Word Vectors

Ling 250/450: Data Science for Linguistics

C.M. Downey

Spring 2026

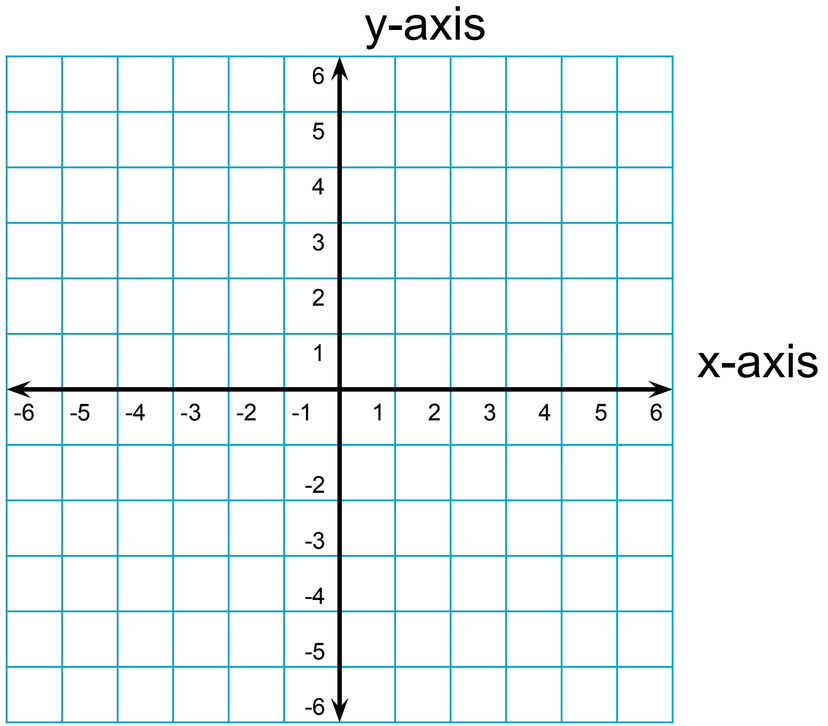
Word Vectors, Intro

Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a **dimension**

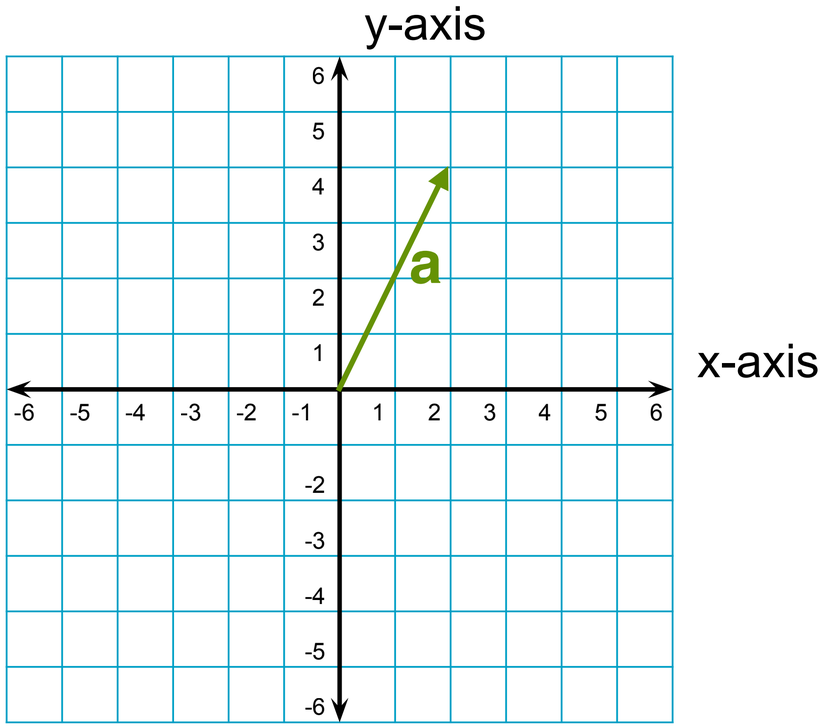
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a **dimension**



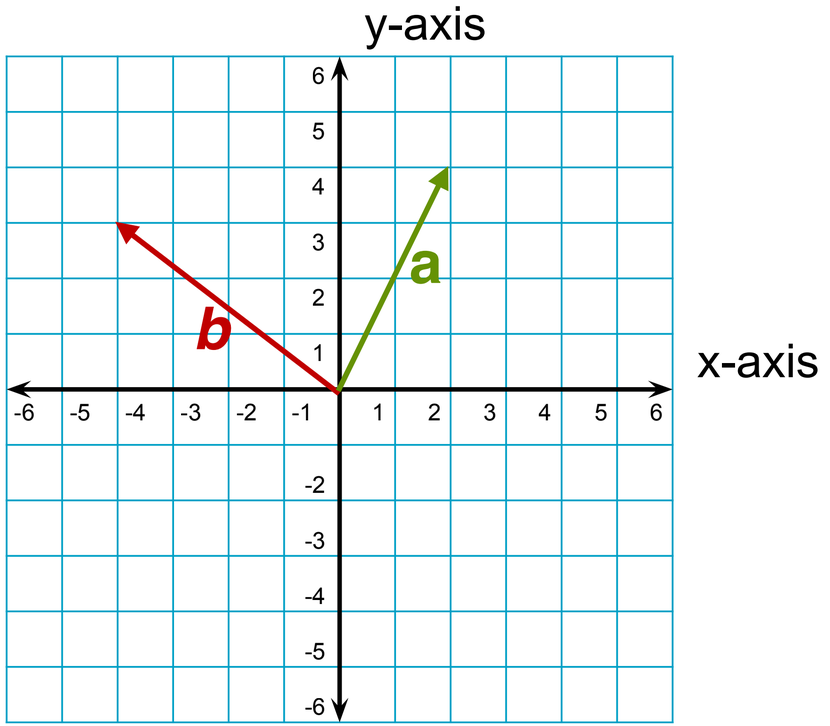
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a **dimension**
  - $\vec{a} = \langle 2,4 \rangle$



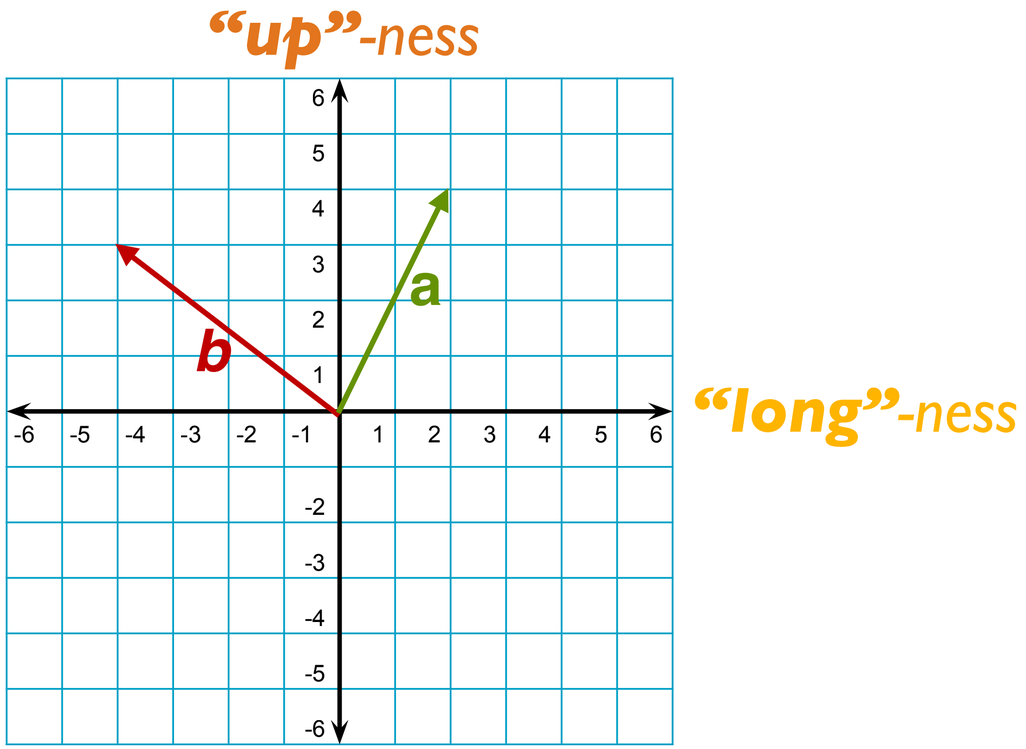
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a **dimension**
  - $\vec{a}$ = ⟨2,4⟩
  - $\vec{b}$ = ⟨-4,3⟩



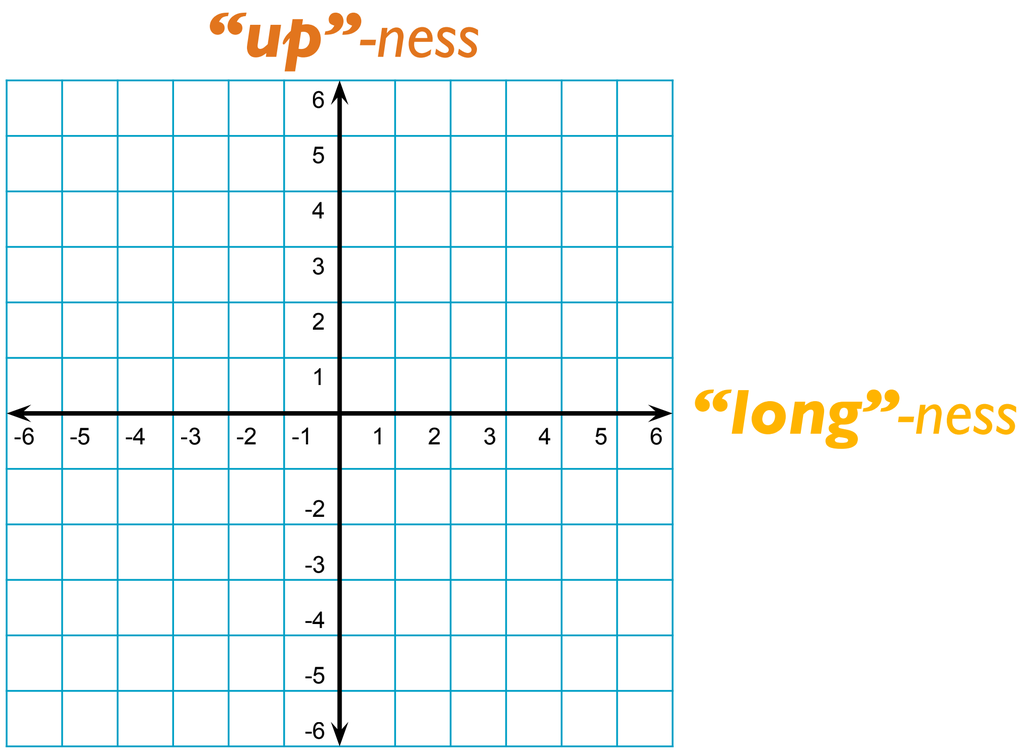
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a “dimension”
  - $\vec{a}$ = ⟨2,4⟩
  - $\vec{b}$ = ⟨-4,3⟩



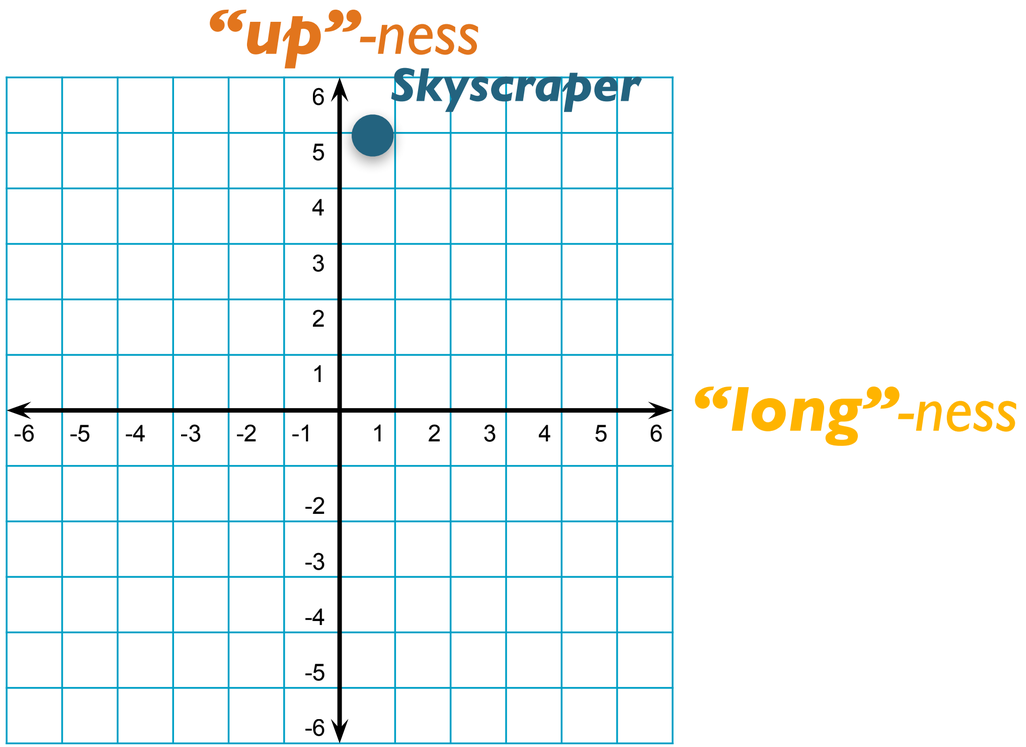
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a “dimension”
  - $\vec{a}$ = ⟨2,4⟩
  - $\vec{b}$ = ⟨-4,3⟩



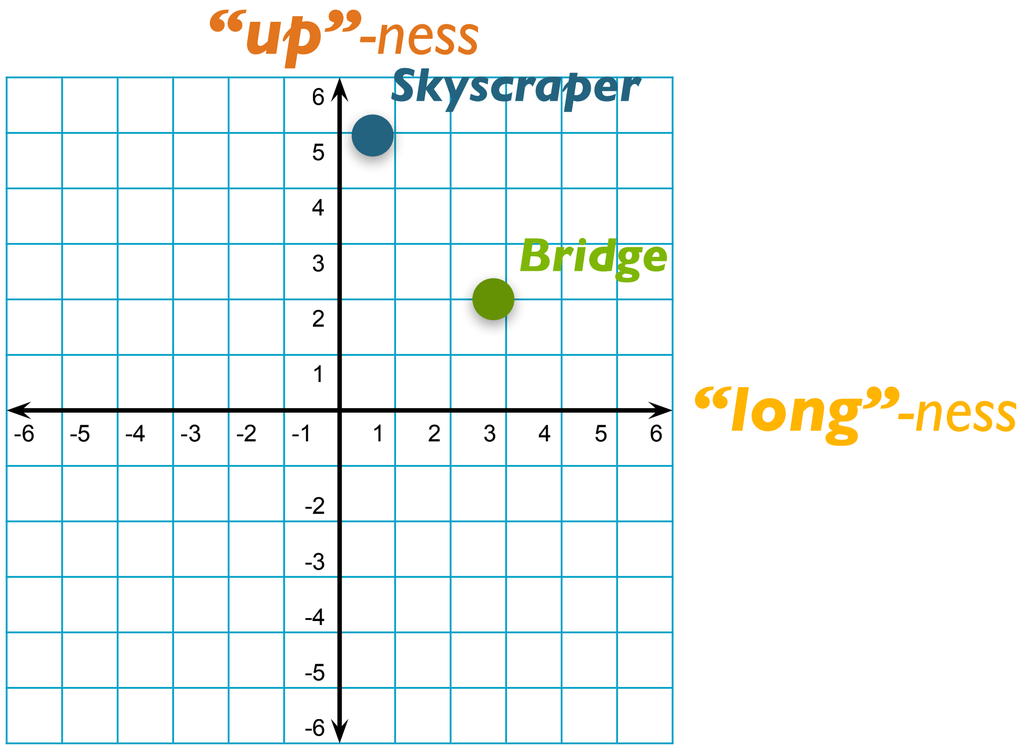
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a “dimension”
  - $\vec{a}$ = ⟨2,4⟩
  - $\vec{b}$ = ⟨-4,3⟩



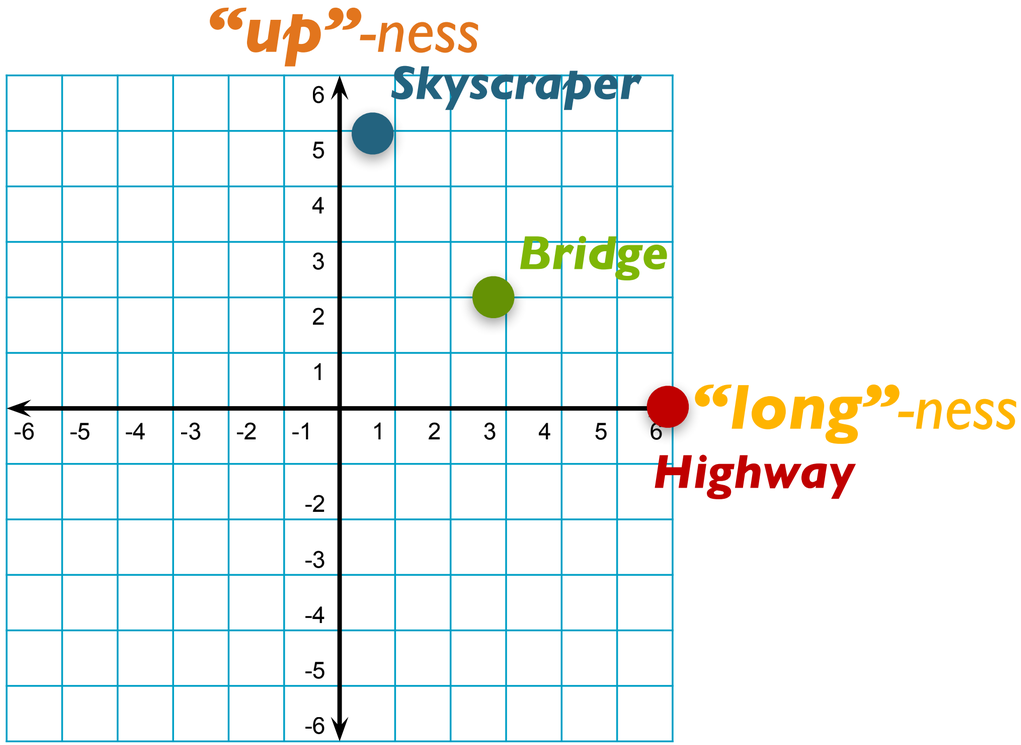
Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a “dimension”
  - $\vec{a}$ = ⟨2,4⟩
  - $\vec{b}$ = ⟨-4,3⟩

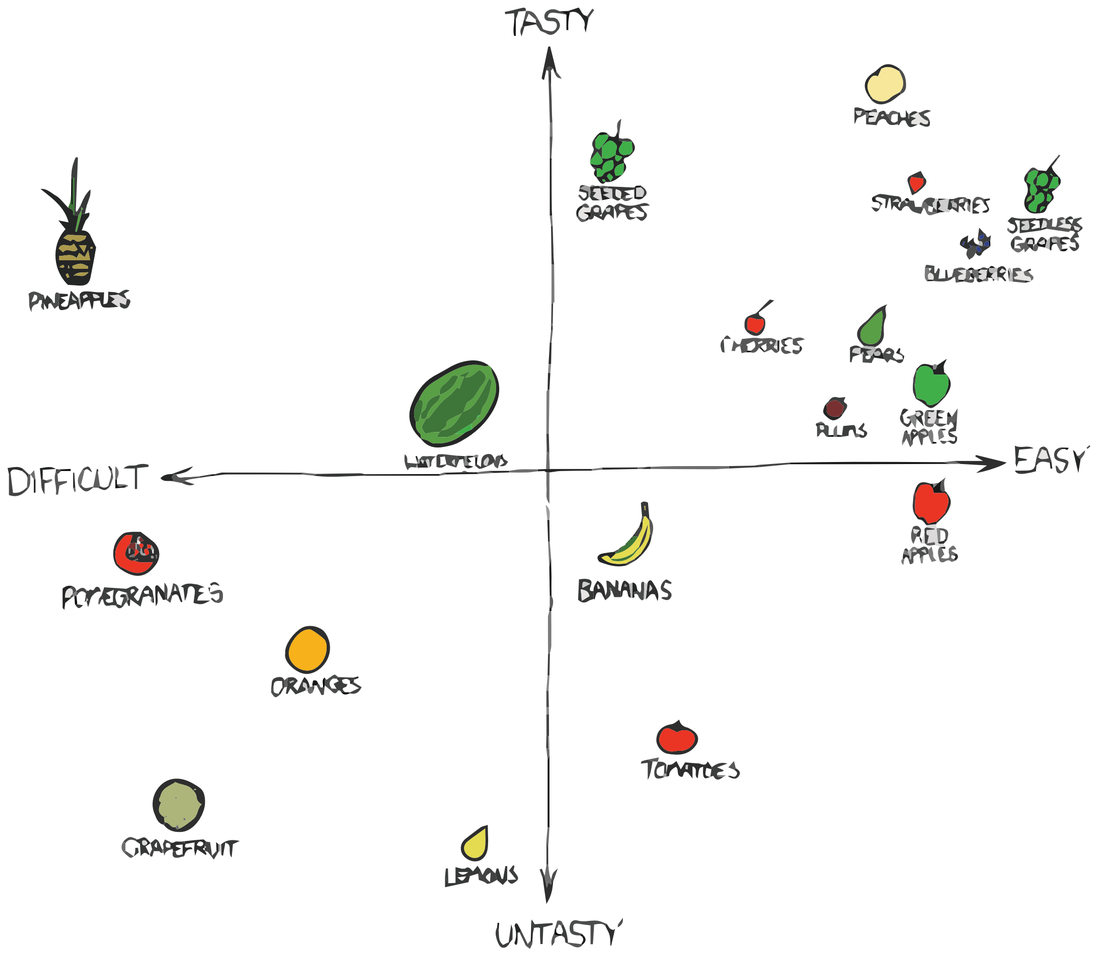


Vectors as information

- A vector is a list of numbers
- Each number can be thought of as representing a “dimension”
  - $\vec{a}$ = ⟨2,4⟩
  - $\vec{b}$ = ⟨-4,3⟩



Vectors as information



xkcd.com/388

Comparing Vectors

Comparing Vectors

- Any theory of meaning needs to capture **relatedness** between words
  - e.g. "cat" and "kitten" are more **closely related** than "scratch" and "ministry"
  - (You can learn to categorize these relationships in a Formal Semantics course)

Comparing Vectors

- Any theory of meaning needs to capture **relatedness** between words
  - e.g. "cat" and "kitten" are more **closely related** than "scratch" and "ministry"
  - (You can learn to categorize these relationships in a Formal Semantics course)
- How do we capture relationships between **word vectors?**
  - Luckily, we have all kinds of ways to **measure** vector relationships
  - These are mostly divided into metrics of **vector similarity** and **vector distance/ dissimilarity**
  - Which metrics are **closest to human intuitions?**

Vector Length

- A vector's **length** is equal to the *square root of the dot product with itself*
- This is an extension of the **Pythagorean Theorem** for right triangles
- Notice that **dot product** is closely related to length

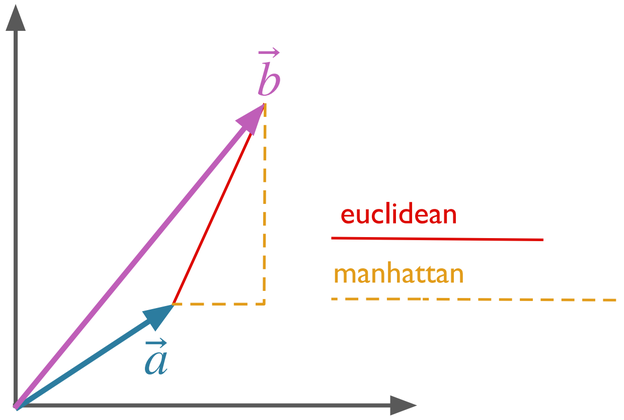
$$\text{length}(x) = \|x\| = \sqrt{x \cdot x}$$

Vector Distances: Manhattan & Euclidean

- **Manhattan Distance**
  - Distance as **cumulative horizontal + vertical moves**
  - Inspired by walking in a **city on a grid**
- **Euclidean Distance**
  - Our normal notion of distance
  - Length of a **straight line** between
- Both are sensitive to **extreme values**

$$d_{\text{manhattan}}(x, y) = \sum_i |x_i - y_i|$$

$$d_{\text{euclidian}}(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$



Vector Similarity: Dot Product

$$\text{sim}_{\text{dot}}(x, y) = x \cdot y = \sum_i x_i y_i$$

Vector Similarity: Dot Product

- Recall: I defined dot product as the "**strength** with which two vectors **go in the same direction**"
  - This can be used as a **similarity metric**
  - Produces a **scalar value**

$$\text{sim}_{\text{dot}}(x, y) = x \cdot y = \sum_i x_i y_i$$

Vector Similarity: Dot Product

- Recall: I defined dot product as the "**strength** with which two vectors **go in the same direction**"
  - This can be used as a **similarity metric**
  - Produces a **scalar value**
- Equals **zero** when the vectors are **orthogonal** (perpendicular)

$$\text{sim}_{\text{dot}}(x, y) = x \cdot y = \sum_i x_i y_i$$

Vector Similarity: Dot Product

- Recall: I defined dot product as the "**strength** with which two vectors **go in the same direction**"
  - This can be used as a **similarity metric**
  - Produces a **scalar value**
- Equals **zero** when the vectors are **orthogonal** (perpendicular)
- **Negative** when the vectors point in **opposite directions**

$$\text{sim}_{\text{dot}}(x, y) = x \cdot y = \sum_i x_i y_i$$

Vector Similarity: Dot Product

- Recall: I defined dot product as the "**strength** with which two vectors **go in the same direction**"
  - This can be used as a **similarity metric**
  - Produces a **scalar value**
- Equals **zero** when the vectors are **orthogonal** (perpendicular)
- **Negative** when the vectors point in **opposite directions**
- Problem: gives **higher similarity** to **longer vectors**

$$\text{sim}_{\text{dot}}(x, y) = x \cdot y = \sum_i x_i y_i$$

Vector Similarity: Cosine

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Vector Similarity: Cosine

- The skew towards longer vectors can be solved by **normalizing** the dot product by the **length** of the vectors

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Vector Similarity: Cosine

- The skew towards longer vectors can be solved by **normalizing** the dot product by the **length** of the vectors
- This happens to be the same as the **cosine of the angle** between them

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Vector Similarity: Cosine

- The skew towards longer vectors can be solved by **normalizing** the dot product by the **length** of the vectors
- This happens to be the same as the **cosine of the angle** between them
  - Equals **-1** if going in **opposite directions**

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Vector Similarity: Cosine

- The skew towards longer vectors can be solved by **normalizing** the dot product by the **length** of the vectors
- This happens to be the same as the **cosine of the angle** between them
  - Equals **-1** if going in **opposite directions**
  - **Zero** if the vectors are **orthogonal**

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Vector Similarity: Cosine

- The skew towards longer vectors can be solved by **normalizing** the dot product by the **length** of the vectors
- This happens to be the same as the **cosine of the angle** between them
  - Equals **-1** if going in **opposite directions**
  - **Zero** if the vectors are **orthogonal**
  - **+1** if going in the **same direction** (regardless of length)

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Vector Similarity: Cosine

- The skew towards longer vectors can be solved by **normalizing** the dot product by the **length** of the vectors
- This happens to be the same as the **cosine of the angle** between them
  - Equals **-1** if going in **opposite directions**
  - **Zero** if the vectors are **orthogonal**
  - **+1** if going in the **same direction** (regardless of length)
- This metric **correlates with human intuitions** of semantic relatedness (demonstrated through experiments)

$$\text{sim}_{\text{cosine}}(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

One-hot Vectors

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**
  - (Below is a one-hot representation of a vocabulary with three words)

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**
  - (Below is a one-hot representation of a vocabulary with three words)
- For a vocabulary with **size** $|V|$, the one-hots need **as many numbers**

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**
  - (Below is a one-hot representation of a vocabulary with three words)
- For a vocabulary with **size** $|V|$, the one-hots need **as many numbers**
  - Tens of thousands of words → giant vectors!

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**
  - (Below is a one-hot representation of a vocabulary with three words)
- For a vocabulary with **size** $|V|$, the one-hots need **as many numbers**
  - Tens of thousands of words → giant vectors!
  - The vectors are **sparse**: contain **mostly zeros**

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**
  - (Below is a one-hot representation of a vocabulary with three words)
- For a vocabulary with **size** $|V|$, the one-hots need **as many numbers**
  - Tens of thousands of words → giant vectors!
  - The vectors are **sparse**: contain **mostly zeros**
- Each word is **equally similar to all others**

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

One-hot Vectors

- The simplest possible word vectors are called **one-hot vectors**
  - **Each word** in the vocabulary is given an **"index"** (integer identification)
  - Word is represented by a vector where $x_i = 1$, with **zeros in all other indices**
  - (Below is a one-hot representation of a vocabulary with three words)
- For a vocabulary with **size** $|V|$, the one-hots need **as many numbers**
  - Tens of thousands of words → giant vectors!
  - The vectors are **sparse**: contain **mostly zeros**
- Each word is **equally similar to all others**
- How do we get **better representations?**

$$\text{cat}: \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \text{dog}: \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \text{bird}: \begin{bmatrix}0\\0\\1\end{bmatrix}$$

Distributional Similarity

- How do we create **semantically-structured** word vectors?

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**
- "You shall know a word by the company it keeps!" (*Firth, 1957*)

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**
- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**
- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**
- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**
- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.
  - We make *tezgüino* from corn.

Distributional Similarity

- How do we create **semantically-structured** word vectors?
- **"Distributional Hypothesis"**: similar words have **similar contexts**
- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.
  - We make *tezgüino* from corn.
- Tezgüino: corn-based alcoholic beverage.

Bag of Words Vectors

	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	...

Bag of Words Vectors

- We want words with **similar contexts** to have **similar vectors**
  - What if we directly represent the context as a vector?

	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	...

Bag of Words Vectors

- We want words with **similar contexts** to have **similar vectors**
  - What if we directly represent the context as a vector?
- Idea: for each word, **count** the number of times that any other word **appeared "close-by"**
  - Define "close-by" as appearing within a **sliding window** around the target word (i.e. $\pm n$ words, where n is chosen by the engineer)
  - Do this counting over a **corpus** (dataset) of language data

	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	...

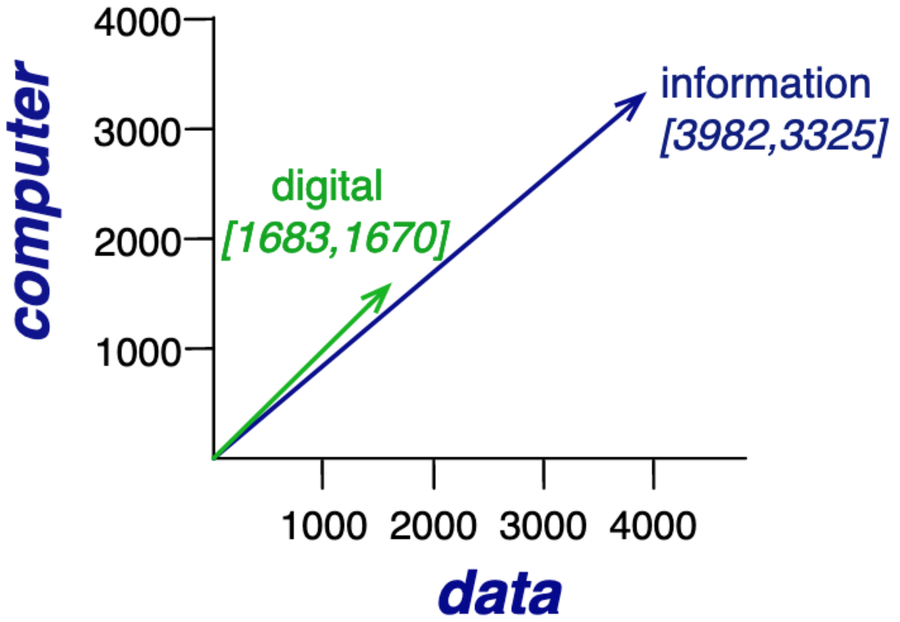
Bag of Words Vectors

- We want words with **similar contexts** to have **similar vectors**
  - What if we directly represent the context as a vector?
- Idea: for each word, **count** the number of times that any other word **appeared "close-by"**
  - Define "close-by" as appearing within a **sliding window** around the target word (i.e. $\pm n$ words, where n is chosen by the engineer)
  - Do this counting over a **corpus** (dataset) of language data
- What is the **size** of these vectors?
  - (Still $|V|$)

	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	...

Bag of Words Vectors

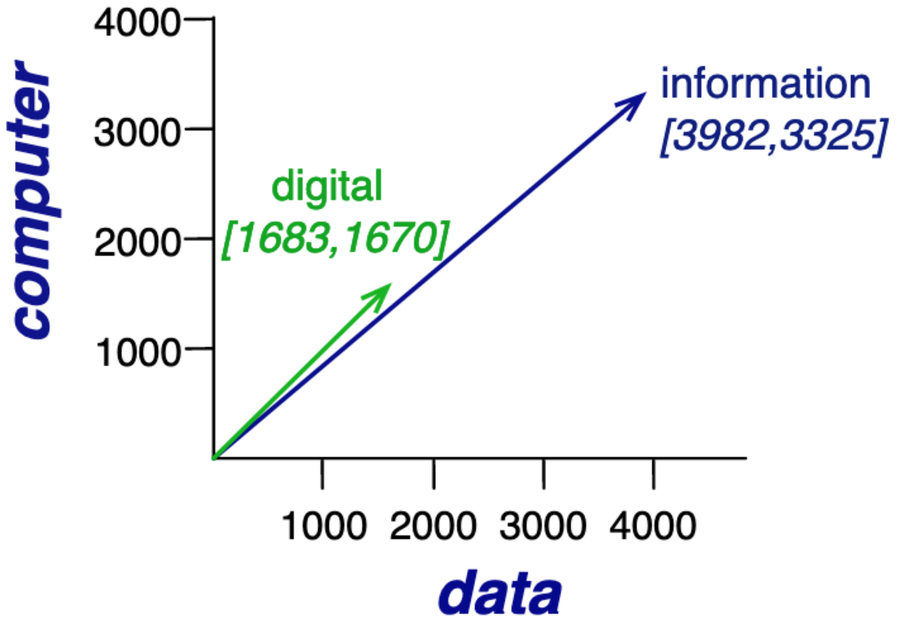
	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...



Bag of Words Vectors

- Can capture **basic similarities** between words

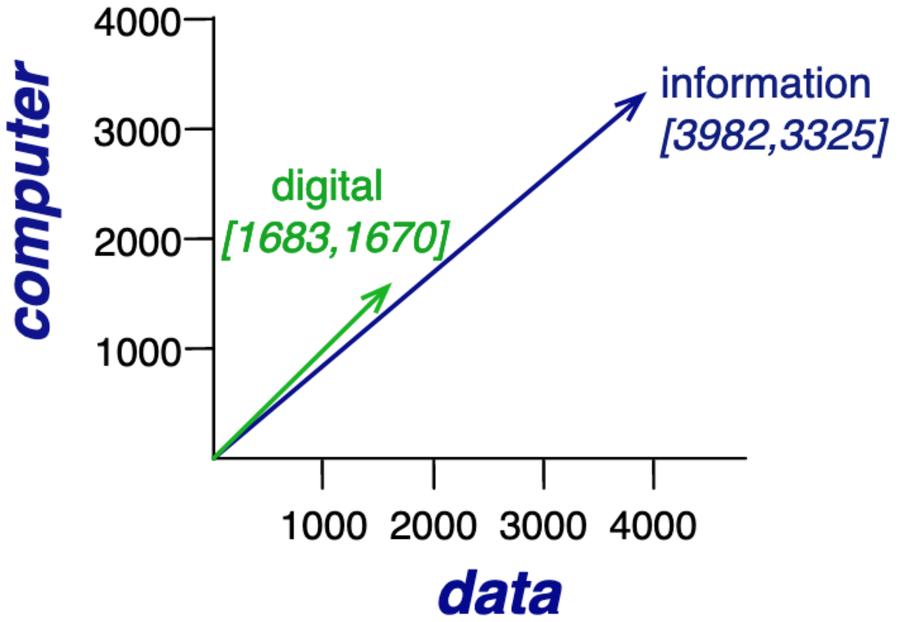
	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...



Bag of Words Vectors

- Can capture **basic similarities** between words
- Usually **re-weighted** by some more sophisticated algorithm
  - (e.g. tf-idf, ppmi)

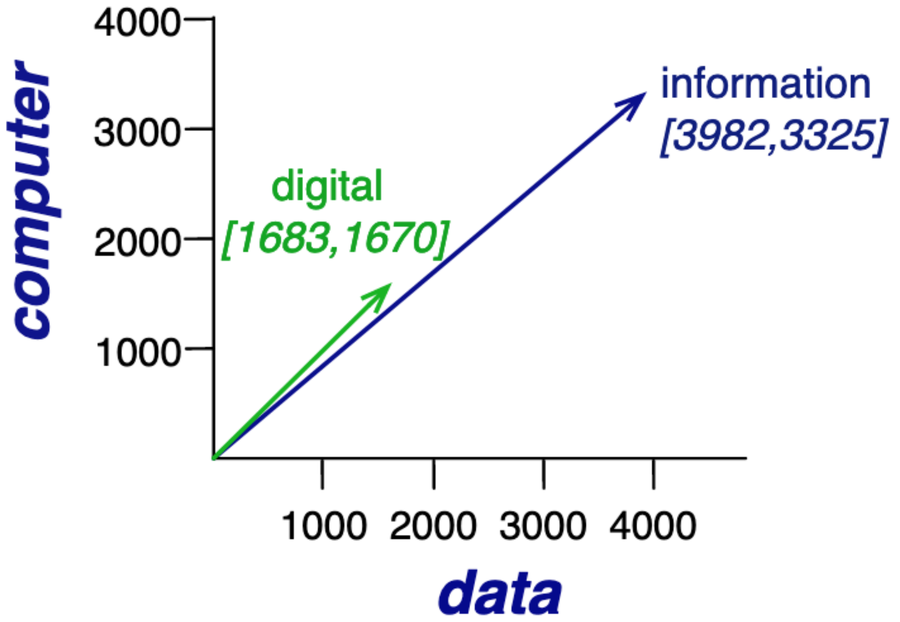
	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...



Bag of Words Vectors

- Can capture **basic similarities** between words
- Usually **re-weighted** by some more sophisticated algorithm
  - (e.g. tf-idf, ppmi)
- Problems:
  - Still very **high-dimensional** ($|V|$)
  - Still very **sparse** (most pairs of words **never co-occur**)

	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	...



Prediction-Based Models (word2vec)

Prediction-based Embeddings

Prediction-based Embeddings

- **Skip-gram** and **Continuous Bag of Words** (CBOW) models

Prediction-based Embeddings

- **Skip-gram** and **Continuous Bag of Words** (CBOW) models
- Words with **similar meanings** share **similar contexts**

Prediction-based Embeddings

- **Skip-gram** and **Continuous Bag of Words** (CBOW) models
- Words with **similar meanings** share **similar contexts**
- Instead of counting, train models to **predict context words**

Prediction-based Embeddings

- **Skip-gram** and **Continuous Bag of Words** (CBOW) models
- Words with **similar meanings** share **similar contexts**
- Instead of counting, train models to **predict context words**
- The **embeddings** (word vectors) used to accomplish the prediction become **semantically structured**

Skip-Gram vs. Continuous Bag of Words

Skip-Gram vs. Continuous Bag of Words

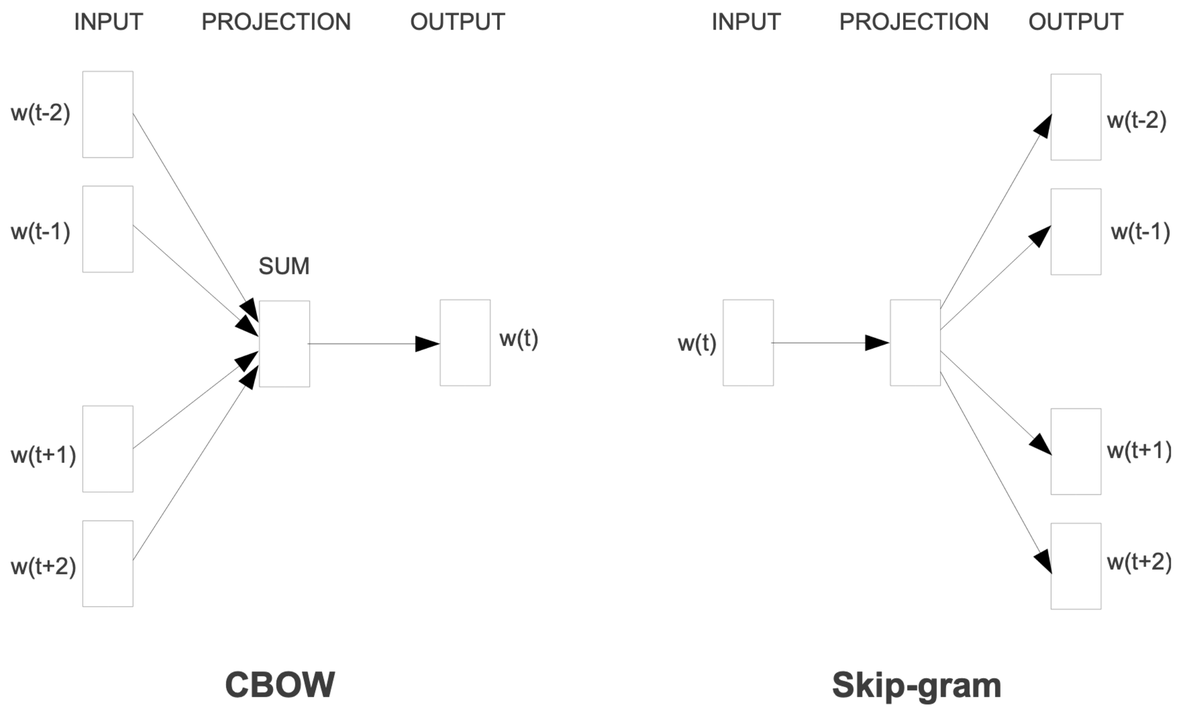
- Continuous Bag of Words (CBOW):
  - $P(word \mid context)$
  - Input: $(w_{t-1}, w_{t-2}, w_{t+1}, wt_{+2} \ldots)$
  - Output: $p(w_t)$

Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW):
  - ***P***(***word*** | ***context***)
  - Input: (w_{t-1}, w_{t-2}, w_{t+1}, wt_{+2} …)
  - Output: $p(\boldsymbol{w_t})$
- **Skip-gram:**
  - ***P***(***context*** | ***word***)
  - Input: $\boldsymbol{w_t}$
  - Output: $p(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \ldots)$

Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW)
  - ***P***(***word*** | ***context***)
  - Input: (w_{t-1}, w_{t-2}, w_{t+1}, wt_{+2} …)
  - Output: $p(w_t)$
- **Skip-gram:**
  - ***P***(***context*** | ***word***)
  - Input: w_t
  - Output: $p(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \ldots)$



Mikolov et al 2013a (the OG word2vec paper)

Skip-Gram Model

Skip-Gram Model

- Learns **two embedding matrices**
  - W : **word**, matrix of shape [vocab_size, embedding_dimension]
  - C : **context** embedding, matrix of same shape

Skip-Gram Model

- Learns **two embedding matrices**
  - W : **word**, matrix of shape [vocab_size, embedding_dimension]
  - C : **context** embedding, matrix of same shape
- Prediction task:
  - Given a word, **predict each neighbor** word in window
  - Compute $p(w_k \mid w_j)$ as proportional to $C_k \cdot W_j$
  - Convert to probability via Softmax

$$p(w_k \mid w_j) = \frac{e^{\mathbf{C}_k \cdot \mathbf{W}_j}}{\sum_i e^{\mathbf{C}_i \cdot \mathbf{W}_j}}$$

Generating Positive Examples

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w     c3       c4
```

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Generating Positive Examples

- Iterate through the corpus

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w     c3       c4
```

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Generating Positive Examples

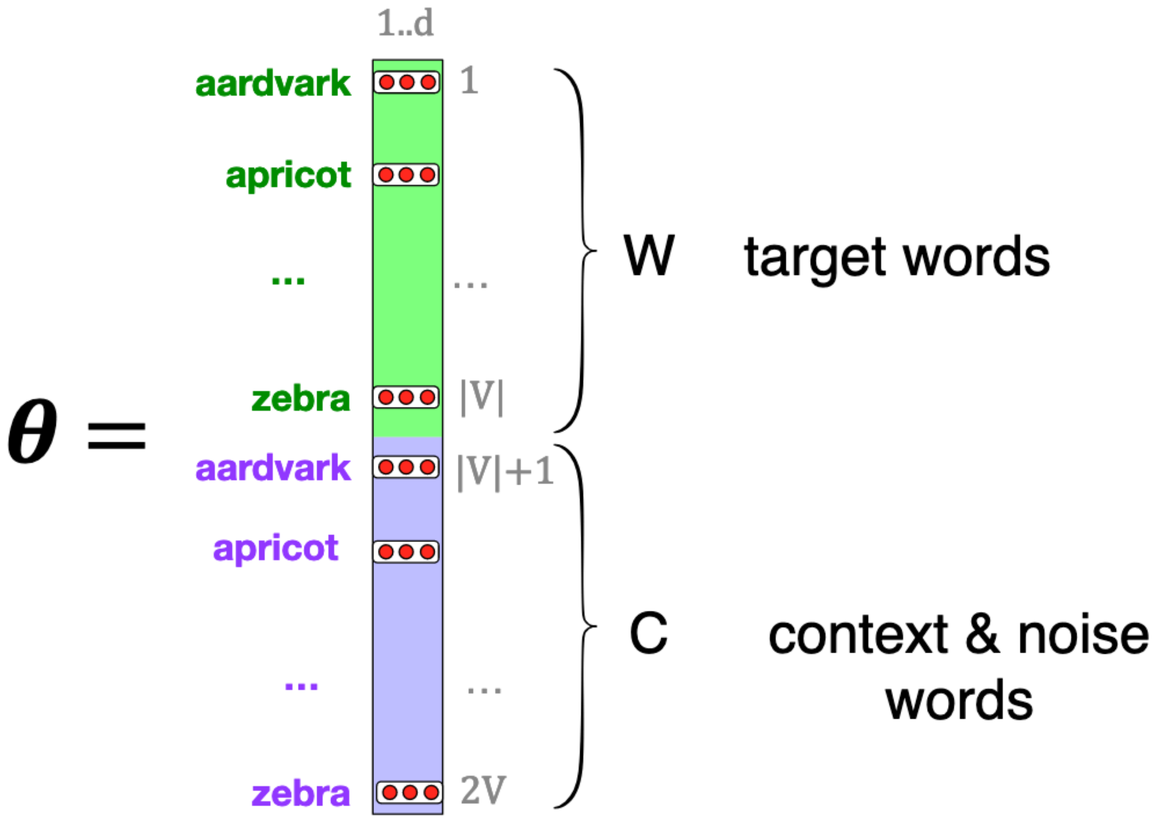
- Iterate through the corpus
- For each word: add **all words within *window_size*** of the current word as **positive** pairs
  - *window_size* is a **hyper-parameter**

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w     c3       c4
```

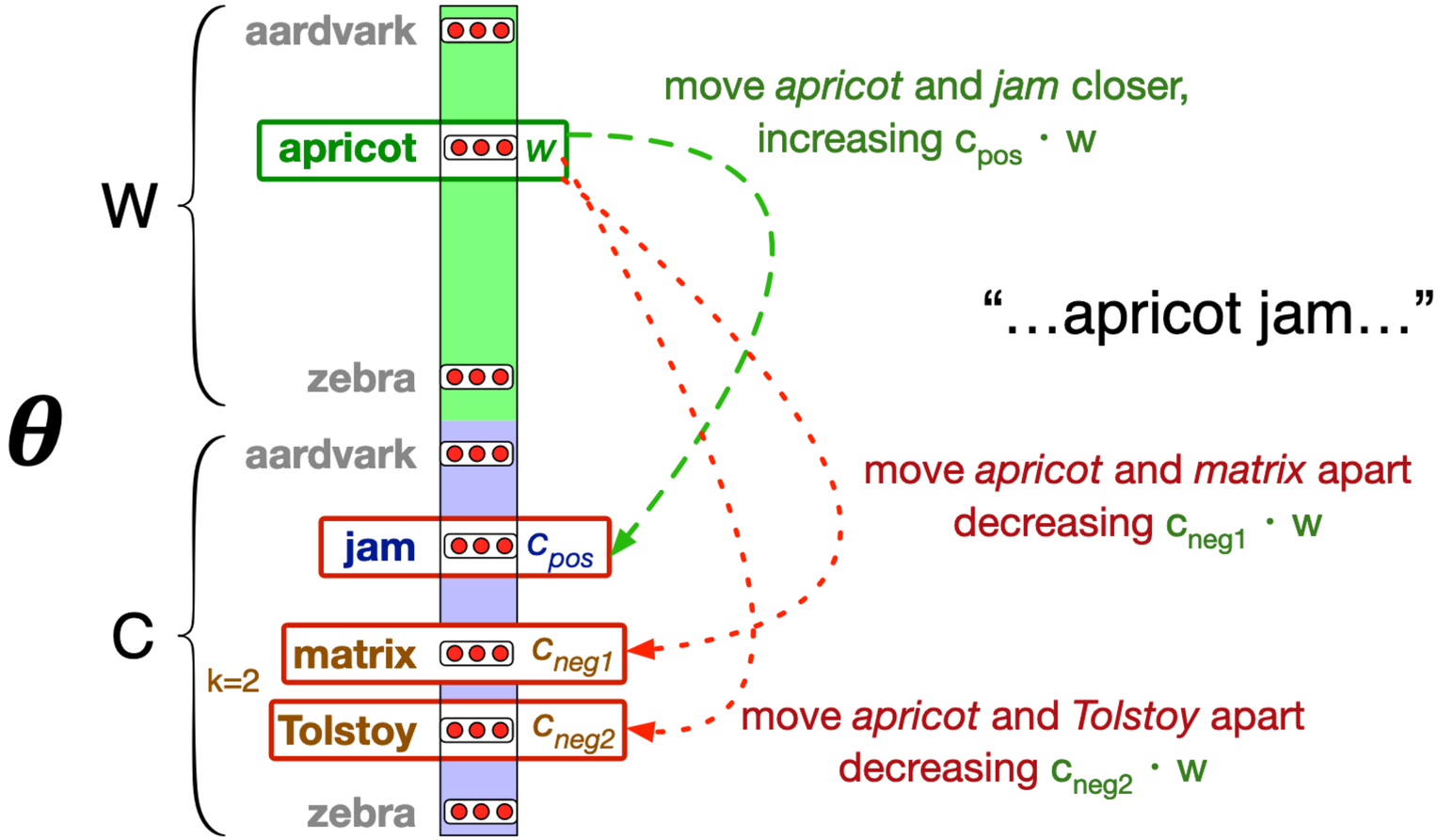
positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Learning: Parameters



Learning: Intuitively



Power of Prediction-based Embeddings

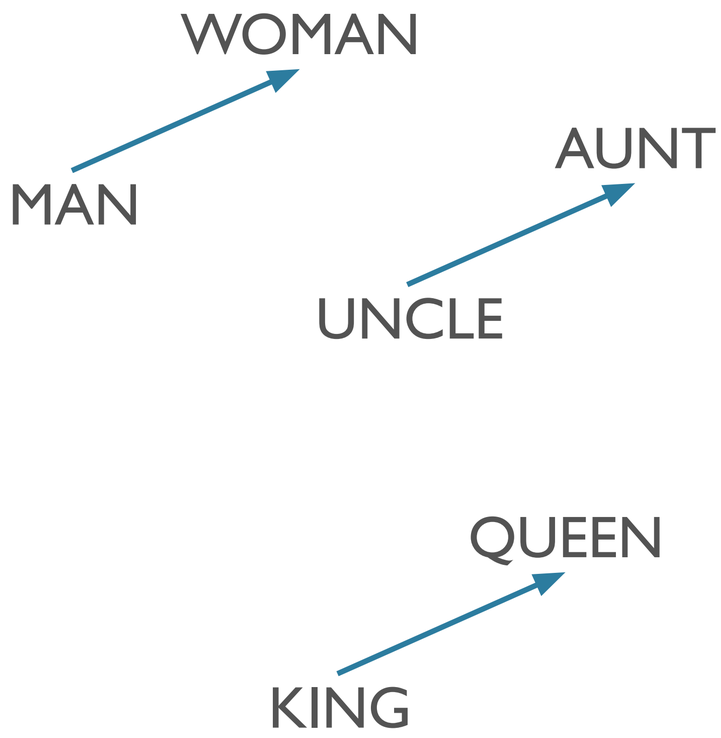
Power of Prediction-based Embeddings

- Count-based embeddings
  - Very high-dimensional ($|V|$)
  - Sparse
  - Pro: features are interpretable (“occurred with word W, N times in corpus”)

Power of Prediction-based Embeddings

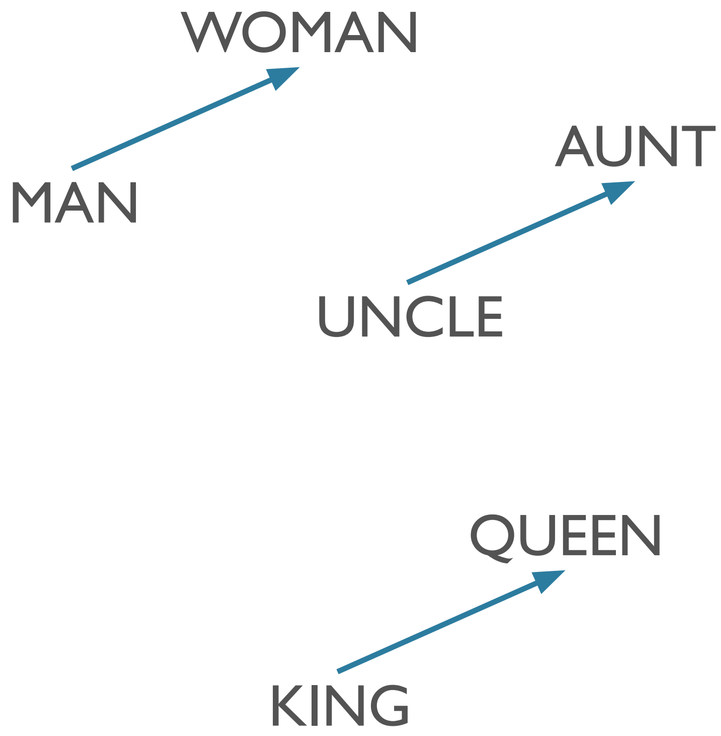
- Count-based embeddings
  - Very high-dimensional ($|V|$)
  - Sparse
  - Pro: features are interpretable (“occurred with word W, N times in corpus”)
- Prediction-based embeddings
  - **“Low”-dimensional** (typically ~256-2048)
  - **Dense**
  - Con: features are **not immediately interpretable**
    - What does “dimension 36 has value -9.63” mean?

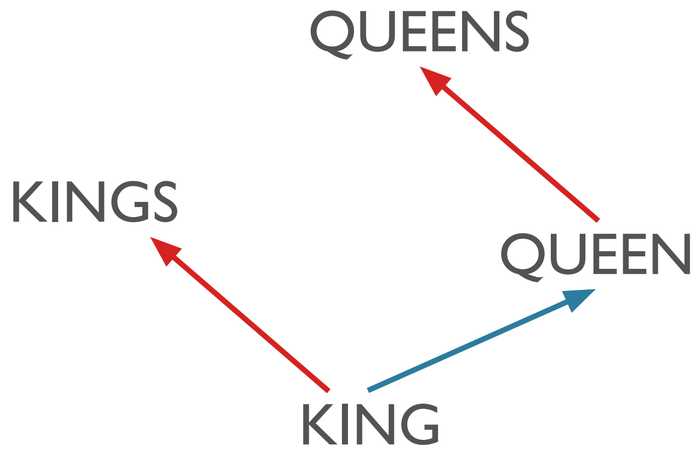
Relationships via Offsets



Mikolov et al 2013b

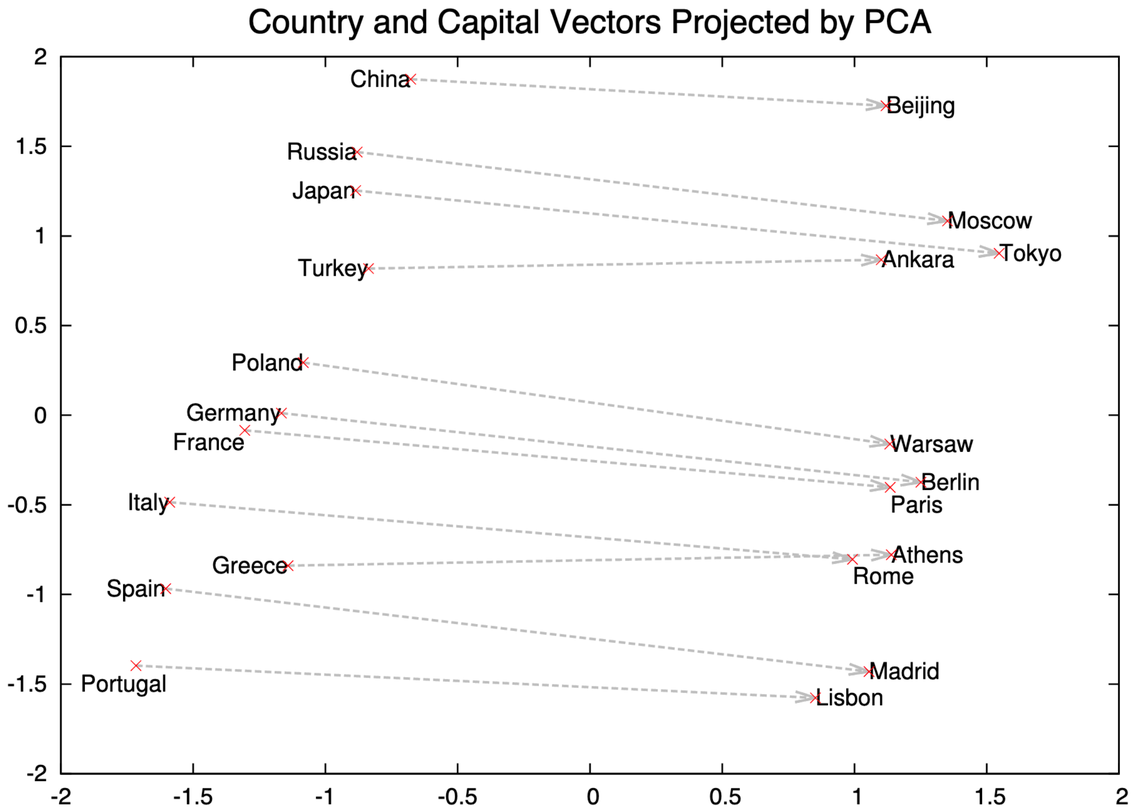
Relationships via Offsets





Mikolov et al 2013b

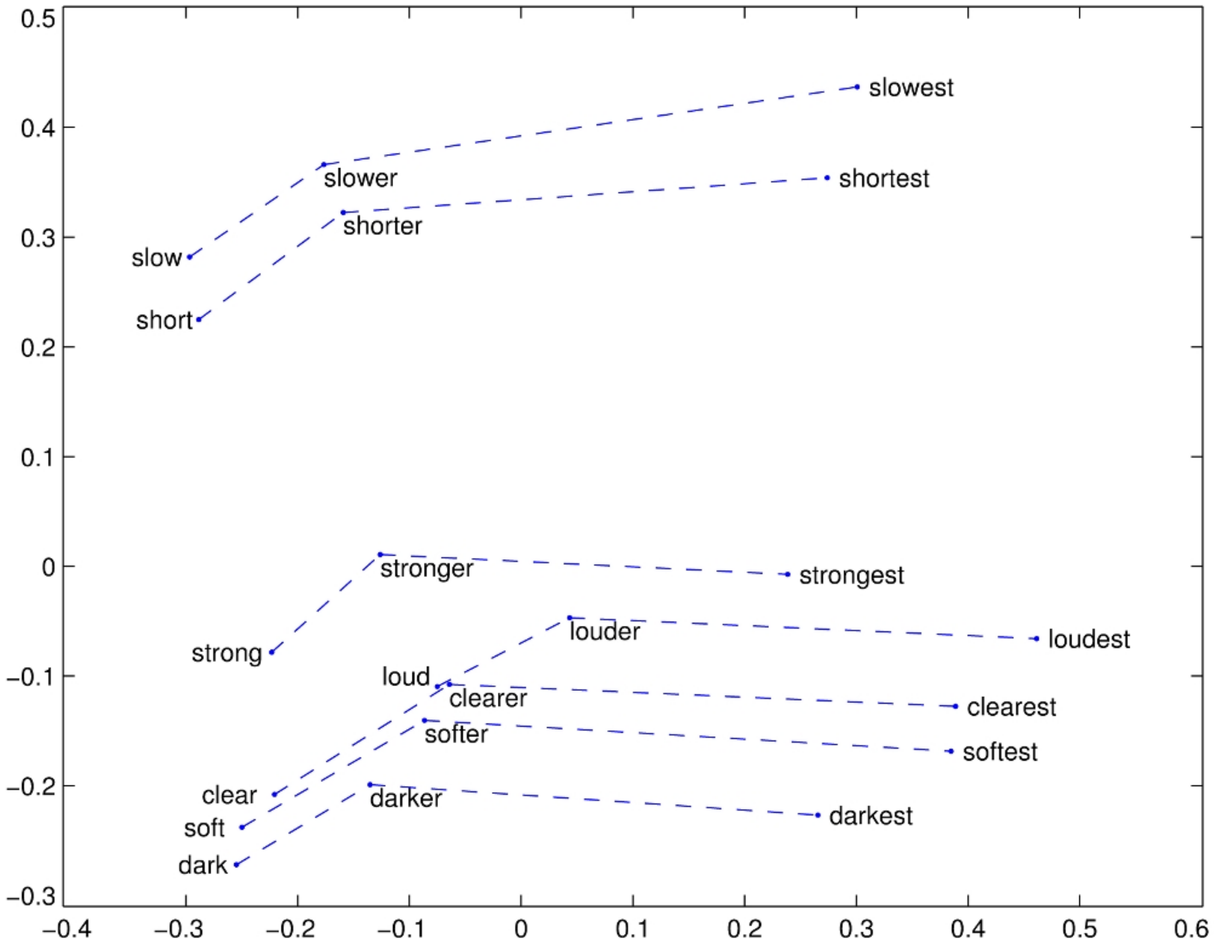
One More Example



[Mikolov et al 2013c]

Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

One More Example



Caveat Emptor

Issues in evaluating semantic spaces using word analogies

Tal Linzen
LSCP & IJN
École Normale Supérieure
PSL Research University
tal.linzen@ens.fr

Abstract

The offset method for solving word analogies has become a standard evaluation tool for vector-space semantic models: it is considered desirable for a space to represent semantic relations as consistent vector offsets. We show that the method's reliance on cosine similarity conflates offset consistency with largely irrelevant neighborhood structure, and propose simple baselines that should be used to improve the utility of the method in vector space evaluation.

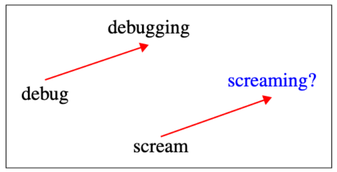


Figure 1: Using the vector offset method to solve the analogy task (Mikolov et al., 2013c).

cosine similarity to the landing point. Formally, if the analogy is given by

$$a : a^* :: b : \_\_\_ \quad (1)$$

Linzen 2016, a.o.

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi[1], **Kai-Wei Chang**[2], **James Zou**[2], **Venkatesh Saligrama**[1,2], **Adam Kalai**[2]
[1]Boston University, 8 Saint Mary's Street, Boston, MA
[2]Microsoft Research New England, 1 Memorial Drive, Cambridge, MA
tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and natural language processing tasks. We show that even word embeddings trained on Google News articles exhibit female/male gender stereotypes to a disturbing extent.

Bolukbasi et al 2016

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al NAACL 2019

Global vs. Contextual Word Vectors

Global vs. Contextual Word Vectors

- **Global** vectors: one vector per **word-type**
  - E.g. word2vec, GloVe
  - No difference between e.g. “play” as a verb, noun, or its different senses

Global vs. Contextual Word Vectors

- **Global** vectors: one vector per **word-type**
  - E.g. word2vec, GloVe
  - No difference between e.g. “play” as a verb, noun, or its different senses
- **Contextual** vectors: one vector per **word-occurrence**
  - “We saw a really great **play** last week.”
  - “Do you want to **play** basketball tomorrow?”
  - Each *occurrence* gets its own vector representation.

Global vs. Contextual Word Vectors

Peters et. al (2018)

	Source	Nearest Neighbors
Global	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
Contextual	Chico Ruiz made a spectacular **play** on Alusik's grounder…	Kieffer, the only junior in the group, was commended for his ability to hit in the clutch, as well as his all-round excellent **play.**
	Olivia De Havilland signed to do a Broadway **play** for Garson…	…they were actors who had been handed fat roles in a successful **play**, and had talent enough to fill the roles competently, with nice understatement.

Masked Language Modeling

Masked Language Modeling

- (“Causal”) Language Modeling: **next word prediction**

Masked Language Modeling

- ("Causal") Language Modeling: **next word prediction**
- **Masked Language Modeling** (a.k.a. Cloze task): **fill-in-the-blank**
  - Nancy Pelosi sent the articles of ____ to the Senate.
  - Seattle ____ some snow, so UW was delayed due to ____ roads.

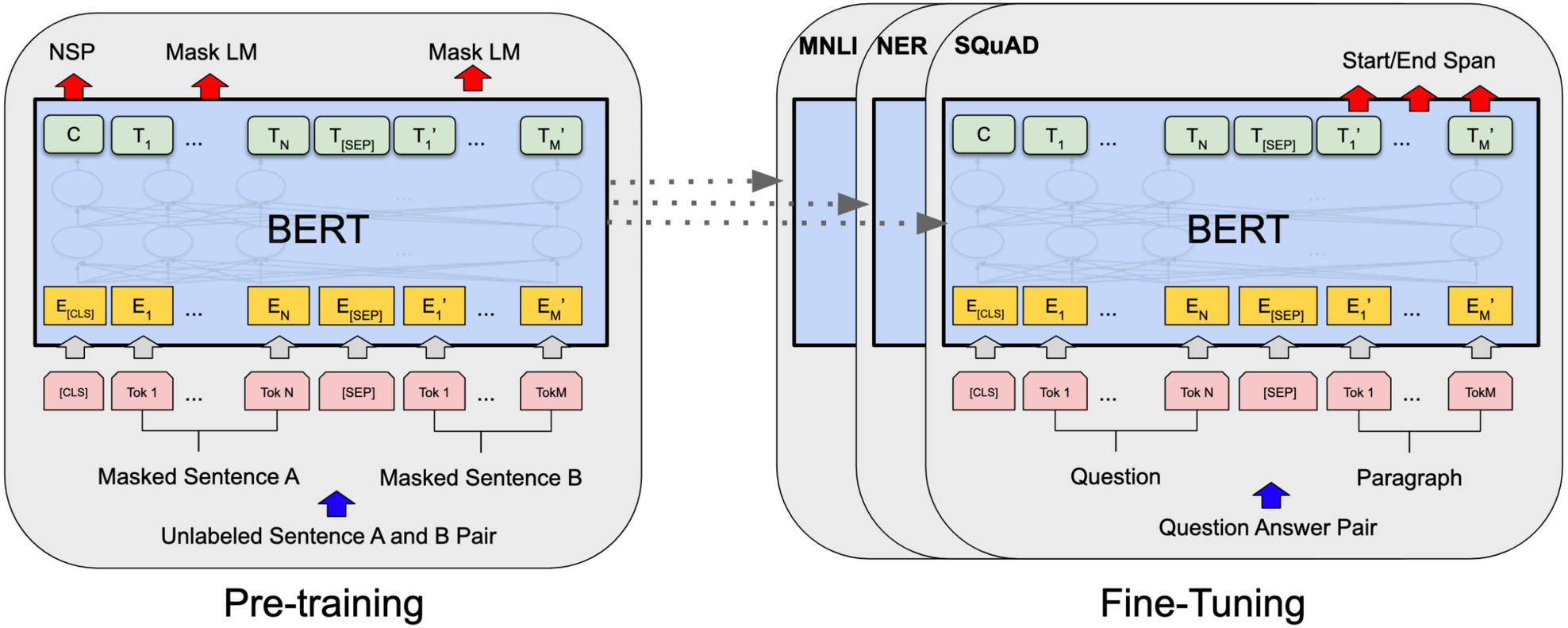
Masked Language Modeling

- ("Causal") Language Modeling: **next word prediction**
- **Masked Language Modeling** (a.k.a. Cloze task): **fill-in-the-blank**
  - Nancy Pelosi sent the articles of ____ to the Senate.
  - Seattle ____ some snow, so UW was delayed due to ____ roads.
- I.e. $P(w_t \mid w_{t+k}, w_{t+(k-1)}, \ldots, w_{t+1}, w_{t-1}, \ldots, w_{t-(m+1)}, w_{t-m})$
  - (very **similar to CBOW** from word2vec)

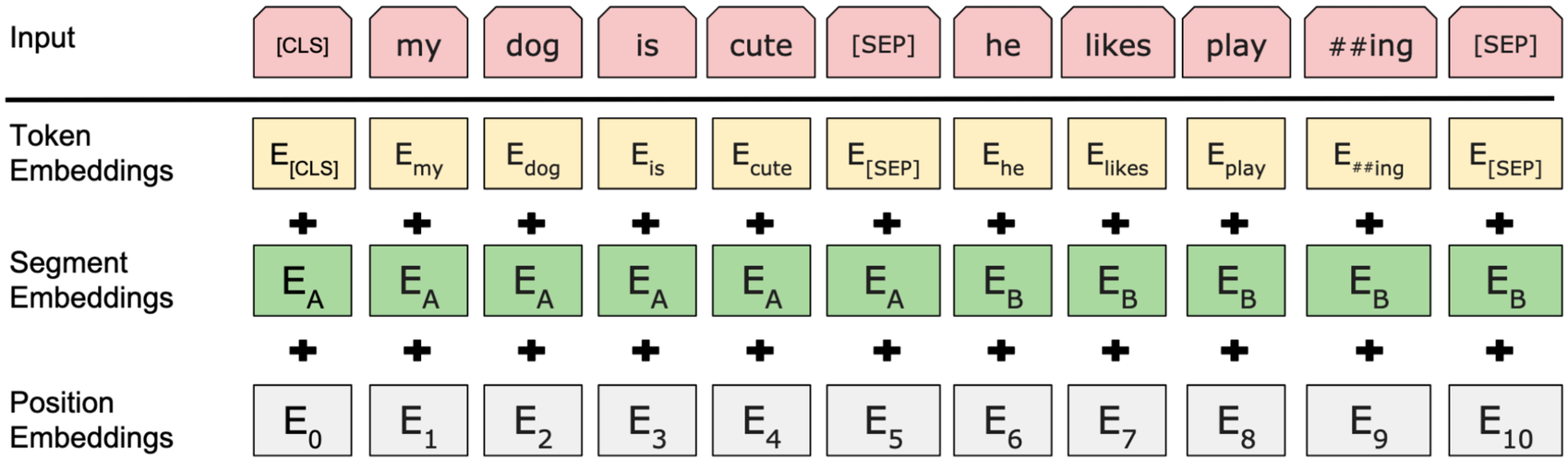
Masked Language Modeling

- ("Causal") Language Modeling: **next word prediction**
- **Masked Language Modeling** (a.k.a. Cloze task): **fill-in-the-blank**
  - Nancy Pelosi sent the articles of ____ to the Senate.
  - Seattle ____ some snow, so UW was delayed due to ____ roads.
- I.e. $P(w_t \mid w_{t+k}, w_{t+(k-1)}, \ldots, w_{t+1}, w_{t-1}, \ldots, w_{t-(m+1)}, w_{t-m})$
  - (very **similar to CBOW** from word2vec)
- Auxiliary training task: **next sentence prediction**
  - Given sentences A and B, binary classification: **did B follow A** in the corpus or not?

Schematically

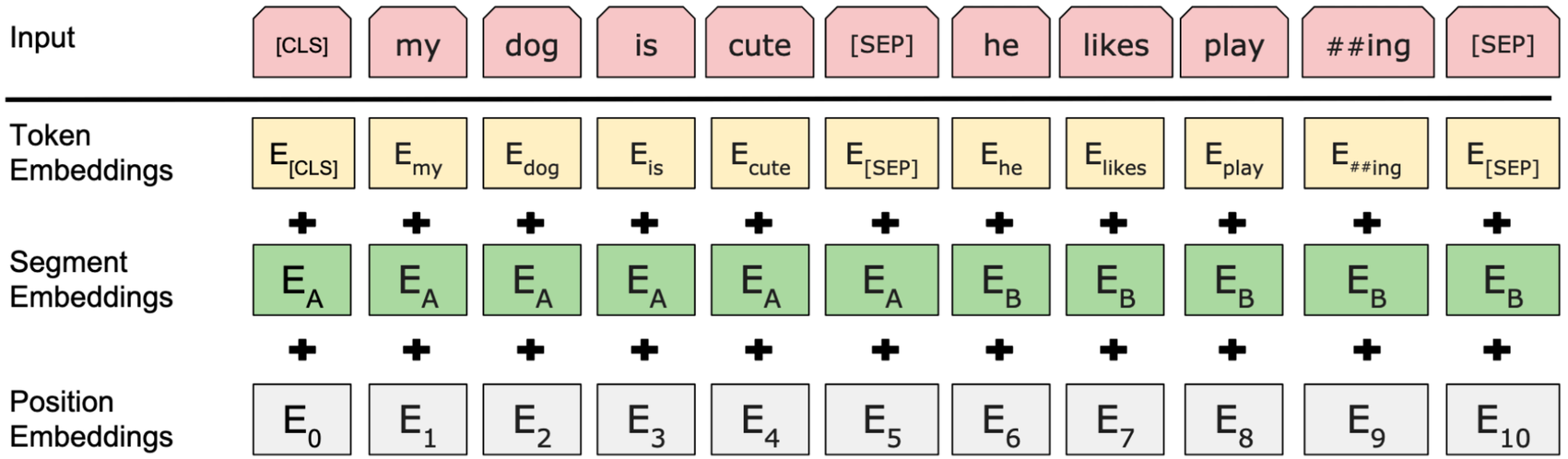


Input Representation



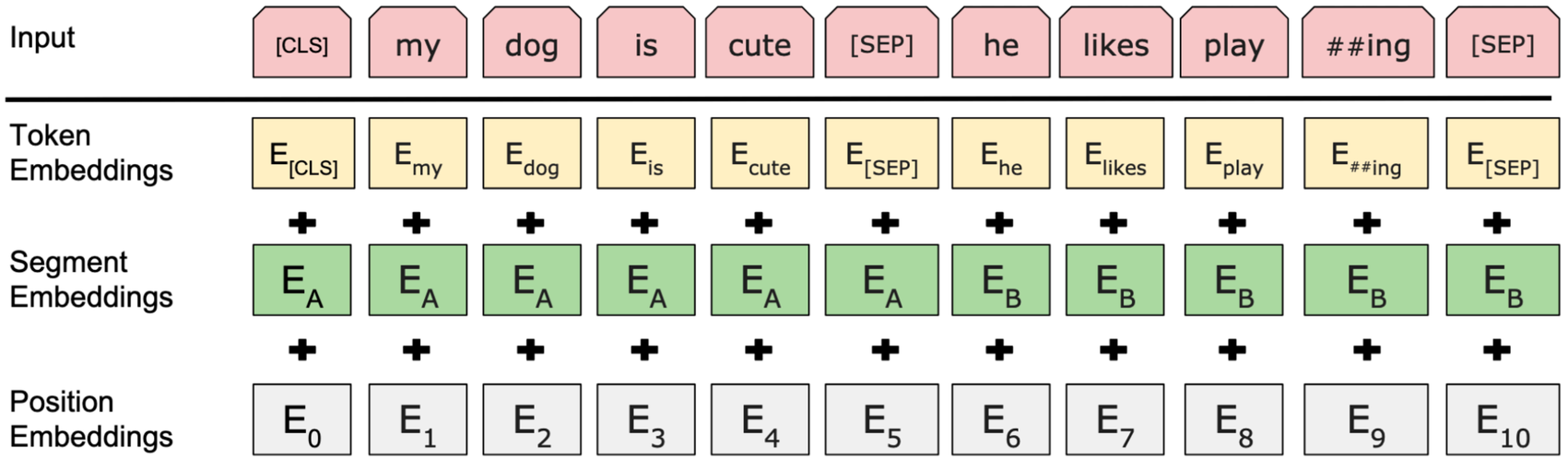
Input Representation

- [CLS], [SEP]: special tokens



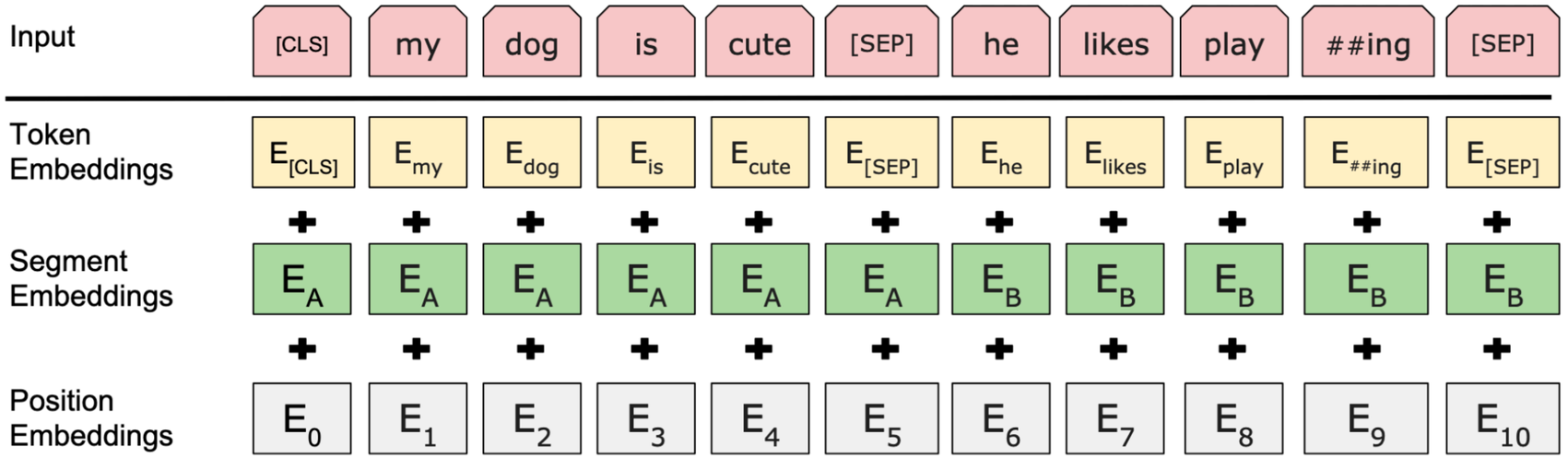
Input Representation

- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?



Input Representation

- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?
- Position embeddings: provide position in sequence (*learned* in this case, not fixed)



Training Details

Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)

Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- **Masking the input text:** 15% of all tokens are chosen, then:
  - 80% of the time: **replaced** by designated **[MASK] token**
  - 10% of the time: **replaced** by **random token**
  - 10% of the time: **unchanged**

Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- **Masking the input text:** 15% of all tokens are chosen, then:
  - 80% of the time: **replaced** by designated **[MASK] token**
  - 10% of the time: **replaced** by **random token**
  - 10% of the time: **unchanged**
- Loss is cross-entropy of the LM prediction **at the masked positions**

Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- **Masking the input text:** 15% of all tokens are chosen, then:
  - 80% of the time: **replaced** by designated **[MASK] token**
  - 10% of the time: **replaced** by **random token**
  - 10% of the time: **unchanged**
- Loss is cross-entropy of the LM prediction **at the masked positions**
- Max seq. length: 128 tokens for first 90%, 512 tokens for final 10%

Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- **Masking the input text:** 15% of all tokens are chosen, then:
  - 80% of the time: **replaced** by designated **[MASK] token**
  - 10% of the time: **replaced** by **random token**
  - 10% of the time: **unchanged**
- Loss is cross-entropy of the LM prediction **at the masked positions**
- Max seq. length: 128 tokens for first 90%, 512 tokens for final 10%
- 1M training steps, batch size 256 = **4 days on 4/16 TPUs** (base/large)

BERT Today

BERT Today

- Still a de-facto gold standard for **contextual word vectors**
  - Representations used to **probe for linguistic features** like Part of Speech, Semantic Role, Narrative Role, Animacy, etc.

BERT Today

- Still a de-facto gold standard for **contextual word vectors**
  - Representations used to **probe for linguistic features** like Part of Speech, Semantic Role, Narrative Role, Animacy, etc.
- Widely used, partly due to its **reasonable size / compute needs**
  - Was considered a "Large Language Model" when it came out
  - Modern LLMs are **at least 1000x times bigger**