Natural Language Processing

WBAI059-05

university of groningen

faculty of science and engineering

Tsegaye Misikir Tashu

Lecture 3: Vector Semantics & Embeddings

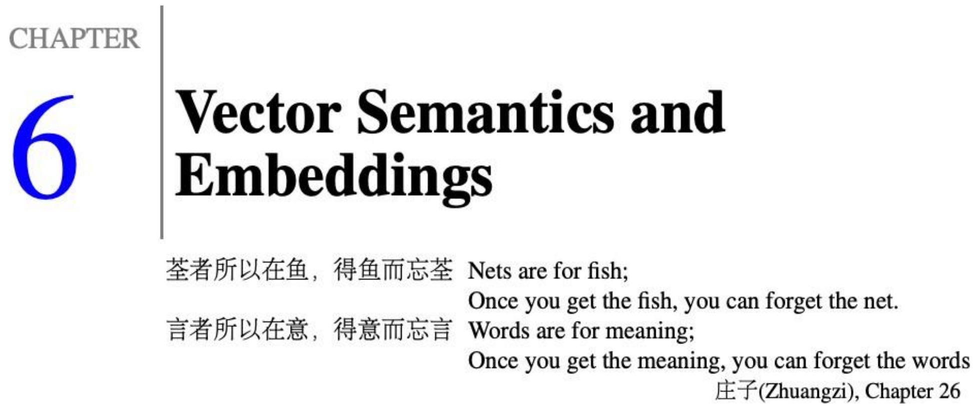
(Some slides are adapted from Dan Jurafsky and *Lena Voita*)

Lecture Plan

1. Word meaning
2. TF-IDF
3. Word2Vec Basics
4. Word2Vec training
5. Evaluating Word2Vec Embeddings

Recommended reading:

JM3 6.2-6.4, 6.6



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

Word Meaning: What do words mean?

What do words mean?

- **n-gram** and **text classification** methods often treat words as mere symbols, represented as strings or indices in a vocabulary.
- Introductory logic courses approach meaning differently:
  - They might define words using **predicate logic**, e.g.
    - The meaning of "dog" is DOG; cat is CAT

      $\forall x\ DOG(x) \longrightarrow MAMMAL(x)$ (All dogs are mammals)

  - This captures some **structure** but still lacks **nuance**, like polysemy (words with multiple meanings) and contextual shift

Lexical Semantics

- The branch of linguistics which is concerned with the systematic study of word meanings.

- The two most fundamental questions addressed by lexical semanticists are:
  a) How to describe the meanings of words, and
  b) How to account for the variability of meaning from context to context.

Lemmas and senses: Let’s start by looking at how one word might be defined in a dictionary.

lemma

mouse (N)

sense

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

- A “sense” or “concept” is the meaning component of a word
- Lemmas can be polysemous (have multiple senses)

Relations between senses: Synonymy

- Synonyms have the same meaning in some or all contexts.

couch / sofa	automobile / car
water / H_20	big / large

- Two words are synonymous if they are substitutable for one another in any sentence without changing the truth conditions of the sentence.

Relations between senses: Synonymy

- Note that there are probably no examples of perfect synonymy.
  - Even if many aspects of meaning are identical still may differ based on politeness, slang, register, genre, etc.

    - water/H_20

      "H_20" in a surfing guide?
    - big/large

      My big sister != my large sister
- In practice, the word synonym is therefore used to describe a relationship of approximate or rough synonymy.

Relation: **Similarity**

- Words with similar meanings. Not synonyms, but sharing some element of meaning.

```
car, bicycle
cow, horse
```

- Knowing how similar two words are can help in computing how similar the meaning of two phrases or sentences are.

Relation: Word relatedness

- The meaning of two words can be related in ways other than similarity.
- such class of connections is called word relatedness also traditionally called word association.

- Words can be related via a semantic frame or field.
  - `coffee, tea`: **Similar**
  - `coffee, cup`: **related**, not similar

Relatedness: Semantic field

- One common kind of relatedness between words is if they belong to the same semantic field.
  - A semantic field is a set of words that cover a particular semantic domain and bear structured relations with each other.
- **Hospitals**

  surgeon, scalpel, nurse, anaesthetic, hospital
- **Restaurants**

  waiter, menu, plate, food, chef
- **Houses**

  door, roof, kitchen, family, bed

Connotation (sentiment)

The aspects of a word's meaning that are related to a writer or reader's emotions, sentiments, opinions, or evaluations.

- Words have **affective** meanings
  - Positive connotations (*happy*)
  - Negative connotations (*sad*)
- Connotations can be subtle:
  - Positive connotation: *copy, replica, reproduction*
  - Negative connotation: *fake, knockoff, forgery*
- Evaluation (sentiment!)
  - Positive evaluation (*great*, *love*)
  - Negative evaluation (*terrible*, *hate*)

Vector Semantics & Embeddings

Vector Semantics

Computational models of word meaning

- Can we build a theory of how to represent word meaning, that accounts for at least some of the criteria?
- We'll introduce **vector semantics**
  - The standard model in language processing!
  - Handles many of our goals!

Computational models of word meaning

- Vector semantics is the standard way to represent word meaning in NLP.
  - The idea of vector semantics is to represent a word as a point in a multidimensional semantic space that is derived from the distributions of word neighbours.
  - Vectors for representing words are called embeddings.

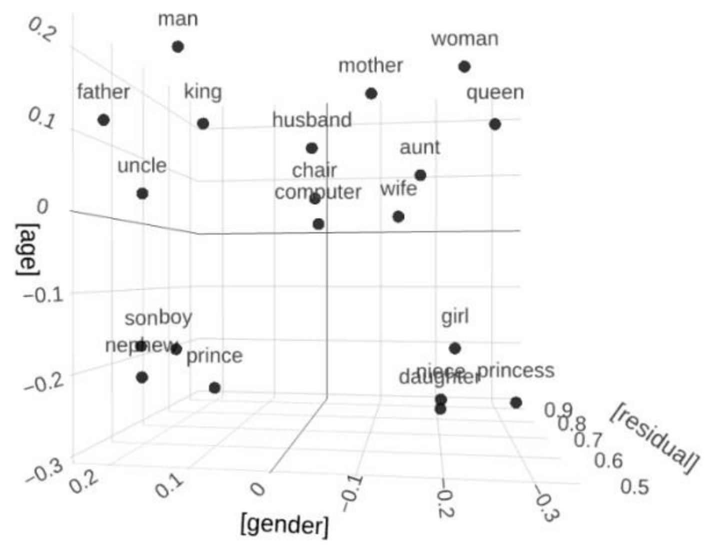
The big idea: model of meaning focusing on the similarity

Each word = a vector

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \qquad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Similar words are "**nearby in the vector space**"



(Bandyopadhyay et al. 2022)

Representing words by their context

- Distributional Hypothesis: A word’s meaning is given by the words that frequently appear close by.
    - “You shall know a word by the company it keeps” (J. R. Firth 1957: 11)
    - If A and B have almost identical environments, we say that they are synonyms.” [Harris 1954]
- When a word appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

What is the meaning of Tella?

Now look how this word is used in different contexts:

A bottle of Tella is on the table. Everyone likes Tella.

Tella makes you drunk.

We make Tella out of corn and barley.

$\Longrightarrow$ Tella is a kind of alcoholic beverage.

With context, you can understand the meaning!

The example is adopted from [Jacob Eisenstein's NLP notes]

What is meaning?

(1) A bottle of ______ is on the table.

(2) Everyone likes ______ .

(3) ______ makes you drunk.

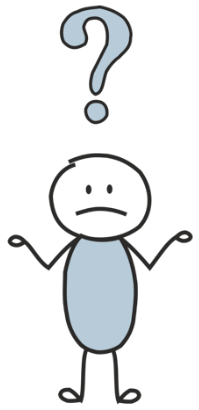
(4) We make ______ out of corn.

What other words fit into these contexts ?

	(1)	(2)	(3)	(4)	…
tella	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

← contexts

← rows show contextual properties: 1 if a word can appear in the context, 0 if not



What is meaning?

(1) A bottle of _______ is on the table.

(2) Everyone likes _______ .

(3) _______ makes you drunk.

(4) We make _______ out of corn.

	(1)	(2)	(3)	(4)	…
tella	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

<u>rows</u> are similar

What is meaning?

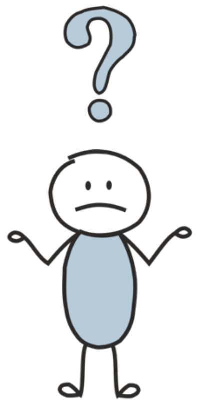
(1) A bottle of _______ is on the table.

(2) Everyone likes _______ .

(3) _______ makes you drunk.

(4) We make _______ out of corn.

	(1)	(2)	(3)	(4)	…
tella	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	



<u>rows</u> are similar $\Longrightarrow$ <u>meanings</u> of the words are similar

Is this true?

What is meaning?

(1) A bottle of ______ is on the table.

(2) Everyone likes ______ .

(3) ______ makes you drunk.

(4) We make ______ out of corn and barley.

	(1)	(2)	(3)	(4)	…
tella	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

This is the distributional hypothesis

rows are similar $\Rightarrow$ meanings of the words are similar

How can we do the same thing computationally?

Count the words in the context: TF-IDF

- A common baseline model
- **Sparse** vectors
- Words are represented by (a simple function of) the **counts** of nearby words

See what other words occur in those contexts: Word2vec

- **Dense** vectors
- Representation is created by training a classifier to **predict** whether a word is likely to appear nearby.
- Later we'll discuss extensions called **contextual embeddings**

Words and Vectors

Term-document matrix

- Each document is represented by a vector of words
- Each row represents a word in the vocabulary and each column represents a document.

	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

- Each column is a vector representing a document as a point in $|V|$-dimensional space

Vectors are the basis of NLP and document similarity

- Documents with similar content tend to share common words. Consequently, their corresponding column vectors in a feature space will also be similar.

	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

More common: word-word matrix (or "term-context matrix")

- Two **words** are similar in meaning if their context vectors are similar

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	aardvark	...	**computer**	**data**	**result**	**pie**	**sugar**	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Cosine for computing word similarity

- To measure similarity between two target words v and w, we need a metric that takes two vectors : Cosine similarity

- The cosine similarity metric between two vectors v and w thus can be computed as:

The dot product of the vectors

$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\boxed{\sum_{i=1}^{N} v_i w_i}}{\boxed{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}}$$

Product of their length

Cosine examples

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\cdot\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry},\text{information}) =$$

$$\frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

$$\cos(\text{digital},\text{information}) =$$

$$\frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

Raw frequency is a bad representation

- The co-occurrence matrices represent each cell by word frequencies.
  - Frequency is useful if sugar appears near the *apricot*, that is useful information.
  - But overly frequent words like *the*, *it,* or ‘*they* are not very informative about the context.
  - Raw frequency is very skewed and not very discriminative.
- This creates a paradox: how do we balance capturing important co-occurrences while minimizing the dominance of frequent but uninformative words?

- Solution 1: use a **weighted function** instead of raw counts!

TF-IDF

- Term frequency (tf): The number of times a word/term t occurs in the document d.

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

- Inverse document Frequence (idf): is the fraction of the total number of documents (N) in the collection, and the number of documents in which term *t* occurs.

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

- IDF is used to give a higher weight to words that occur only in a few documents.

The tf-idf weighted value for a term in a document:

$$TF\text{-}IDF = tf \; x \; idf$$

Raw counts:

	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:

	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

Sparse vs dense vectors

- The vectors in the word-document occurrence matrix are
  - Long: vocabulary size
  - Sparse: most are 0’s

- Alternative learn vectors which are
  - short (50-1000) dimensional
  - dense (most elements are non-zero) vectors.

Why dense vectors?

- Short vectors are easier to use as **features** in ML systems
- Dense vectors generalize better than explicit counts (points in real space vs points in integer space)

- Sparse vectors can not capture higher-order co-occurrence
  - w_1 co-occurs with “car”, w_2 co-occurs with “automobile”
  - They should be similar but they aren’t because “car” and “automobile” have distinct dimensions.

- In practice, they work better!

Word2Vec

Word2Vec

- Input: a large text corpora V, d
  - V: a pre-defined vocabulary
  - d: dimension of word vectors (e.g. 300)
  - Text corpora (words $w_1, \ldots, w_T$)
    - Wikipedia + Gigaword 5: 6B
    - Twitter: 27B
    - Common Crawl: 840B
- Output: $f : V \to \mathbb{R}^d$

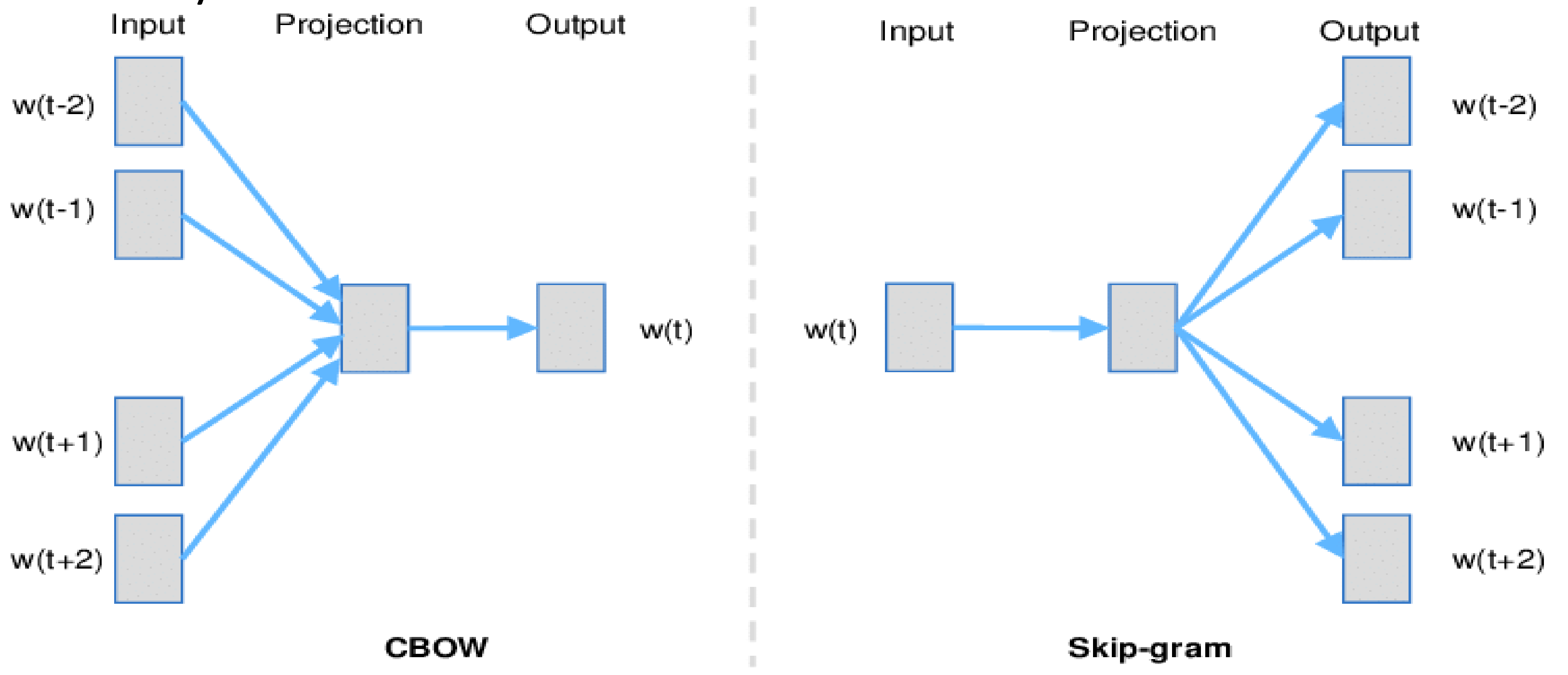
$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \qquad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Learn *f* by training classifiers to predict words and take learned weights as word vectors.

Word2Vec: a Prediction-Based Method

- (Mikolov et al 2013a): Efficient Estimation of Word Representations in Vector Space
- (Mikolov et al 2013b): Distributed Representations of Words and Phrases and their Compositionality



- Predict center word from (bag of) context words

- Predict context (”outside”) words (position independent) given center word

Word2vec: Overview

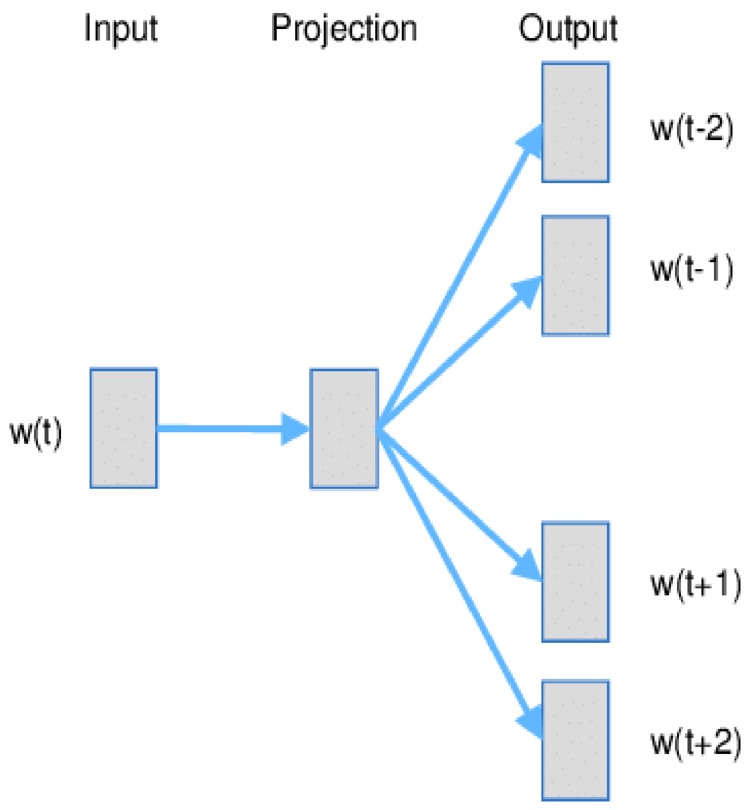
- The idea of Word2vec is to use each word to predict other words in its context.

- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector.

- Go through each position in the text, which has a centre (c) word and context ("outside") words (o)

- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)

- Keep adjusting the word vectors to maximize this probability.

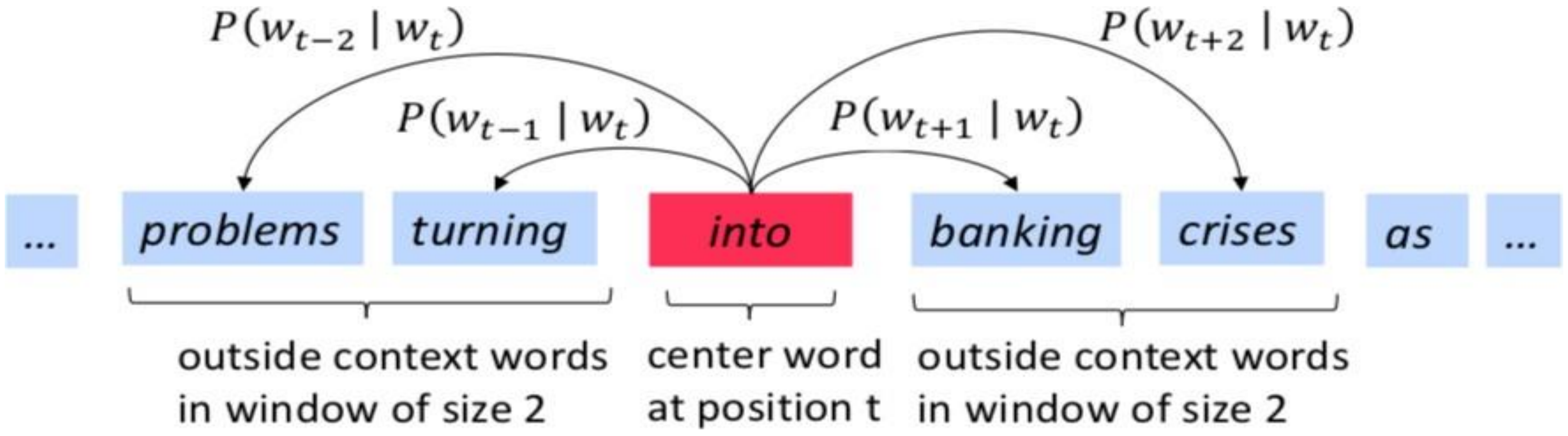


Skip-gram

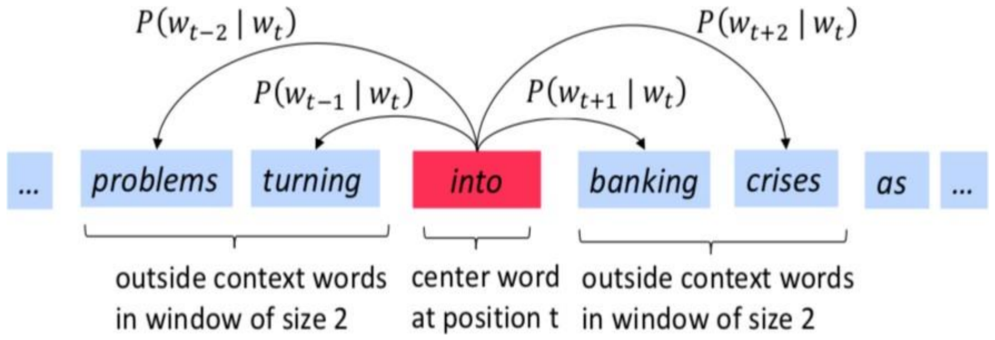
Skip-gram: The main idea of word2vec

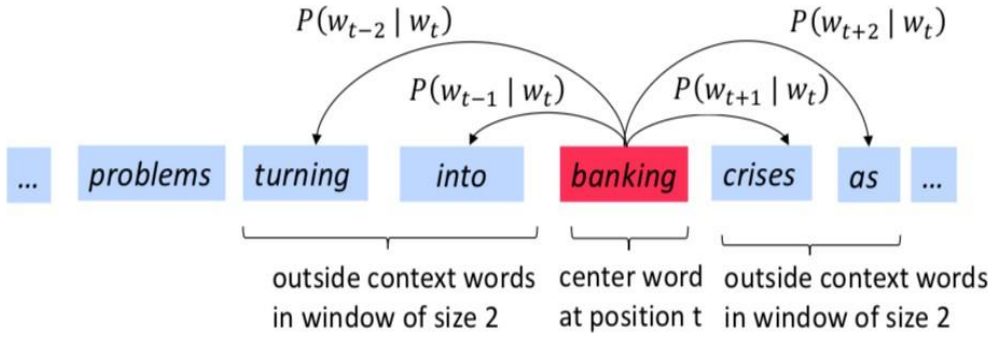
- Assume that we have a large corpus $w_1, w_2, \ldots, w_T \in V$
- **Key idea:** Use each word to predict other words in its context
- Context: a fixed window of size *2m* (*m = 2* in the example)

  P(b | a) = Given the centre word is ‘a’, what is the probability that ‘b’ is a context word?



Skip-gram : Example





Convert the training data into:
(into, problems)
(into, turning)
(into, banking)
(into, crises)
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)
…

Our goal is to find parameters that can maximize

$P(\text{problems} \mid \text{into}) \times P(\text{turning} \mid \text{into}) \times P(\text{banking} \mid \text{into}) \times P(\text{crises} \mid \text{into})$ X

$P(\text{turning} \mid \text{banking}) \times P(\text{into} \mid \text{banking}) \times P(\text{crises} \mid \text{banking}) \times P(\text{as} \mid \text{banking}$

Skip-gram: Objective function

For each position *t = 1, 2,..., T* in a text corpus, predict context words within a window of fixed size m, given centre word w_t.

Likelihood = $L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

go over text

with a sliding window

compute probability of the context word given the central

Minimizing objective function ⟺ Maximizing predictive accuracy

How to calculate $P\ (w_{t\ +\ j}|w_t\ ;\theta)$?

- We have two sets of vectors for each word in the vocabulary
  - u_t vector when it is a centre word
  - v_c vector when it is a context word
- Use the inner product between u_t and v_c to measure how likely word *t* appears with context word *c*

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

Where

- $P(w_{t+j}|w_t)$ is the probability of a context word w_{t+j} given the target word w_t w
- u_{wt} is the **word vector** for the target word w_t
- v_{wt+j} is the **word vector** for the context word w_{t+j}.
- v_k represents the word vectors for all words in the vocabulary V.

Word2vec: Prediction function

Exponentiation makes anything positive

Do product compare the similarity of u and v

$$P(w_{t+j} \mid w_t) = \frac{\exp\left(\boxed{\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}}}\right)}{\boxed{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}}$$

Normalize over entire vocabulary to give probability distribution

- This is an example of the softmax function
- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies the probability of the largest xi
  - "soft" because still assigns some probability to smaller xi

Important note here is that

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

- The goal is to **maximize** the probability of the correct context words given a target word.
- Since we use **log probabilities**, the model **minimizes** the negative log likelihood.
- The equation essentially means: **if two words appear in similar contexts, their word vectors should be similar**.
- When trained on a large corpus, this function helps Word2Vec learn **meaningful word representations**, following the **distributional hypothesis** (i.e., words appearing in similar contexts have similar meanings).

Which Embedding do we use?

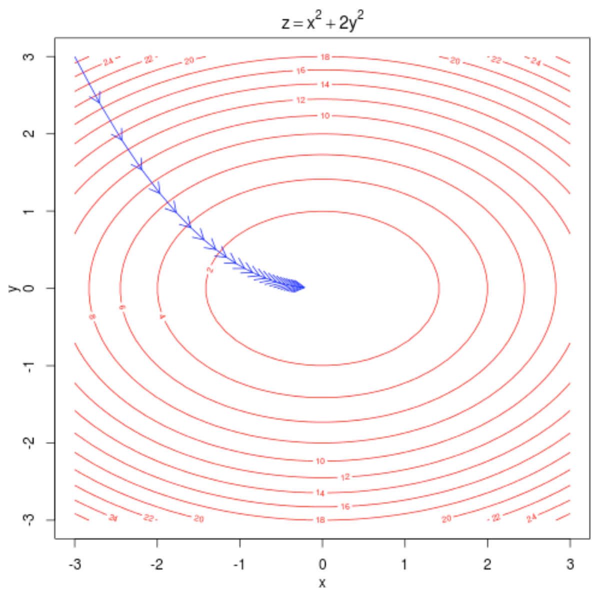
- We learn the **input embeddings** W for the target word w_t.
- We also learn the **output embeddings** W’ for the context word w_{t+j}
- After training, we typically use the **input embeddings** from *W* as the final word representations.

To train the model: Optimize value of parameters to minimize loss

To train a model, we gradually adjust parameters to minimize a loss or increase the probability.

- θ represents **all** the model parameters, in one long vector
- In our case, with *d*-dimensional vectors and *V*-many words, we have
- Remember: every word has two vectors

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



- We optimize these parameters by walking down the gradient.
- We compute all vector gradients!

How to train: by Gradient Descent, One Word at a Time: One training step in detail

Loss for word t in window j

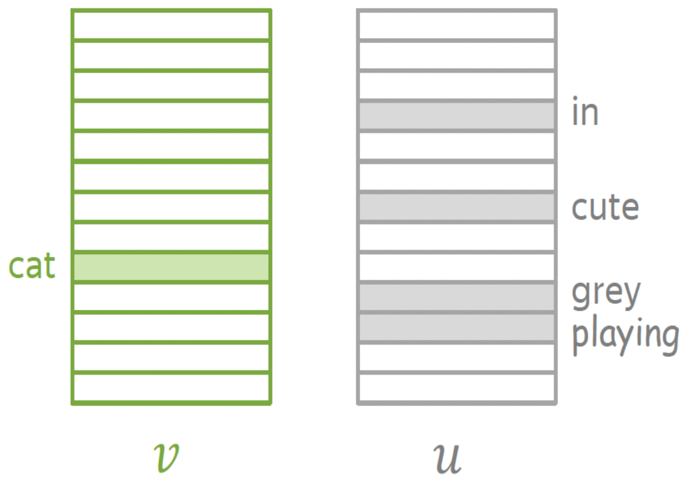
$$\text{Loss} = J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m,\\ j\ne 0}} \log P\left(w_{t+j}\middle|w_t,\theta\right) = -\frac{1}{T}\sum_{t=1}^{T}\underbrace{\sum_{\substack{-m\le j\le m,\\ j\ne 0}} J_{t,j}(\theta)}$$

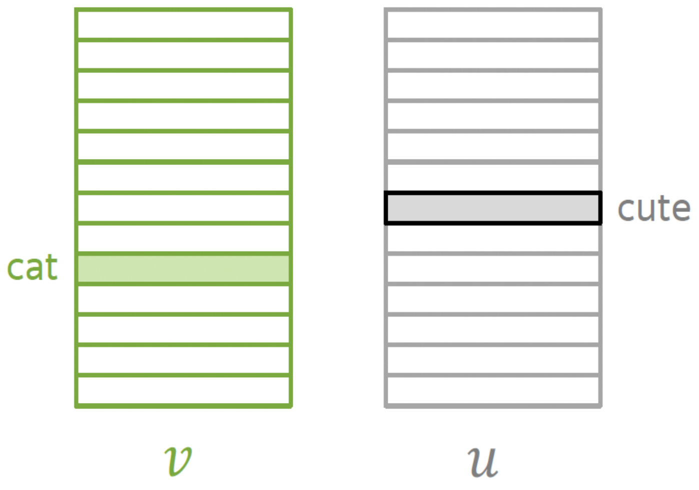
pick one window

... I saw a cute grey cat playing in the garden ...

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

Pick on word

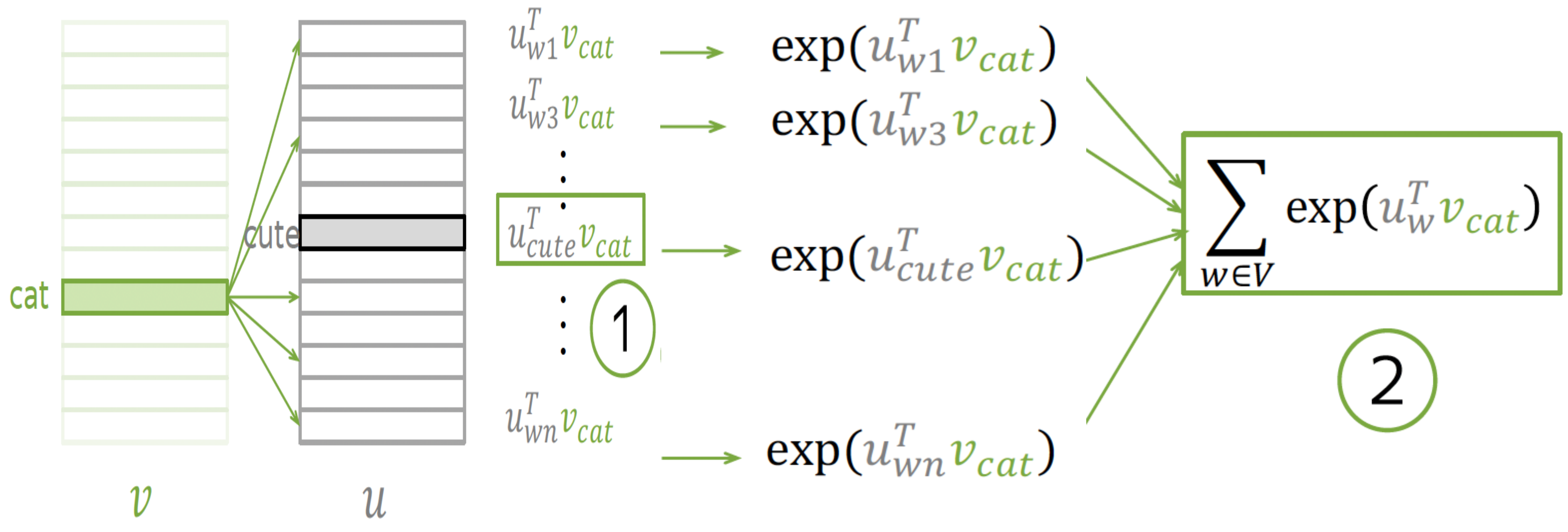




Look at the loss component for this step:

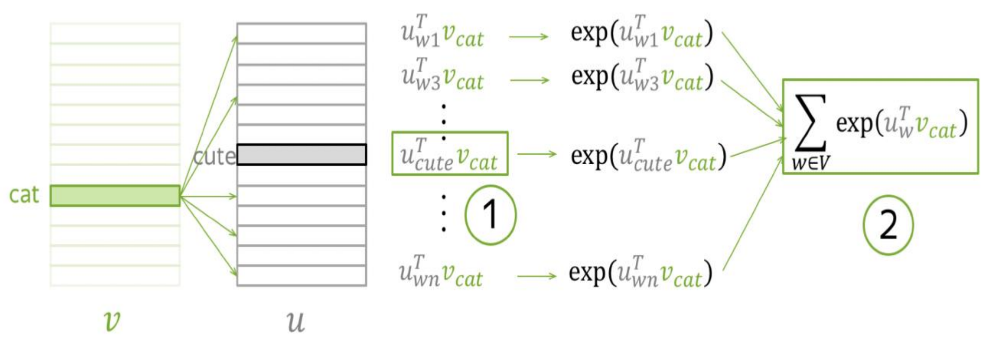
$$-\log P(cute|cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{w\in V}\exp(u_w^T v_{cat})} = -u_{cute}^T v_{cat} + \log \sum_{w\in V} \exp(u_w^T v_{cat})$$

1. Take dot product of v_{cat} with all u 2. exp 3. sum all



1. Take dot product of v_{cat} with **all** u 2. exp   3. sum all

$$-\log P(cute|cat)$$
$$= -u_{cute}^T v_{cat} + \log \sum_{w \in V} \exp(u_w^T v_{cat})$$



4. get loss (for this one step)

$$J_{t,j}(\theta) = \underbrace{-u_{cute}^T v_{cat}}_{1} + \underbrace{\log \sum_{w \in V} \exp(u_w^T v_{cat})}_{2}$$

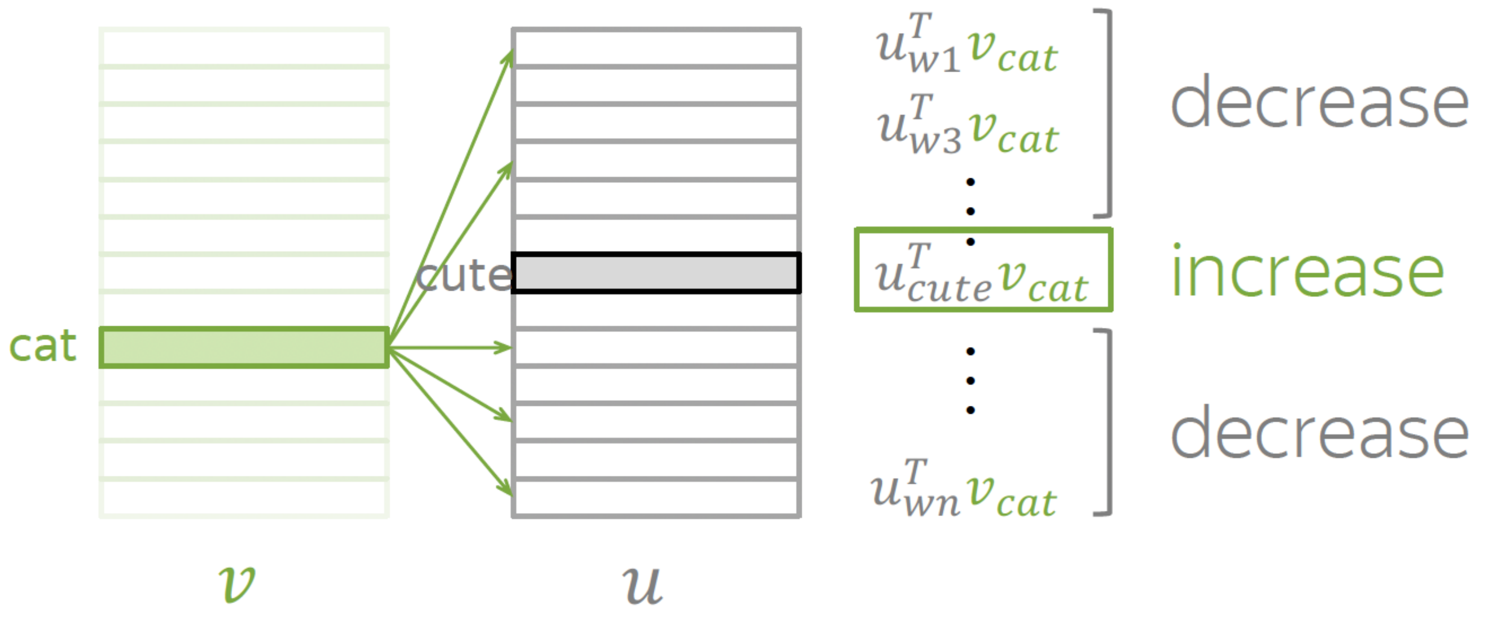
5. evaluate the gradient, make an update

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \ \forall\, w \in V$$

One Update Intuition

$$-\log P(cute|cat) = -u_{cute}^T v_{cat} + \log \sum_{w \in V} \exp(u_w^T v_{cat})$$

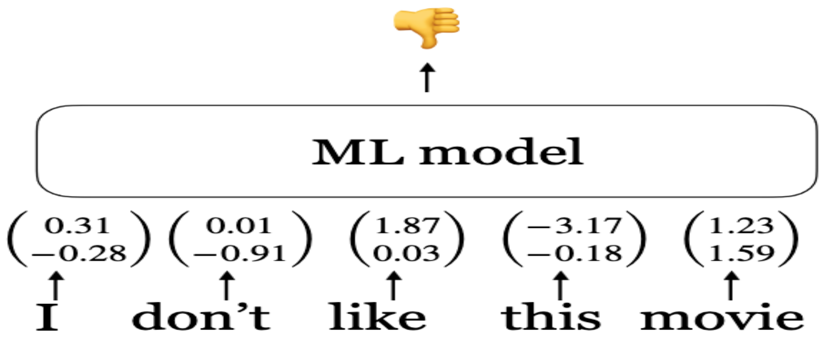


Evaluating word embeddings

Extrinsic vs intrinsic evaluation

Extrinsic

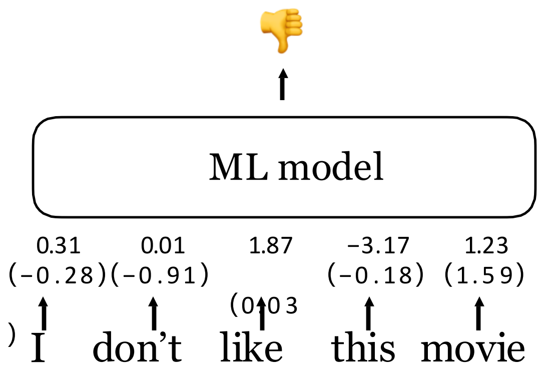
- Plugging in the word embeddings into a real NLP system and see whether this improves performance.
- Could take a long time but still the most important evaluation metric



Intrinsic

- Evaluate on a specific/intermediate subtask- **word similarity**
- Fast to compute

Extrinsic evaluation



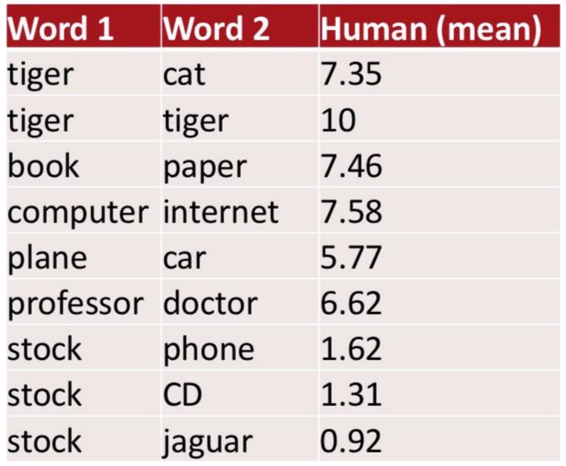
- A straightforward solution: given an input sentence $x_1, x_2, \ldots, x_n$
- Instead of using a bag-of-words model, we can compute $\text{vec}(x) = \mathbf{e}(x_1) + \mathbf{e}(x_2) + \ldots + \mathbf{e}(x_n)$
- And then train a logistic regression classifier on $vec(x)$

Intrinsic evaluation: word similarity

Word similarity

Example dataset: wordsim-353
353 pairs of words with human judgement

http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/



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

Cosine similarity:

$$\cos(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_j}{||\boldsymbol{u}_i||_2 \times ||\boldsymbol{u}_j||_2}.$$

Looking at word vectors

https://projector.tensorflow.org/

Questions