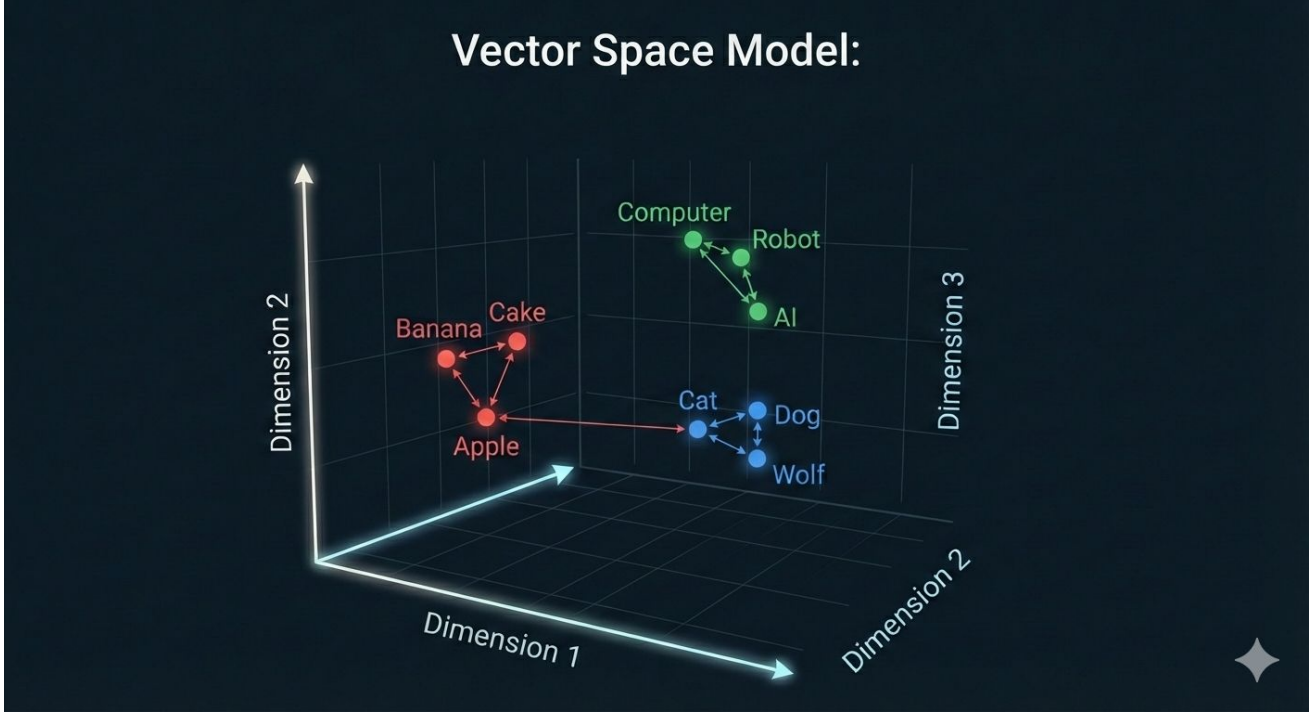


Natural Language Understanding

CSCI 4907/6515

Lecture 4: February 3, 2026

Aya Zirikly

So far, we've represented words as strings of characters and counted words

	amazing	terrible	happy	bike	…
Doc 1	2	4	1	1	…
Doc 2	7	1	4	2	…
Doc 3	1	1	1	6	…
…	…	…	…	…	…

how many times did this word appear?

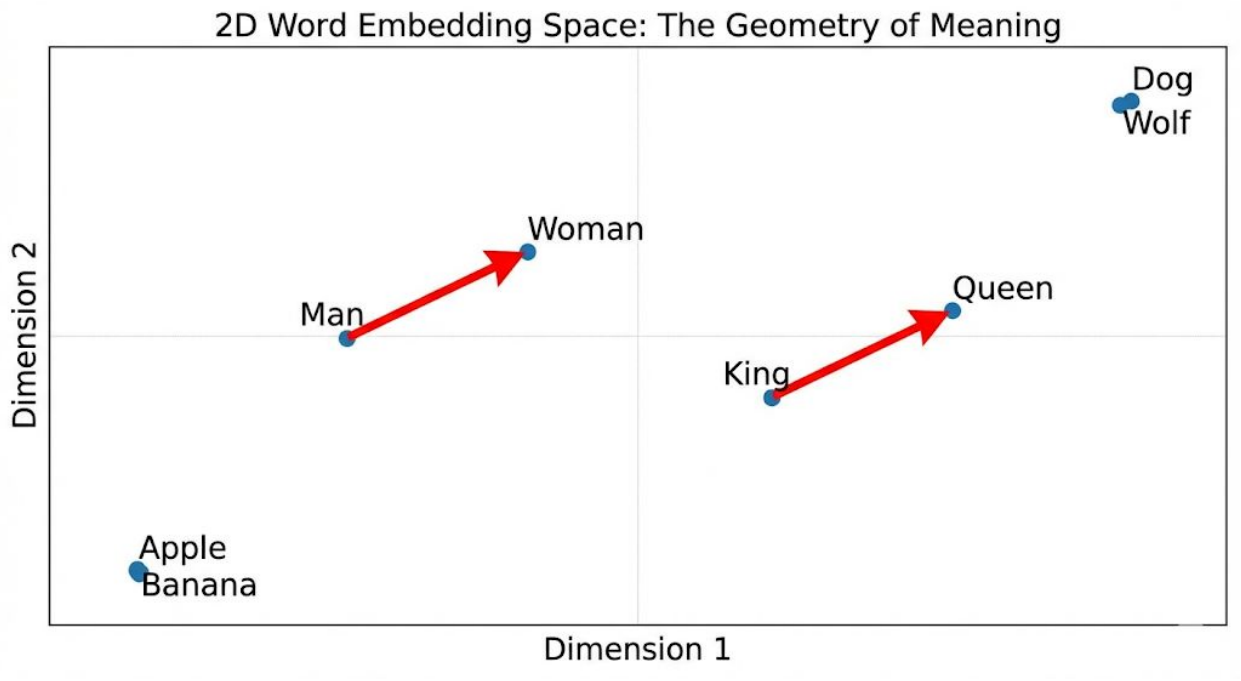
how likely is this word to appear in this context?

From counting to connecting

The Problem with BoW: In a count-based vector, the sentences "The dog chased the cat" and "A canine pursued the feline" have zero overlap. To a computer, they are as different as apples and asteroids.

The Vector Space Model (VSM) Solution: We map words into a multi-dimensional "semantic space."

If we give words coordinates, we can do Word Algebra.



Context

Concept

So far, we've represented words as strings of characters and counted words

Ideally, we actually want to know something about the meaning of the words

	amazing	terrible	happy	bike	…
Doc 1	2	4	1	1	…
Doc 2	7	1	4	2	…
Doc 3	1	1	1	6	…
…	…	…	…	…	…

So far, we've represented words as strings of characters and counted words

Ideally, we actually want to know something about the meaning of the words

- For example, it would be nice to know that if the word "amazing" is a word that is likely to occur in a positive document, then so is "awesome"

	amazing	terrible	happy	bike	…
Doc 1	2	4	1	1	…
Doc 2	7	1	4	2	…
Doc 3	1	1	1	6	…
…	…	…	…	…	…

Word similarity for question answering

"**fast**" is similar to "**rapid**"

"**tall**" is similar to "**height**"

Question answering:

*Q: "How **tall** is Mt. Everest?"*
*Candidate A: "The official **height** of Mount Everest is 29029 feet"*

Word similarity for plagiarism detection

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as Ebay, Amazon, and computing-giant

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc.

Positive or negative sentiment?

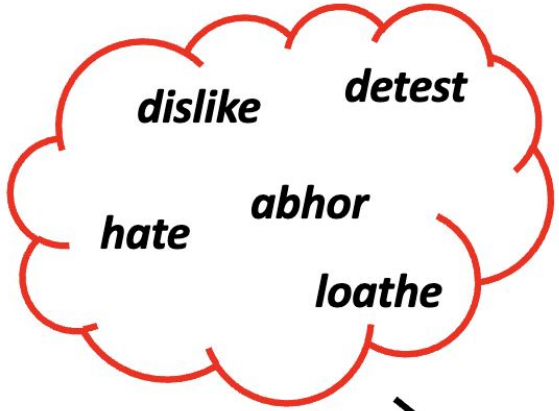
I **hated** this movie

I **detest** this movie

Positive or negative sentiment?

I **hated** this movie

I **detest** this movie



Vector space models

- How do we construct meanings from scratch based on text data?
- And more specifically, given two words, how can we determine if they are similar in meaning?

Wishlist for representations of word meaning

- **Word senses**: each “lemma” can have multiple word meanings
- **Synonymy**: two words are synonymous if they are substitutable for one another in any sentence without changing the situations in which the sentence is true
  - **Principle of contrast:** There are no total synonyms (e.g., H_2O vs. water)
- **Word similarity:** cat - dog
- **Word relatedness:** e.g., coffee - cup; scalpel - surgeon
- **Semantic fields** (college)

Wishlist for representations of word meaning con’d

- **Semantic frames and roles:** set of words that denote perspectives/participants in an event (e.g., buy-sell)
- **Lexical entailments, such as factivity**
  - Kim knows it is hot outside.
  - Kim thinks it is hot outside.
- **Register/formality:** what does the usage of this word say about the current situation?
- **Index of social address:** what does the usage of this word say about the speaker?
- **Anything else?**

Principle of contrast

"Every two forms contrast in meaning."

True synonyms do not exist. If a language keeps two different words (e.g., *"Big"* and *"Large"*), there must be a reason. They might share 90% of their meaning, but there is always a nuance (a "contrast") that separates them.

- *Example:* You can say *"My big sister"* (meaning older), but you cannot say *"My large sister"* (unless you are commenting on her weight).

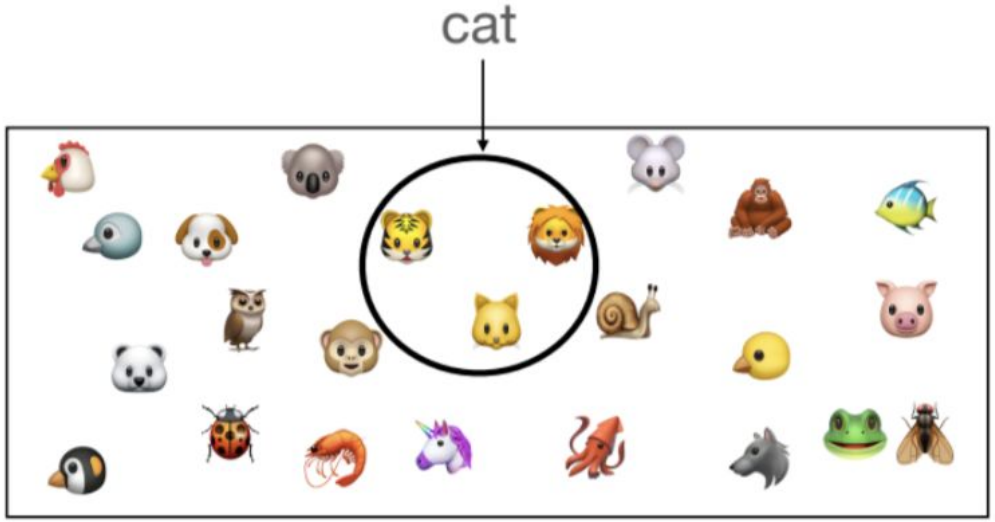
Why this matters in VSM

If we only had 10 dimensions, the vectors for *"Big"* and *"Large"* might end up identical v_big ≈ v_large.
The **Principle of Contrast** tells us this is a failure state.
A good VSM must have enough capacity to capture the 10% of contexts where these words *cannot* be swapped.

We need high-dimensional vectors (300+ dimensions) rather than just a few.

Theories of word meaning

- A word refers to sets



Theories of word meaning

- A word refers to sets
- A word refers to a prototype/exemplars

cat

cat = weighted combination([whiskers, ears, tail,..., fur, purr, eats food, needs oxygen])

Theories of word meaning

- A word refers to sets
- A word refers to a prototype/exemplars

In NLP/NLU:

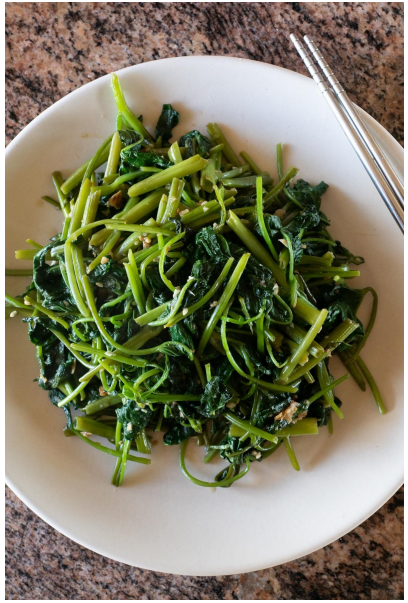
- **Distributional hypothesis:** Words are defined by the contexts in which they are used
  - “I know a word by the company it keeps”

“I know a word by the company it keeps”

(6.1) Ongchoi is delicious sauteed with garlic.
(6.2) Ongchoi is superb over rice.
(6.3) ...ongchoi leaves with salty sauces...

(6.4) ...spinach sauteed with garlic over rice...
(6.5) ...chard stems and leaves are delicious...
(6.6) ...collard greens and other salty leafy greens

“I know a word by the company it keeps”

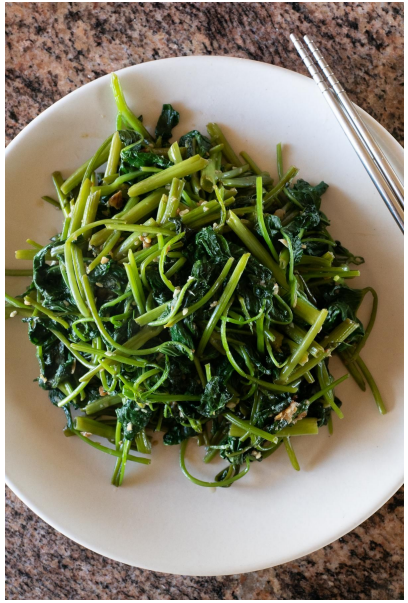


(6.1) Ongchoi is delicious sauteed with garlic.
(6.2) Ongchoi is superb over rice.
(6.3) ...ongchoi leaves with salty sauces...

(6.4) ...spinach sauteed with garlic over rice...
(6.5) ...chard stems and leaves are delicious...
(6.6) ...collard greens and other salty leafy gree

“I know a word by the company it keeps”

Ongchoi ~Spinach/Chard



(6.1) Ongchoi is delicious sauteed with garlic.

(6.2) Ongchoi is superb over rice.

(6.3) ...ongchoi leaves with salty sauces...

(6.4) ...spinach sauteed with garlic over rice...

(6.5) ...chard stems and leaves are delicious...

(6.6) ...collard greens and other salty leafy gree

Standard approach to representing word meaning in NLP/NLU: Vector Space Models

- We will model words as vectors (i.e., as lists of numbers / points in space)
- Ideally, words that have similar meanings will be nearby in space
- How do we create these vectors? Today, we will consider two ways:
  - Term-document matrices
  - Term-term matrices

Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

Term-document matrices

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Semantically related

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

If we treat the columns as vectors, the angle between these two plays is small.
→ These two plays belong to the same genre (History/Tragedy) without reading them.

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

This word is useless for distinguishing meaning

Term-document matrices

Cells can contain counts, binary present/not present values, etc.

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Similar documents contain similar words
Similar words occur in similar documents

Word meaning = the set of documents a word occurs in

Term-term matrix

Term-term matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of $\pm$ 4 words

Term-term matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of $\pm$ 4 words
- A word is now defined by a vector over counts of context words

Term-term matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of $\pm$ 4 words
- A word is now defined by a vector over counts of context words
  - Instead of each vector being of length D
- Each vector is now of length |V|

Term-term matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of $\pm$ 4 words
- A word is now defined by a vector over counts of context words
  - Instead of each vector being of length D
- Each vector is now of length |V|
- The word-word matrix is |V|x|V|

Term-term matrix

The domestic cat is a small, typically furry, carnivorous mammal.

Your cat's online owners manual, featuring articles about breed information, cat selection, training, grooming and care for cats and kittens.

Wish you had a secret decoder guide to cat behavior and cat language? Here's a primer to things your cat wishes you understood.

"The cat does not offer services," William Burroughs wrote. "The cat offers itself." But it does so with unapologetic ambivalence.

Welcome to the new WebMD Cat Health Center. WebMD veterinary experts provide comprehensive information about cat health care, offer nutrition and feeding …

Yes, they're independent and willful, but felines can be taught certain behaviors—to the benefit of both cat and human.

Term-term matrix

The domestic **cat** is a small, typically furry, carnivorous mammal.

Your **cat's** online owners manual, featuring articles about breed information, **cat** selection, training, grooming and care for **cats** and kittens.

Wish you had a secret decoder guide to **cat** behavior and **cat** language? Here's a primer to things your **cat** wishes you understood.

"The **cat** does not offer services," William Burroughs wrote. "The **cat** offers itself." But it does so with unapologetic ambivalence.

Welcome to the new WebMD **Cat** Health Center. WebMD veterinary experts provide comprehensive information about **cat** health care, offer nutrition and feeding …

Yes, they're independent and willful, but felines can be taught certain behaviors—to the benefit of both **cat** and human.

Term-term matrices

The domestic **cat** is a small, typically furry, carnivorous mammal.

Your **cat's** online owners manual, featuring articles about breed information, **cat** selection, training, grooming and care for **cats** and kittens.

Wish you had a secret decoder guide to **cat** behavior and **cat** language? Here's a primer to things your **cat** wishes you understood.

"The **cat** does not offer services," William Burroughs wrote. "The **cat** offers itself." But it does so with unapologetic ambivalence.

Welcome to the new WebMD **Cat** Health Center. WebMD veterinary experts provide comprehensive information about **cat** health care, offer nutrition and feeding …

Yes, they're independent and willful, but felines can be taught certain behaviors—to the benefit of both **cat** and human.

Term-term matrix

The domestic **cat** is a small, typically furry, carnivorous mammal.

Your **cat's** online owners manual, featuring articles about breed information, **cat** selection, training, grooming and care for **cats** and kittens.

Wish you had a secret decoder guide to **cat** behavior and **cat** language? Here's a primer to things your **cat** wishes you understood.

"The **cat** does not offer services," William Burroughs wrote. "The **cat** offers itself." But it does so with unapologetic ambivalence.

Welcome to the new WebMD **Cat** Health Center. WebMD veterinary experts provide comprehensive information about **cat** health care, offer nutrition and feeding …

Yes, they're independent and willful, but felines can be taught certain behaviors—to the benefit of both **cat** and human.

	the	domes-tic	is	a	your	online	owners	breed	informa-tion	selec-tion
cat	1000	40	500	700	400	3	80	100	15	6

Term-term matrix

The domestic **cat** is a small, typically furry, carnivorous mammal.

Your **cat's** online owners manual, featuring articles about breed information, **cat** selection, training, grooming and care for **cats** and kittens.

Wish you had a secret decoder guide to **cat** behavior and **cat** language? Here's a primer to things your **cat** wishes you understood.

"The **cat** does not offer services," William Burroughs wrote. "The **cat** offers itself." But it does so with unapologetic ambivalence.

Welcome to the new WebMD **Cat** Health Center. WebMD veterinary experts provide comprehensive information about **cat** health care, offer nutrition and feeding …

Yes, they're independent and willful, but felines can be taught certain behaviors—to the benefit of both **cat** and human.

	the	domes-tic	is	a	your	online	owners	breed	informa-tion	selec-tion
cat	1000	40	500	700	400	3	80	100	15	6

Word meaning = the set of words this word co-occurs with

Term-term matrix

	cat	kitten	cute	adorable	gradients
cat	0	0	1	1	0
kitten	0	0	1	1	0
cute	1	1	0	0	0
adorable	1	1	1	0	0
gradients	0	0	1	1	1

Similar words don’t necessarily co-occur

Term-term matrix

	cat	kitten	cute	adorable	gradients
cat	0	0	1	1	0
kitten	0	0	1	1	0
cute	1	1	0	0	0
adorable	1	1	1	0	0
gradients	0	0	1	1	1

…but they occur in the same contexts

Term-term matrix

Question: How to pick the size of the context window?

Term-term matrix

Question: How to pick the size of the context window?

The size of the window depends on the representation goals

- The shorter the window the more syntactic the representation (+/- 1-3)
- The larger the window the more semantic the representation (+/- 4-10)

Question: When to use term-term matrix vs. term-document matrix?

- What differences do you expect to emerge between word vectors constructed using term-term vs. term-document matrices?
- Term document:
  - More broad, topical similarity
  - Good for document classification tasks, retrieval
- Term term:
  - More grammatical similarity (nouns, verbs)
  - More lexical similarity (synonyms)

Term-document vs. Term-term matrices

Assume I have a term-document matrix built using New York Times articles. Which of the following pairs do you think will be most similar according to cosine similarity?

A. w1 = gas, w2 = economy

B. w1 = gas, w2 = petrol

C. w1 = gas, w2 = fuel

Questions?

Vector Space Models

Vector space models: Words are represented as vectors (points in space)

Questions:

- What properties of the word does this capture?
- What properties of the word does it miss?

Problem with raw counts

- Some words happen in basically every document, e.g., “the”, “of”, “good”, etc.
- These words won’t tell us much about the meaning of any particular word
- We’d rather have a measure that takes into account whether appearing in a document or near a context word is **particularly informative**
- Two approaches:
  - TF-IDF (term frequency-inverse document frequency) weighting
  - Pointwise mutual information (PMI)

TF-IDF weighting (term-document)

TF-IDF weighting (term-document)

Term Frequency - Inverse document frequency

The value in each cell is calculated as: $w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
the					
good					
happy					
cat					
gradient					

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

$\text{idf}_t = N / \text{df}_t$

E.g., how many times does ‘the’ occur in Doc 1

N = total num of docs; df_{the} = how many docs does “the” occur in

Example (TF-IDF)

Example (TF-IDF)

- Calculate the TF-IDF value for (1) ‘the’ and (2) ‘gradient’ for Document 1.
- What effect does this have? Why is this beneficial?

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
the	10	17	5	4	20
good					
happy					
cat					
gradient	10	0	0	0	0

Example (TF-IDF)

- Calculate the TF-IDF value for (1) ‘the’ and (2) ‘gradient’ for Document 1.
- What effect does this have? Why is this beneficial?

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
the	10 x 1	17	5	4	20
good					
happy					
cat					
gradient	10 x 5	0	0	0	0

Example (TF-IDF)

- Calculate the TF-IDF value for (1) ‘the’ and (2) ‘gradient’ for Document 1.
- What effect does this have? Why is this beneficial?

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
the	10 x 1	17	5	4	20
good					
happy					
cat					
gradient	10 x 5	0	0	0	0

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

$\mathrm{idf}_t = N / \mathrm{df}_t$

Question: Does length of document affect the tf-idf matrix?

Question: Does length of document affect the tf-idf matrix?

Question: Does length of document affect the tf-idf matrix?

Longer Documents will unfairly dominate the retrieval results simply because they have more words (and thus higher term frequencies for everything, and this can skew the importance of certain terms.

Imagine you are searching for the word **"galaxy"**.

- **Document A (Short):** A 50-word abstract about galaxies. "Galaxy" appears **2 times**.
- **Document B (Long):** A 500-page sci-fi novel. "Galaxy" appears **50 times**.

In raw TF-IDF:

- Document A has a raw TF of 2.
- Document B has a raw Term Frequency (TF) of 50.
- **Result:** The model thinks the novel is ***more relevant***, even though the abstract is arguably more focused on the topic. The novel just "talked more."

Imagine you are searching for the word **"galaxy"**.

- **Document A (Short):** A 50-word abstract about galaxies. "Galaxy" appears **2 times**.
- **Document B (Long):** A 500-page sci-fi novel. "Galaxy" appears **50 times**.

Document Length Normalization

In raw TF-IDF:

- Document A has a raw TF of 2.
- Document B has a raw Term Frequency (TF) of 50.
- **Result:** The model thinks the novel is ***more relevant***, even though the abstract is arguably more focused on the topic. The novel just "talked more."

Frequency normalization

$$TF(t, d) = \frac{\text{count}(t, d)}{\text{total words in } d}$$

This turns the count into a probability/percentage.

- Doc A: 2 / 50 = 0.04 (4% density)
- Doc B: 50 / 100,000 = 0.0005 (0.05% density)

Log normalization

Raw term frequency (**tf**) is misleading because relevance does not scale linearly.
$\mathbf{w_{□,d} = 1 + \log_{10}(tf_{□,d})}$ *(if tf > 0)* $\mathbf{w_{□,d} = 0}$ *(if tf = 0)*

Word Count (tf)	Linear Weight (Raw)	Log Weight (1 + $\log_{10}$)	Impact
0	0	0	None
1	1	1	Baseline
10	10	2	**2x** more important than 1
100	100	3	**3x** more important than 1
1,000	1,000	4	**4x** more important than 1

Log normalization

We calculate a weight based on the total number of documents (N) and the number of documents containing the term ($\mathbf{df}_t$):

$$\mathbf{idf}_t = \log_{10}(N / \mathbf{df}_t)$$

- N = Total number of documents in the collection
- $\mathbf{df}_t$ = Document Frequency (how many docs have word t)

Word	Document Frequency ($\mathbf{df}_t$)	Calculation (N / $\mathbf{df}_t$)	IDF Weight	Meaning
"the"	1,000,000	1	**0**	**Ignored** (Stop word)
"market"	10,000	100	**2**	**Common** (Topic word)
"bioluminescent"	10	100,000	**5**	**Critical** (Specific Keyword)

Cosine normalization

If we compare a tweet about "NASA" to a textbook about "NASA", the textbook's vector is huge (magnitude) simply because it has more words.

- We want to compare the **proportion** of words, not the total count.
- We shrink (or stretch) every vector so it has a length of exactly **1**. This projects all documents onto the "Unit Hypersphere."

Euclidean Norm calculate the length (or magnitude) of the vector $\mathbf{v}$:

$$||\mathbf{v}||_2 = \sqrt{\left(\Sigma\ v_i^2 \right)}$$

Then, we divide every element in the vector by this length:

$$\mathbf{v}\_\text{normalized} = \mathbf{v} \,/\, ||\mathbf{v}||_2$$

Example

Imagine two documents with the words **["sun", "moon"]**:

- **Doc A (Short):** [1, 1]
- **Doc B (Long):** [10, 10] *(Same content ratio, just longer)*

Step 1: Calculate Lengths

- **Length A:** $\sqrt{(1^2 + 1^2)} = \sqrt{2} \approx$ **1.41**
- **Length B:** $\sqrt{(10^2 + 10^2)} = \sqrt{200} \approx$ **14.1**

Step 2: Normalize

- **Doc A:** [1/1.41, 1/1.41] ≈ **[0.707, 0.707]**
- **Doc B:** [10/14.1, 10/14.1] ≈ **[0.707, 0.707]**

Both topics have the same **topic distribution**, regardless of how long they are

Positive Pointwise Mutual Information (PPMI)

- Used for term-term matrices
- High-level idea: How much more do two words co-occur than we would expect by chance
- Pointwise Mutual Information:

Positive Pointwise Mutual Information (PPMI)

- Used for term-term matrices
- High-level idea: How much more do two words co-occur than we would expect by chance
- Pointwise Mutual Information

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Pointwise mutual information (**PMI**) is used to measure to what extent two words, w_1 and w_2, are more likely to co-occur than by chance.

$$\text{PMI}(w_1, w_2) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- PMI is a measure of **association** from information theory
- If w_1 and w_2 are independent, then $P(w_1, w_2) = P(w_1)P(w_2)$
- PMI values range from -inf to inf

Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora

Why is this problematic?

- **The "-∞" Crash (Sparsity)** In real language data, most word pairs never appear together. If count(w, c) = 0, then P(w, c) = 0. **log(0) = -∞**
    - You cannot fill a matrix with negative infinity. It breaks standard linear algebra operations.

Unreliable Estimates (The "Silence" Logic)

- **Positive counts are facts:** If we see "Sherlock" and "Holmes" together 100 times, we are certain they are related.
- **Zero/Low counts are ambiguous:** If we see "Sherlock" and "pancake" together 0 times, is it because they are *antonyms*? Or just because our dataset isn't big enough?
- *Lesson:* In NLP, **absence of evidence ≠ evidence of absence**. Negative PMI values are usually just statistical noise derived from rare data.

Positive Pointwise Mutual Information (PPMI)

$$PPMI(w, c) = \max(PMI(w, c) , 0)$$

- If the association is positive (useful), keep it.
- If the association is negative (noise/missing), clamp it to 0.
- This restores the sparsity of the matrix (lots of zeros), which is computationally efficient.

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora

Why?

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora

- So we just replace negative PMI values by 0
- Positive PMI (PPMI) between word1 and word2:

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

Positive Pointwise Mutual Information

But note that there can be some signal in negative values (e.g., 'the of')

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora

- So we just replace negative PMI values by 0
- Positive PMI (PPMI) between word1 and word2:

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

Computing PMI

Assume we have a term-context matrix F with W rows (i.e., words) and C columns (i.e., contexts), where $f_{i,j}$ gives the number of times word w_i co-occurs with context word c_j.

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$\text{PMI}(w_i, c_j) = \log_2 \frac{p(w_i, c_j)}{p(w_i)p(c_j)}$$

$$p(w_i, c_j) = \frac{f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}} = \frac{\text{count}(w_i, c_j)}{\text{total co-occurrences}}$$

$$\text{PMI}(w_i, c_j) = \log_2 \frac{p(w_i, c_j)}{p(w_i)p(c_j)}$$

$$p(w_i, c_j) = \frac{f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}} = \frac{\text{count}(w_i, c_j)}{\text{total co-occurrences}}$$

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{ij}} = \frac{\text{\# of times } w_i \text{ co-occurs with a context word}}{\text{total co-occurrences}}$$

$$\text{PMI}(w_i, c_j) = \log_2 \frac{p(w_i, c_j)}{p(w_i)p(c_j)}$$

$$p(w_i, c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} = \frac{\text{count}(w_i, c_j)}{\text{total co-occurrences}}$$

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} = \frac{\text{\# of times } w_i \text{ co-occurs with a context word}}{\text{total co-occurrences}}$$

$$p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} = \frac{\text{\# of times } c_j \text{ co-occurs with a term word}}{\text{total co-occurrences}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Figure 6.10 Co-occurrence counts for four words in 5 contexts in the Wikipedia corpus, together with the marginals, pretending for the purpose of this calculation that no other words/contexts matter.

PMI(information, data) = ???

PMI example

$$p(\text{information, data}) = \frac{3,982}{11,716} = .3399$$

$$p(\text{information}) = \frac{7,703}{11,716} = .6575$$

$$p(\text{data}) = \frac{5,673}{11,716} = .4842$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$\text{PMI(information, data)} = \log_2 \frac{.3399}{(.6575)(.4842)} = 0.0944$$

PMI example

$$p(\text{information, data}) = \frac{3,982}{11,716} = .3399$$

$$p(\text{information}) = \frac{7,703}{11,716} = .6575$$

$$p(\text{data}) = \frac{5,673}{11,716} = .4842$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$\text{PMI(information, data)} = \log_2 \frac{.3399}{(.6575)(.4842)} = 0.0944$$

So what does this tell us about the words “information” and “data”?

PMI example

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Questions?

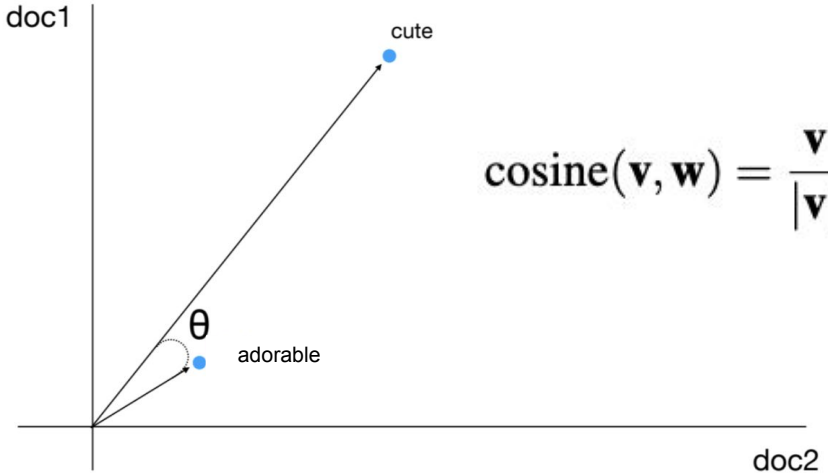
Computing Similarity

	doc1	doc2	doc3	doc4	doc5
cat	1	0	1	0	0
kitten	0	1	0	1	0
cute	1	0	1	0	0
adorable	1	1	1	0	0
gradients	0	0	1	1	1

do cute and adorable have "similar" meanings?

Computing Similarity

Cosine similarity



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

Example

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

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

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

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

Example

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

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

$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .018$$

$$\cos(\text{digital},\text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

Summary so far

- We now have a distributional profile of every word in the vocabulary
- That takes into account the fact that some words aren’t that discriminative
- But these counts are both noisy and sparse – many zeroes

From Counting to Predicting

Count-Based Methods (LSA, PMI)

- **Approach:** Scan the entire corpus first. Count how many times words appear together. Build a giant matrix.
- **Problem:**
  - **Heavy:** The matrix is huge (Vocabulary × Vocabulary).
  - **Static:** Updating the model with new data requires rebuilding the matrix.
  - **Sparse:** Mostly zeros.

Alternative: dense vectors

- **vectors which are**
  - **short** (length 50-1000)
  - **dense** (most elements are non-zero)

Why Prediction Wins

- **Iterative:** The model learns one sentence at a time (Streaming). You don't need to hold the whole internet in RAM.
- **Dense:** It naturally produces short, dense vectors (e.g., 300 dimensions) instead of sparse ones (e.g., 50,000 dimensions).
- **Nuance:** It captures complex relationships (like analogies) better than raw counts.

Benefits of dense vectors over sparse vectors

- Dense vectors may be easier to use as features in machine learning (fewer weights to learn)
- Lower dimensional = less computationally intensive
- Dense vectors may generalize better
  - Lower dimensionality may force abstraction: learn patterns that are not obvious from raw data
  - Lower dimensionality may remove noise: remove columns that don't improve predictive power
- Dense vectors may capture "second order" effects
  - E.g., if W1 occurs with C1, W2 occurs with C2, and C1 and C2 are similar -> W1, W2 similar
- In practice, they work better
- BUT: these dense vectors are less interpretable

Quick note on terminology

- Word vectors
- Word embeddings: tend to refer to dense vectors

Getting dense word embeddings

The approach we will consider today: Using weights in a classifier that predicts whether a word was seen in a particular context (word2vec)

Word2vec

- Popular embedding method
- Very fast to train
- Code available on the web
- Idea: **predict** rather than **count**

Prediction-Based Methods -dense vectors (word2vec)

- **Insight:** Instead of *counting* statistics, let's train a **classifier** to *predict* them.
- **The "Fake" Task:** We create a game for the neural network:
  - *"Given the word '_____', predict the probability that the next word is 'fox'."*
- **The Side Effect:** We don't actually care about the prediction. We care about the **weights** the model learns to solve the puzzle. These weights *are* the word embeddings.

Word2vec

- Instead of **counting** how often each word *w* occurs near "*apricot*"
- Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

- We don’t actually care about this task
  - But we'll take the learned classifier weights as the word embeddings

Use running text as implicitly supervised training data!

- A word *s* near *apricot*
  - Acts as gold ‘correct answer’ to the question
  - “Is word *w* likely to show up near *apricot*?”
- No need for hand-labeled supervision

Word2Vec: Skip-Gram Task

Word2vec provides a variety of options. Let's do

- "skip-gram with negative sampling" (SGNS)

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 logistic regression to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

Supervised machine learning

We need:

1. Function with parameters to learn
2. Training set of (t, c) pairs with labels
3. Loss function
4. Learning algorithm

Supervised machine learning

We need:

1. **Function with parameters to learn**
2. Training set of (t, c) pairs with labels
3. Loss function
4. Learning algorithm

Skip-Gram Training Data

Training sentence:

... lemon, a tablespoon of apricot jam a pinch ...

c1 c2 target c3 c4

Asssume context words are those in +/- 2 word window

Skip-Gram Goal

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

- (*apricot*, *jam*)
- (*apricot*, *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!
- Classification model
  - $P(+ \mid t, c) = 1/(1+\exp(-e_t \cdot e_c))$

Sigmoid function again!

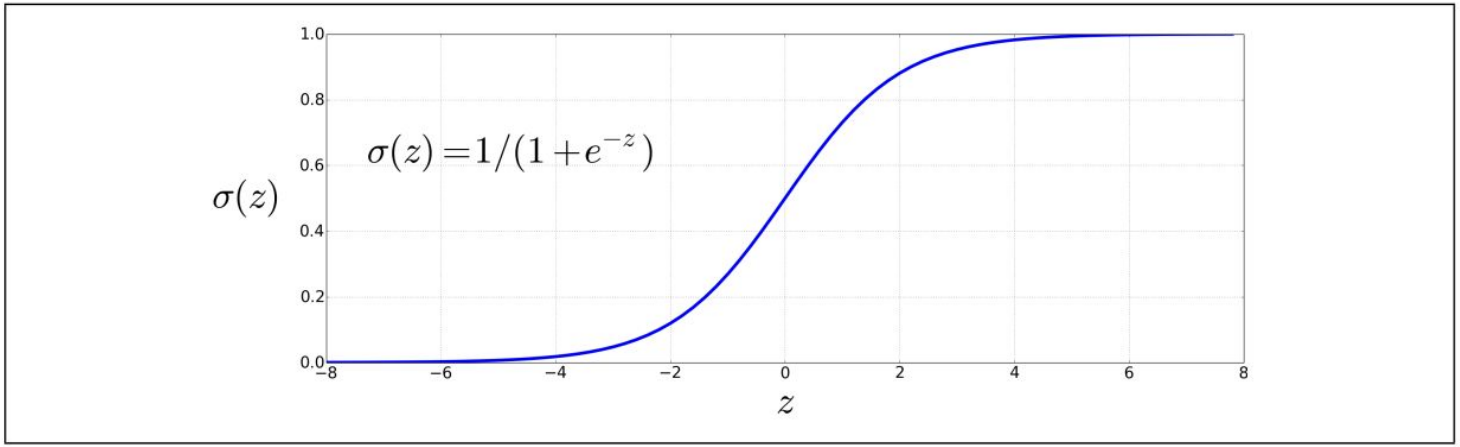


Figure 5.1 The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $(0, 1)$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

To compute this, we just need embeddings for the words

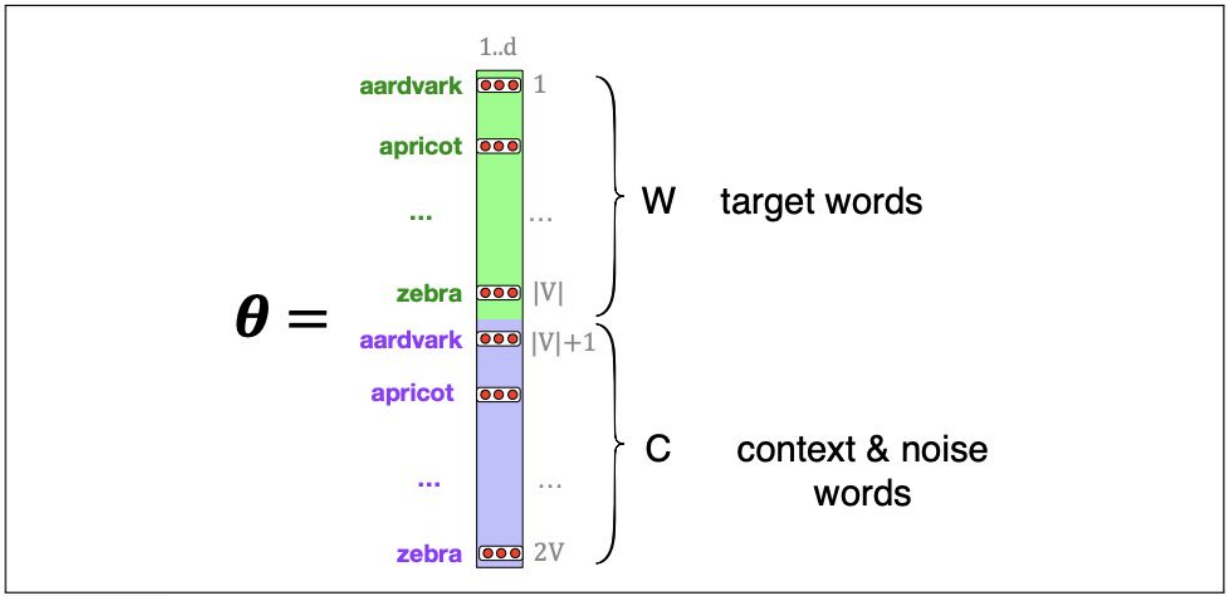


Figure 6.13 The embeddings learned by the skipgram model. The algorithm stores two embeddings for each word, the target embedding (sometimes called the input embedding) and the context embedding (sometimes called the output embedding). The parameter θ that the algorithm learns is thus a matrix of $2|V|$ vectors, each of dimension d, formed by concatenating two matrices, the target embeddings **W** and the context+noise embeddings **C**.

Supervised machine learning

We need:

1. **Function with parameters to learn**
2. Training set of (t, c) pairs with labels
3. Loss function
4. Learning algorithm

Supervised machine learning

We need:

1. Function with parameters to learn
2. **Training set of (t, c) pairs with labels**
3. Loss function
4. Learning algorithm

Skip-Gram Training Data

- Training sentence:
- ... lemon, a tablespoon of apricot jam a pinch ...
- c1 c2 t c3 c4

- Training data: input/output pairs centering on *apricot*
- Assume a +/- 2 word window

Skip-Gram Training

- Training sentence:
- ... lemon, a tablespoon of apricot jam a pinch ...
- c1 c2 t c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

- For each positive example, we'll create *k* negative examples.
- Using *noise* words
- Any random word that isn't *t*

Skip-Gram Training

- Training sentence:
- ... lemon, a tablespoon of apricot jam a pinch ...
- c1 c2 t c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Terminology

- This approach is **self-supervised**
  - It doesn’t require any hand-labelled labels
  - And yet we can use the data itself to get supervisory signal

Terminology

- This approach is **self-supervised**
  - It doesn’t require any hand-labelled labels
  - And yet we can use the data itself to get supervisory signal

Supervised machine learning

We need:

1. Function with parameters to learn
2. **Training set of (t, c) pairs with labels**
3. Loss function
4. Learning algorithm

Supervised machine learning

We need:

1. Function with parameters to learn
2. Training set of (t, c) pairs with labels
3. **Loss function**
4. Learning algorithm

Goal:

- **Maximize** the similarity of the target word, context word pairs (w , c_{pos}) drawn from the positive data

Goal:

- **Maximize** the similarity of the target word, context word pairs (w , c_{pos}) drawn from the positive data
- **Minimize** the similarity of the (w , c_{neg}) pairs drawn from the negative data.

Loss function for one *w with* c_{pos} , c_{neg1} ...c_{negk}

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the *k* negative sampled non-neighbor words.

$$L_{CE} = -\log \left[P(+|w, c_{pos}) \prod_{i=1}^{k} P(-|w, c_{neg_i}) \right]$$

Loss function for one *w with* c_{pos} , c_{neg1} ...c_{negk}

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the *k* negative sampled non-neighbor words.

$$L_{CE} = -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right]$$

$$= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right]$$

Loss function for one *w with* c_{pos} , c_{neg1} ...c_{negk}

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the *k* negative sampled non-neighbor words.

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right]
\end{aligned}
$$

Loss function for one *w with* c_{pos} , c_{neg1} ...c_{negk}

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the *k* negative sampled non-neighbor words.

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log \sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

Supervised machine learning

We need:

1. Function with parameters to learn
2. Training set of (t, c) pairs with labels
3. **Loss function**
4. Learning algorithm

Supervised machine learning

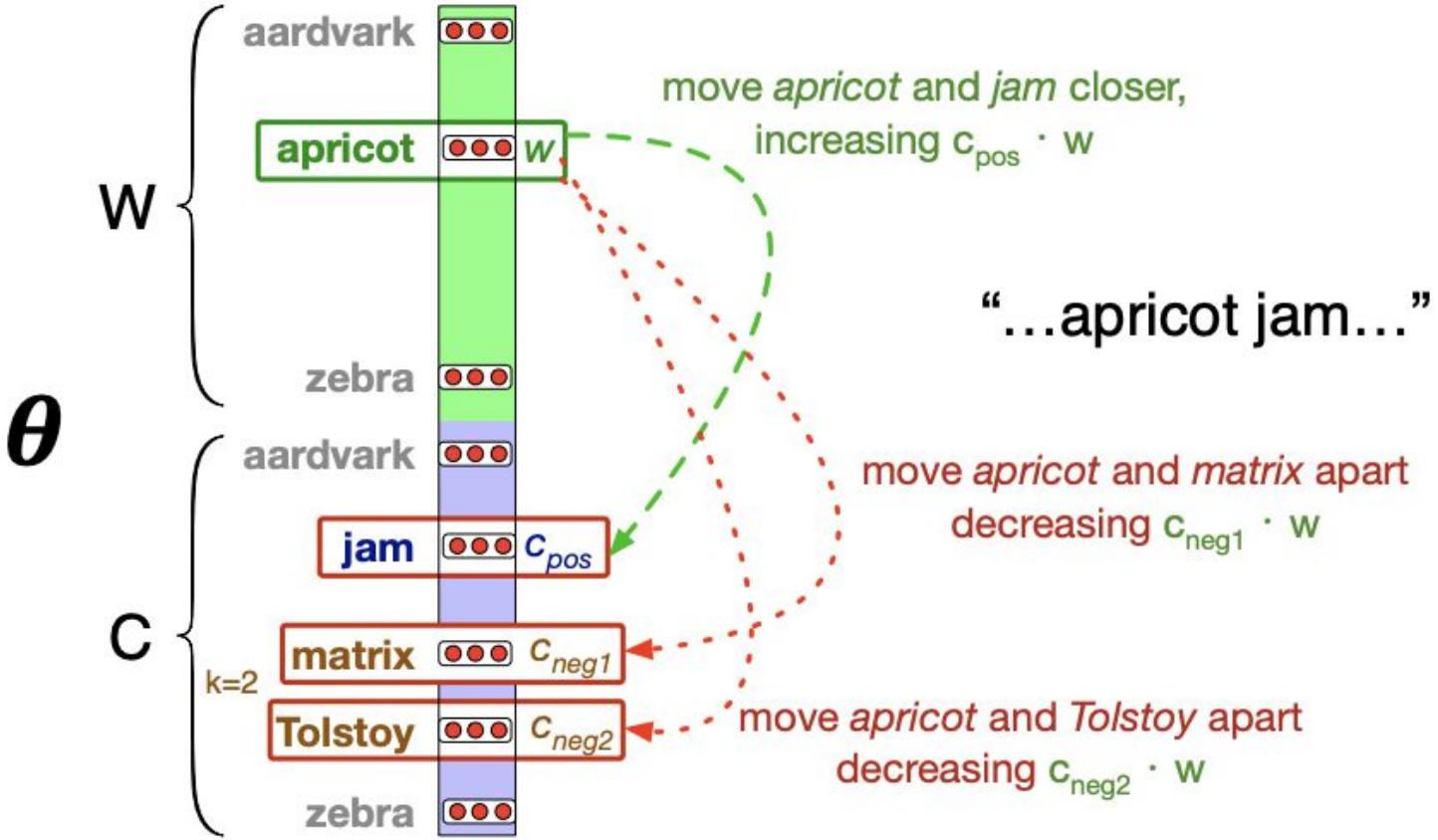
We need:

1. Function with parameters to learn
2. Training set of (t, c) pairs with labels
3. Loss function
4. **Learning algorithm**

Stochastic Gradient Descent

1. Randomly initialize w and c vectors (word vectors + context vectors)
2. See training instance: (w, c_pos), (w, c_neg1), (w, c_neg2)
3. Adjust w and c so as to make
    a. The positive pairs more similar (and thus more likely)
    b. The negative pairs more dissimilar (and thus less likely)

Intuition of one step of gradient descent



Reminder: gradient descent

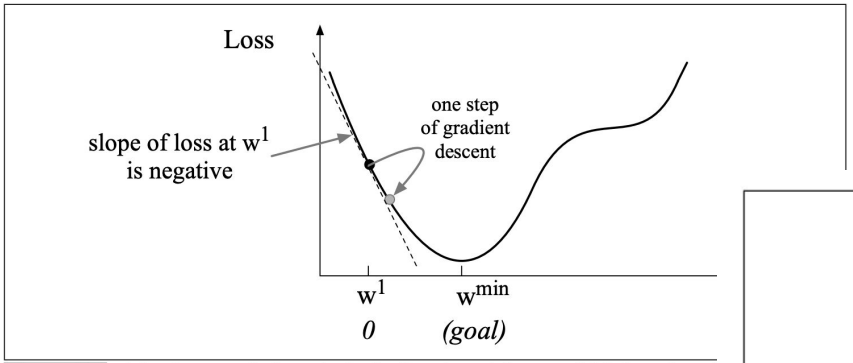


Figure 5.4 The first step in iteratively finding the minimum of this loss functi
w in the reverse direction from the slope of the function. Since the slope is neg
to move w in a positive direction, to the right. Here superscripts are used for
so w^1 means the initial value of w (which is 0), w^2 the value at the second step,

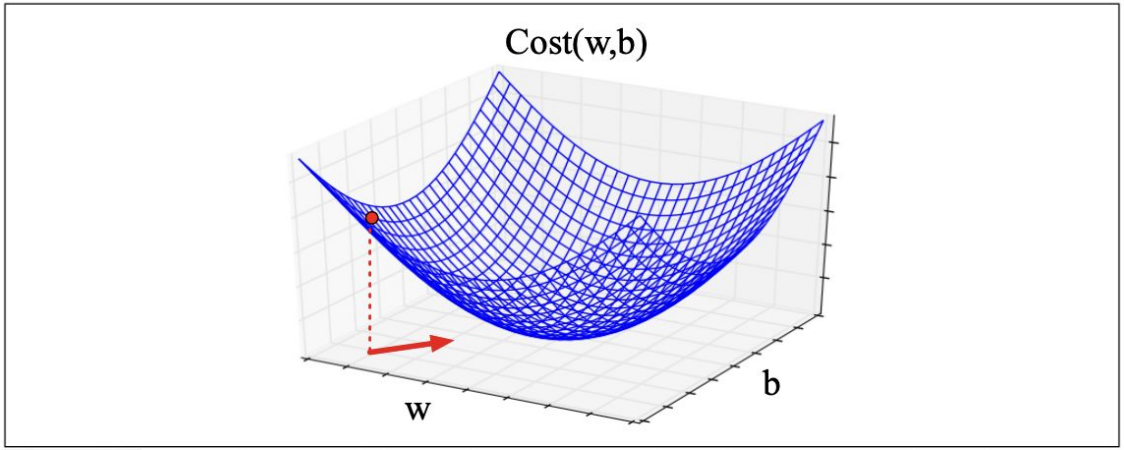


Figure 5.5 Visualization of the gradient vector at the red point in two dimensions w and b, showing a red arrow in the x-y plane pointing in the direction we will go to look for the minimum: the opposite direction of the gradient (recall that the gradient points in the direction of increase not decrease).

Reminder: gradient descent

- At each step
  - Direction: We move in the reverse direction from the gradient of the loss function
  - Magnitude: we move the value of this gradient $\frac{d}{dw} L(f(x;w),y)$ weighted by a **learning rate** η
  - Higher learning rate means move *w* faster

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w),y)$$

The derivatives of the loss function

$$L_{CE} = -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right]$$

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1]w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)]w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^{k} [\sigma(c_{\mathrm{neg}_i} \cdot w)]c_{neg_i}$$

Update equation in SGD

Start with randomly initialized C and W matrices, then incrementally do updates

$$c_{pos}^{t+1} = c_{pos}^{t} - \eta[\sigma(c_{pos}^{t} \cdot w^{t}) - 1]w^{t}$$

$$c_{neg}^{t+1} = c_{neg}^{t} - \eta[\sigma(c_{neg}^{t} \cdot w^{t})]w^{t}$$

$$w^{t+1} = w^{t} - \eta\left[[\sigma(c_{pos} \cdot w^{t}) - 1]c_{pos} + \sum_{i=1}^{k}[\sigma(c_{\mathrm{neg}_i} \cdot w^{t})]c_{neg_i}\right]$$

Two sets of embeddings

SGNS learns two sets of embeddings

Target embeddings matrix W

Context embedding matrix C

It's common to just add them together, representing word *i* as the vector $w_i + c_i$

Two sets of embeddings

The Target Vector:

- **Role:** The "Center of Attention."
- **When used:** When the word is the **input** given to the model.
- *Example:* "Given the word **dog**, predict its neighbors." (Here, "dog" uses its Target vector).

The Context Vector:

- **Role:** The "Neighbor."
- **When used:** When the word is a **possible output** or neighbor.
- *Example:* "Given the word 'fuzzy', is **dog** a likely neighbor?" (Here, "dog" uses its Context vector).

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

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after naïve bayes
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don't co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
  - Throw away the classifier code and keep the embeddings.

Dense embeddings you can download!

- **word2vec**
  - https://code.google.com/archive/p/word2vec/
- **Fasttext**
  - http://www.fasttext.cc/
- **Glove**
  - http://nlp.stanford.edu/projects/glove/

Slides credit

David Bamman (Berkeley), Alane Suhr, Kasia Hitczenko, Micha Elsner, Nathan Schneider, Sharon Goldwater, Hal Daumé III, Ellie Pavlick, Sam Bowman, Dan Jurafsky, Christopher Potts, Emily Bender