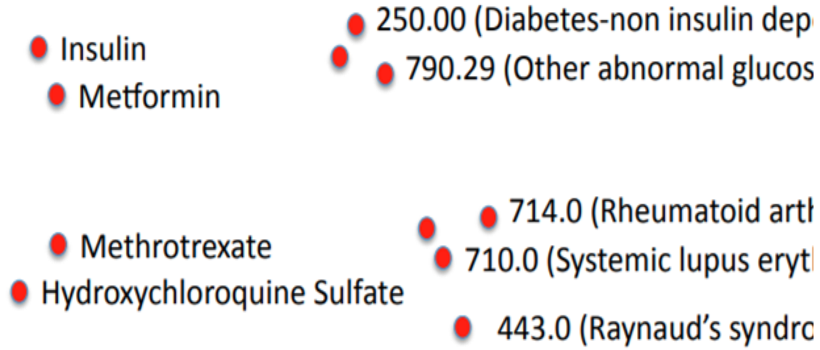
Extensions

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

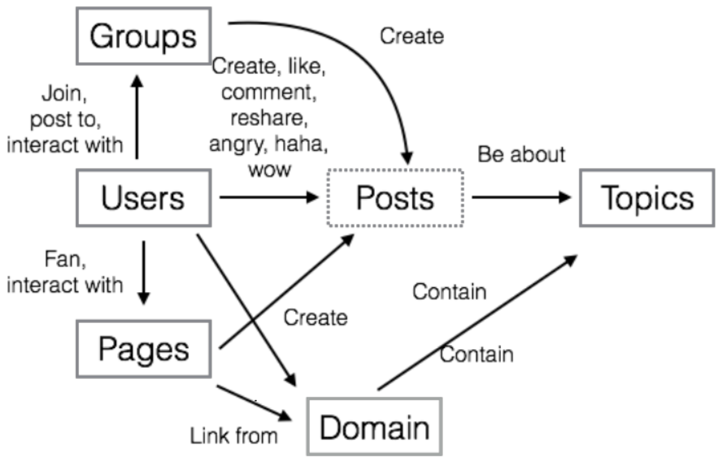
anything?



Node2Vec
[Grover and Leskovec 2016]



Concept2Vec
[Choi et al. 2016]



World2Vec
[Facebook AI Research]

Summary (1/2)

- Annotated Database for Lexical Semantics -- **WordNet**
- **Word vectors**: the use of a vector of numbers to represent a word
- **Distributional word vectors**: the use of distributional statistics (e.g., word co-occurrence counts) in defining word vectors
- Computing similarity of two vectors:
  - **dot product** is a simple starting point
  - **cosine similarity** accounts for vector length and works better for word vectors

Summary (2/2)

- Simple distributional word vectors: word-word co-occurrence counts

- Improving by reducing the influence of common context words
- **TF-IDF (Term Frequency – Inverse Document Frequency)**
- **PMI (Pointwise Mutual Information)**

- Learning representations from data: **word2vec**
  - **skipgram** (w/ negative sampling)
  - **CBOW (Continuous Bag-of-Words)**