

# WORD EMBEDDING

## *Count-based and Predictive Models*

---

Prof. Marco Viviani

[marco.viviani@unimib.it](mailto:marco.viviani@unimib.it)



# From sparse to dense vectors

- **A Co-occurrence Matrix in reality is constituted by a very large number of words**
  - For each word, TF-IDF or PPMI or other weighted vectors are:
    - **long** (length  $|V| = 20,000$  to  $50,000$ )
    - **sparse** (most elements are equal to zero)
- There are techniques to learn **lower-dimensional vectors** for words, which are:
  - **short** (length = 50 to 1,000) (usually around 300)
  - **dense** (most elements are non-zero)
- These dense vectors in a **latent space** are called **embeddings**.

# WORD EMBEDDINGS

---

Low-dimensional dense word vectors

# Learning Embeddings (Dense Vectors)

- **Models** to generate word embeddings
  - Models producing “static” embeddings include:
    - Count-based models (Distributed semantic models)
    - Predictive models (Neural network models)
  - Models producing “dynamic”, “contextualized” embeddings:
    - Contextual models (Transformers) → Next block of slides

# Count-based models

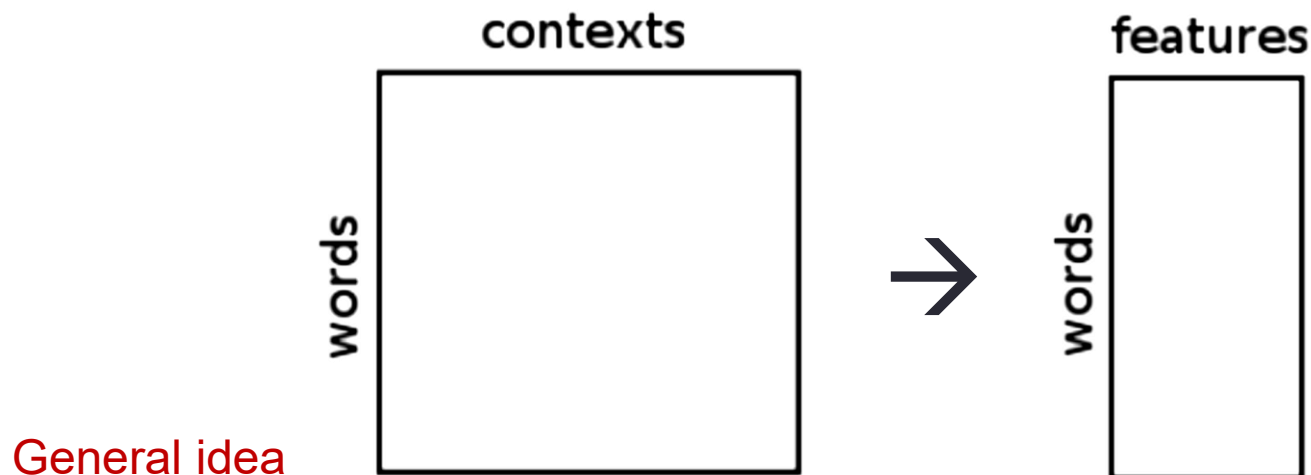
- **Count-based models**

- Compute the statistics of how often each **word co-occurs** with its neighbor words in a large text corpus;
- Then **map** these count-statistics down to a **small, dense vector** for each word.

- Count-based models learn vectors by doing **dimensionality reduction** on a term-context matrix.
  - The **term-context matrix** contains the information on how frequently each “word” (stored in rows), is seen in some “context” (the columns).
- They **factorize** this matrix to yield a **lower-dimensional matrix** of **words** and (latent) **features**, where each row yields a (dense) vector representation for each word.

# Count-based models

- Latent Dirichlet Allocation (LDA)
  - Based on a term-document matrix (suitable for topic modeling)
- **Singular Value Decomposition** (SVD) → Linear algebra
  - **Latent Semantic Analysis** (LSA) → Specific use of SVD to uncover hidden semantic structure in text data
- **GloVe** (Pennington, Socher, Manning, 2014)



# Predictive models

- **Predictive models** directly try to predict a word from its **neighbors** in terms of learned small, dense embedding vectors (considered parameters of the model)
- **Neural-network-inspired** models
- **word2vec** (Mikolov et al., 2013)
- **FastText** (Bojanowski et al., 2016)

# COUNT-BASED MODELS

---

Singular Value Decomposition (SVD)



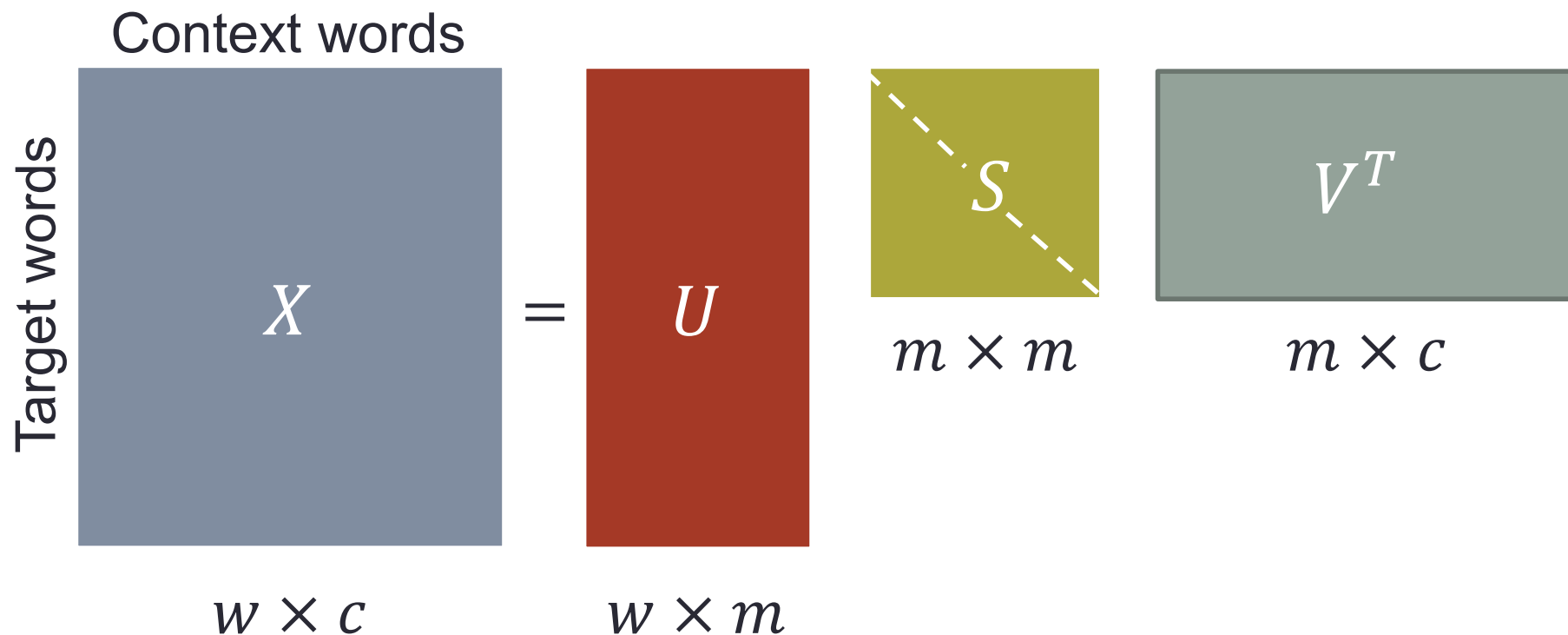
# Singular Value Decomposition

- Any rectangular  $w \times c$  matrix  $X$  can be expressed as the product of 3 matrices:
  - $U$ : a  $w \times m$  matrix where the  $w$  rows correspond to rows of the original matrix  $X$ , but the  $m$  columns represents a dimension (**feature**) in a **new latent space**.
  - $S$ : diagonal  $m \times m$  matrix of **singular values** expressing the **importance** of each dimension (**feature**).
  - $V^T$ : transposed  $m \times c$  matrix where the  $c$  columns correspond to the columns of the original matrix  $X$ , but the  $m$  rows correspond to singular values.

**Classic linear algebra result.**

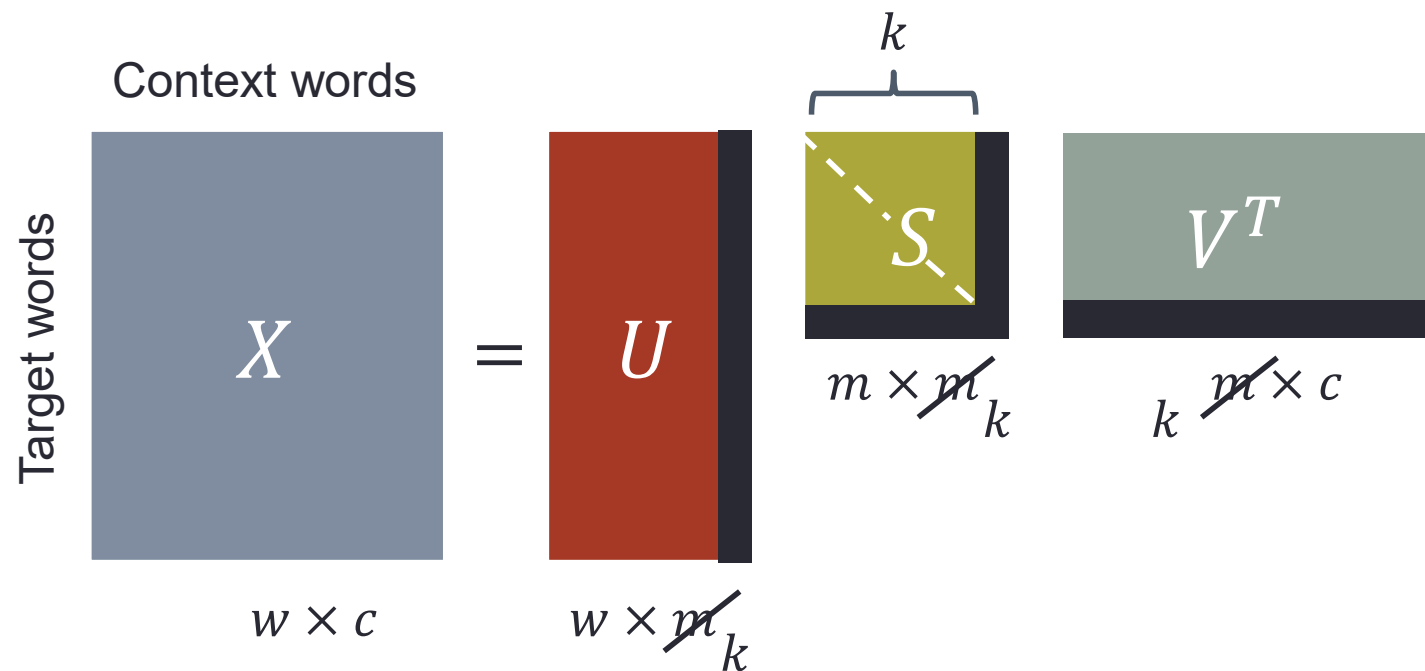
Golub, G. H., & Reinsch, C. (1971). Singular value decomposition and least squares solutions. In Linear Algebra (pp. 134-151). Springer, Berlin, Heidelberg.  
[https://link.springer.com/chapter/10.1007%2F978-3-662-39778-7\\_10](https://link.springer.com/chapter/10.1007%2F978-3-662-39778-7_10)

# Singular Value Decomposition



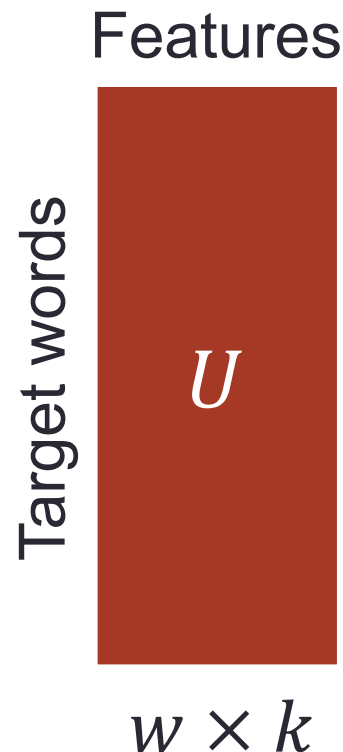
# SVD and Embedding: Latent Semantic Analysis

- If, instead of keeping all  $m$  dimensions, we just keep the **top- $k$**  singular values, we obtain a **low-rank approximation** of the original matrix  $X$ .



# SVD and Embedding: Latent Semantic Analysis

- Instead of multiplying, **we just make use of the matrix  $U$ .**
- In this way, we obtain the following matrix:
- Each row of  $U$ :
  - A  $k$ -dimensional vector,
  - Representing a word in the vocabulary.
- 300 dimensions are commonly used.
  - $k = 300$



# SVD applied to term-context matrix

$$X = \begin{matrix} & & \begin{matrix} I & like & enjoy & deep & learning & NLP & flying \end{matrix} \\ \begin{matrix} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \end{matrix} & \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

$|V|$

# SVD applied to term-context matrix

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} U \\ |V| \times |V| \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix} \\ |V| \times |V| \end{bmatrix} \begin{bmatrix} V^T \\ |V| \times |V| \end{bmatrix}$$

The diagram illustrates the SVD decomposition of a term-context matrix  $X$  of size  $|V| \times |V|$ . The matrix  $X$  is equal to the product of three matrices:  $U$  (size  $|V| \times |V|$ ), a diagonal matrix of singular values (size  $|V| \times |V|$ ), and  $V^T$  (size  $|V| \times |V|$ ). The singular value matrix is shown with its first  $k$  columns and rows highlighted in red, indicating the rank of the matrix. The diagonal elements are  $\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_k, \dots, \sigma_V$ . The matrix  $V^T$  is shown with its first  $k$  columns and rows highlighted in red, indicating the rank of the matrix.

# SVD applied to term-context matrix

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} U \\ |V| \times k \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix} \begin{bmatrix} V^T \\ k \times |V| \end{bmatrix}$$

# SVD applied to term-context matrix

Embedding for  
the word  $w_i$

$$\begin{array}{c} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_i \\ \dots \\ w_{|V|} \end{array} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \end{bmatrix} \begin{matrix} \\ \\ \\ W \\ \\ \\ \end{matrix}$$

$|V| \times k$



# Simple SVD word vectors in Python

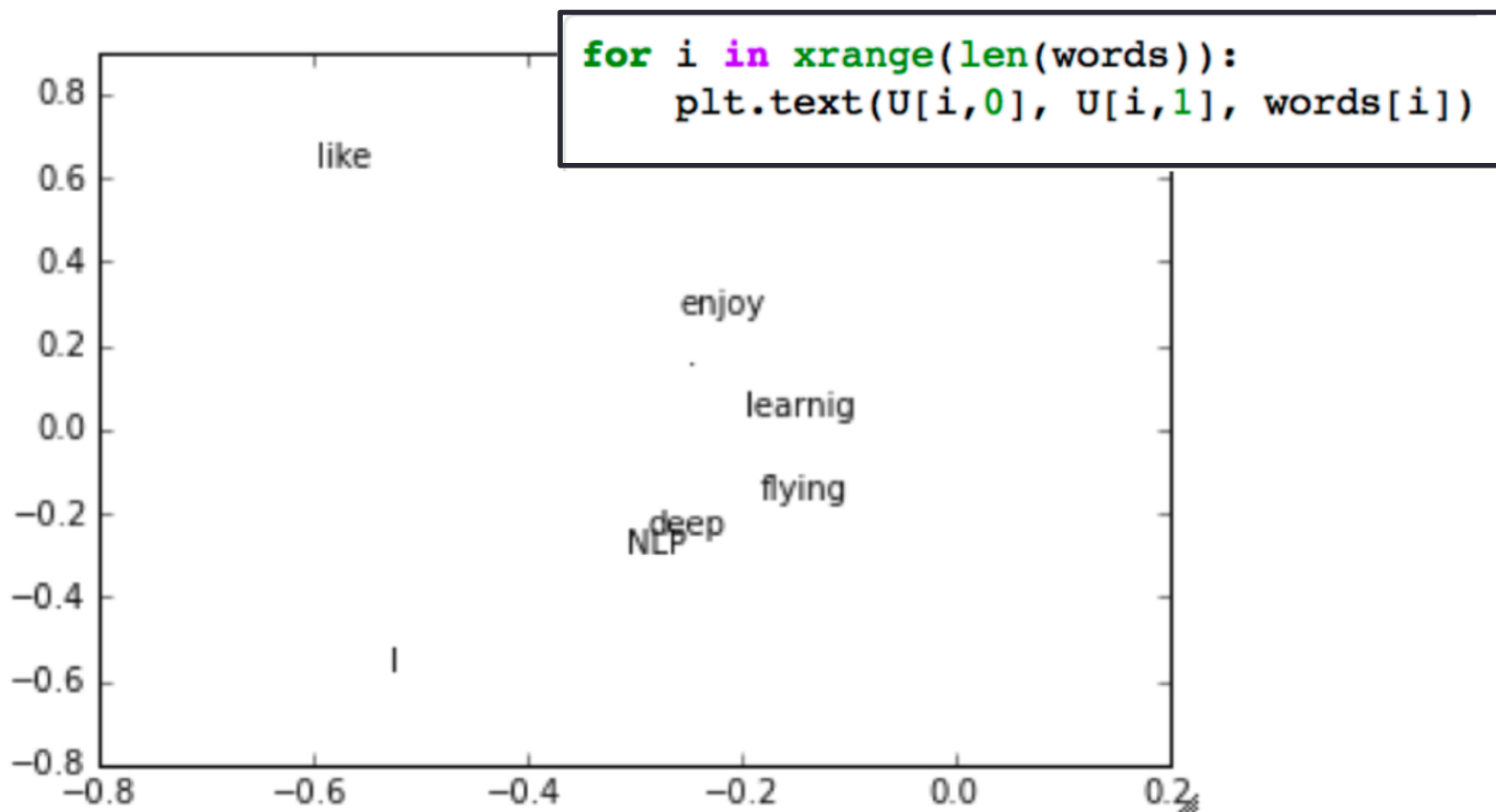
- **Corpus**: I like deep learning. I like NLP. I enjoy flying.

```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep", "learnig", "NLP", "flying", "."]
X = np.array([[0, 2, 1, 0, 0, 0, 0, 0],
              [2, 0, 0, 1, 0, 1, 0, 0],
              [1, 0, 0, 0, 0, 0, 1, 0],
              [0, 1, 0, 0, 1, 0, 0, 0],
              [0, 0, 0, 1, 0, 0, 0, 1],
              [0, 1, 0, 0, 0, 0, 0, 1],
              [0, 0, 1, 0, 0, 0, 0, 1],
              [0, 0, 0, 0, 1, 1, 1, 0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

# Simple SVD word vectors in Python

- Printing first two columns of  $U$  corresponding to the 2 biggest singular values



# Singular Value Decomposition

## Drawbacks:

- The dimensions of the matrix **change very often** (new words are added very frequently and corpus changes in size).
- The matrix can be extremely **sparse** since most words do not co-occur.
- **Quadratic cost** to perform SVD.

# COUNT-BASED MODELS

---

GloVe

# Origins (2014)

## GloVe: Global Vectors for Word Representation

**Jeffrey Pennington, Richard Socher, Christopher D. Manning**

Computer Science Department, Stanford University, Stanford, CA 94305

`jpennin@stanford.edu, richard@socher.org, manning@stanford.edu`

### Abstract

Recent methods for learning vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities using vector arithmetic, but the origin of these regularities has remained opaque. We analyze and make explicit the model properties needed for such regularities to emerge in word vectors. The result is a new global log-

the finer structure of the word vector space by examining not the scalar distance between word vectors, but rather their various dimensions of difference. For example, the analogy “king is to queen as man is to woman” should be encoded in the vector space by the vector equation  $king - queen = man - woman$ . This evaluation scheme favors models that produce dimensions of meaning, thereby capturing the multi-clustering idea of distributed representations (Bengio, 2009).

# Introduction

- The model leverages **statistical information** by **training** only on the non-zero elements in a word-word co-occurrence matrix, rather than:
  - on the entire sparse matrix (e.g., SVD)
  - on individual context windows in a large corpus (e.g., word2vec).
- **Global corpus statistics** are captured directly by the model.

# Basic notation

- $X \rightarrow$  the term-context matrix.
- $X_{ij} \rightarrow$  the frequency of word  $j$  occurring in context of word  $i$ .
- $X_i = \sum_k X_{ik} \rightarrow$  the global frequency of any word appearing in the context of word  $i$ .
- $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} \rightarrow$  probability that word  $j$  appears in the context of word  $i \rightarrow$  **co-occurrence probability**

# Example

- Can **certain aspects of meaning** be extracted directly from co-occurrence probabilities?
- Consider two words  $i$  and  $j$  that exhibit a particular aspect of interest; for concreteness, suppose we are interested in the concept of **thermodynamic phase**, for which we might take  $i = \textit{ice}$  and  $j = \textit{steam}$ .
- The **relationship of these words** can be examined by studying the **ratio** of their co-occurrence probabilities with various “probe” words (i.e., context words),  $k$ .



# Example

- For words  $k$  related to  $i = \text{ice}$  but not  $j = \text{steam}$ , say  $k = \text{solid}$ , the ratio  $\frac{P_{ik}}{P_{jk}}$  **should be large**.
- Similarly, for words  $k$  related to  $j = \text{steam}$  but not  $i = \text{ice}$ , say  $k = \text{gas}$ , the ratio  $\frac{P_{ik}}{P_{jk}}$  **should be small**.
- For words  $k$  like *water* or *fashion*, that are either related to both  $i = \text{ice}$  and  $j = \text{steam}$ , or to neither, the ratio  $\frac{P_{ik}}{P_{jk}}$  **should be close to “1”**.

# Meaning extraction

- **Co-occurrence probabilities** for target words *ice* and *steam* with selected context words from a 6 billion token corpus.

Probability and Ratio	$k = \text{solid}$	$k = \text{gas}$	$k = \text{water}$	$k = \text{fashion}$
$P(k \text{ice})$	$1.9 \times 10^{-4}$	$6.6 \times 10^{-5}$	$3.0 \times 10^{-3}$	$1.7 \times 10^{-5}$
$P(k \text{steam})$	$2.2 \times 10^{-5}$	$7.8 \times 10^{-4}$	$2.2 \times 10^{-3}$	$1.8 \times 10^{-5}$
$P(k \text{ice})/P(k \text{steam})$	8.9	$8.5 \times 10^{-2}$	1.36	0.96

- Only in the ratio capture **non-discriminative** words like *water* and *fashion*, because:
  - **large values** (much greater than 1) correlate well with properties specific to *ice*.
  - **small values** (much less than 1) correlate well with properties specific of *steam*.

# The GloVe model

- Starting point for word vector learning?
- **Co-occurrence probabilities ratios** instead of probabilities themselves.
- Co-occurrence probabilities ratios **capture relevant information** about words' relationships.
  - These ratios capture the distinguishing information between word meanings — they tell us **how much more likely one context is for one word than another**.
  - If we only used raw probabilities  $P_{ij}$  we'd lose that comparative semantic signal → some words (like “the”, “and”) have high probabilities for almost all words, making them uninformative.

# The GloVe model

- The ratio  $\frac{P_{ik}}{P_{jk}}$  depends on three words  $i$ ,  $j$ , and  $k$
- The **most general model** takes the form:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

where  $w \in \mathbb{R}^d$  are word vectors and  $\tilde{w} \in \mathbb{R}^d$  are separate context word vectors.

# The GloVe model

- The GloVe model **constructs this  $F$  function** to learn word vectors representation.
- After a series of steps, which we omit, a **simplification** over  $F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$  is as follows:

$$w_i^T \tilde{w}_k + (b_i) + (\tilde{b}_k) = \log(X_{ik}) \quad (*)$$

some parameters to be selected

We are interested in these vectors!

# The GloVe model

- Then, GloVe builds an **objective function  $J$**  that associates word vectors to text statistics.
  - **Least squares regression model.**
- Cast the equation (\*) as a **least squares regression model.**

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543)

# Pharenteses: Least squares regression

- A least squares regression model, often referred to as **linear regression**, is a statistical approach used to:

*Model the relationship between a dependent variable and one or more independent variables by finding the best-fitting linear equation.*

- The “least squares” part of the name refers to the method used to **estimate the parameters** of the linear equation by:

*Minimizing the sum of the squared differences between the observed values and the values predicted by the model.*

# The GloVe model

- Then, GloVe builds an **objective function**  $J$  that associates word vectors to text statistics.
  - Least squares regression model.
- Cast the equation (\*) as a **least squares regression problem** with a **weighting function**  $f(X_{ik})$ .

$$J = \sum_{i,k=1}^V f(X_{ik}) (w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log(X_{ik}))^2$$

where  $V$  is the size of the vocabulary.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543)



# The GloVe model – Simplification

$$J = \frac{1}{2} \sum_{i,k=1}^V f(X_{ik}) (w_i^T \tilde{w}_k - \log(X_{ik}))^2$$

- We end up with  $U$  and  $V$  from all the vectors  $u = w$  and  $v = \tilde{w}$
- What to do with the two sets of vectors?
  - Both capture similar co-occurrence information. It turns out, the best solution is to simply sum them up (one of many hyperparameters explored in GloVe):

$$X_{final} = U + V$$

# PREDICTIVE MODELS

---

word2vec and FastText

# Origins (2013)

---

## Efficient Estimation of Word Representations in Vector Space

---

**Tomas Mikolov**

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

**Kai Chen**

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

**Greg Corrado**

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

**Jeffrey Dean**

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

### Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

# Origins (2013)

---

## Distributed Representations of Words and Phrases and their Compositionality

---

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

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

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

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

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

### Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

# word2vec

- This technique provides a tool to create collections of similar concepts automatically, on **raw texts** and without advanced language skills on the part of the user.
- Raw texts are used as **implicitly supervised training data**.
  - No need for hand-labeled supervision.
  - Not all the “traditional” pre-processing steps performed with the BoW and TF-IDF representations are necessary → Next slides.

# word2vec

- The largest the training set, the better the performance
  - Very good performances are obtained by employing **very large texts** in the learning phase (> 10M of words).
- The texts should include as **many different words as possible**.
- Code **available** on the Web:
  - <https://code.google.com/archive/p/word2vec/>
  - <https://www.tensorflow.org/text/tutorials/word2vec>
  - <https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial>

# Main idea (neural network word embeddings)

- Similar to language modeling but predicting **context**, rather than next word.

$$P(context|word) \quad \leftarrow \text{maximize}$$

- In practice:

$$J = 1 - P(context|word) \quad \leftarrow \text{minimize}$$

- We **adjust the vector representations of words** to minimize the loss.


# Directly learning low-dimensional vectors

## Relevant literature:

- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). **Learning representations by back-propagating errors**. *Cognitive modeling*, 5(3).
- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). **A neural probabilistic language model**. *Journal of machine learning research*, 3(Feb), 1137-1155.
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). **Natural language processing (almost) from scratch**. *Journal of machine learning research*, 12(Aug), 2493-2537.
- The **word2vec papers** illustrated before and explained in the next slides.



# Two basic architectures

- There are **two architectures** used by word2vec:
  - **Skip-gram**
  - Continuous bag-of-words (**CBOW**) Algorithms for producing word vectors
- Two (moderately efficient) training methods:
  - **Softmax**
  - **Negative sampling**

# Softmax and negative sampling

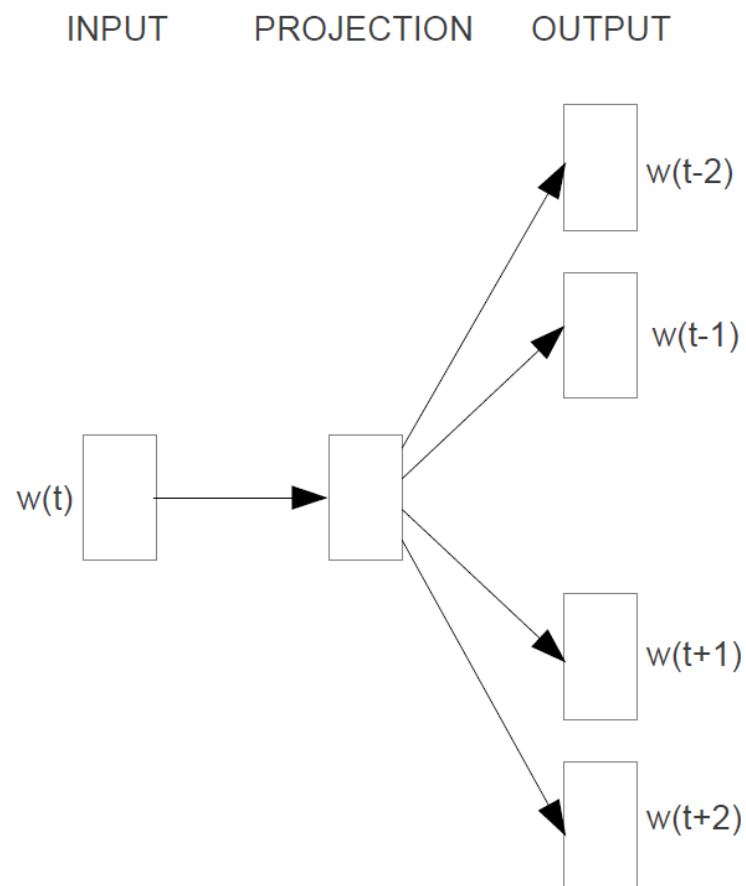
## Softmax

- A function used, in the context of word2vec and word embedding, to predict the context words (or target words) for a given input word → all the vocabulary.
- Softmax “bottleneck”.

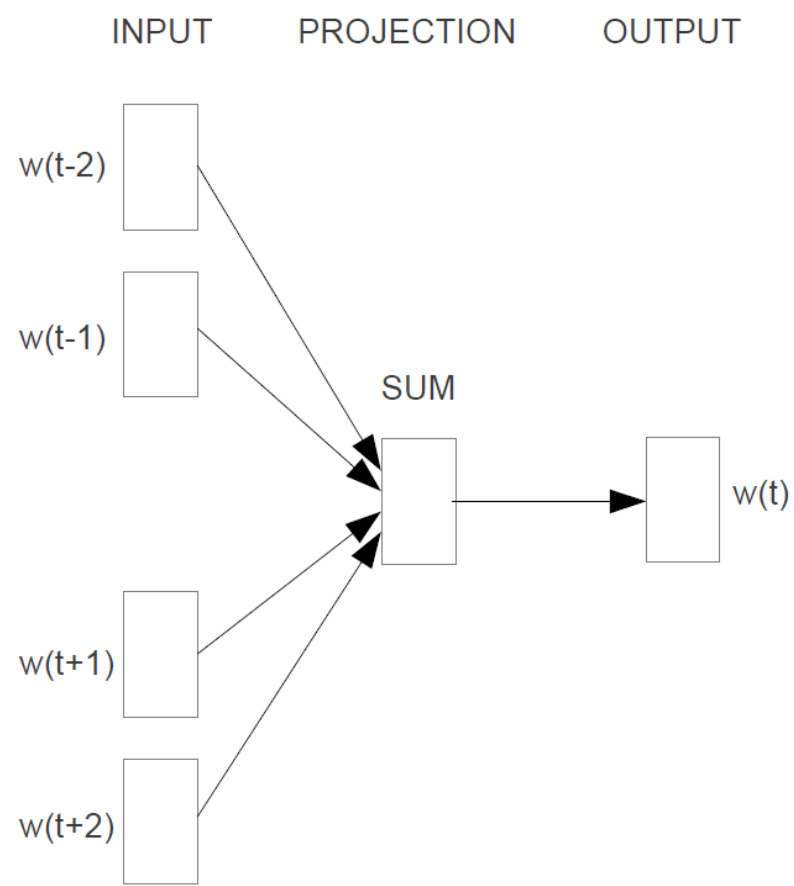
## Negative sampling

- A technique introduced to address the computational inefficiency of softmax in training word embeddings.
- Instead of predicting the entire vocabulary, select a small number of negative samples (typically a few dozen) and the true context words.
  - The negative examples are words that do not appear in the context of the target word.
- The model is trained to assign higher probabilities to the true context words and lower probabilities to the negative samples.

# word2vec Architecture



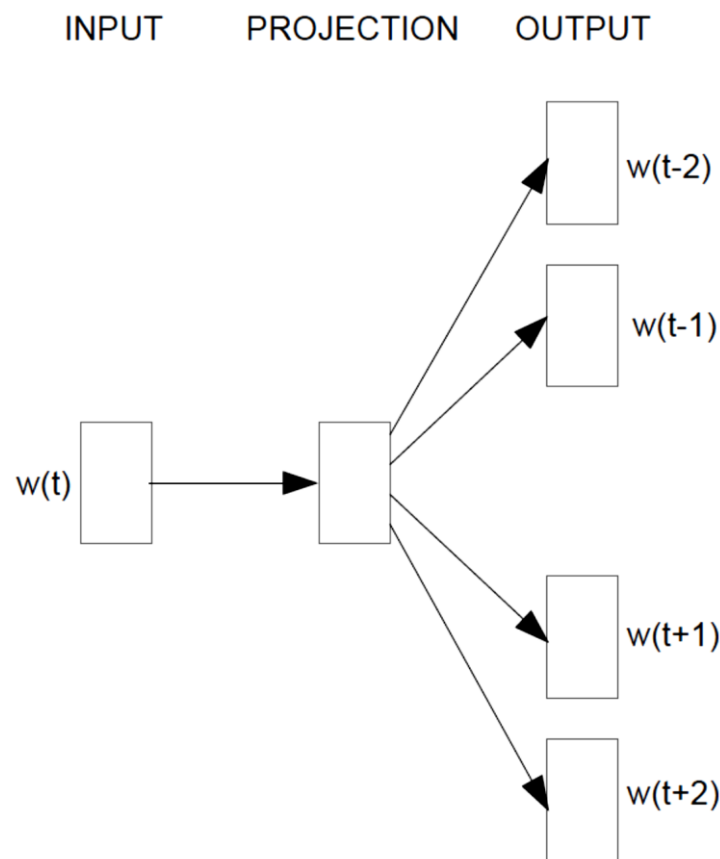
**Skip-gram**



**CBOW**

# Skip-gram model

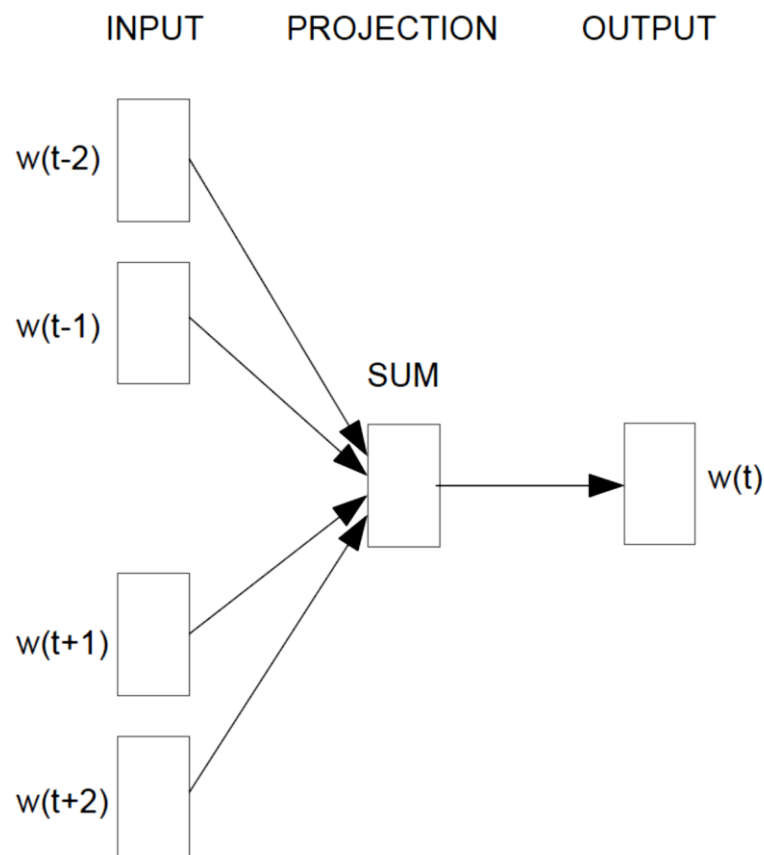
- **Predict the surrounding words** (context words), based on the current word (the center word).
- Mikolov et. al. 2013. Efficient Estimation of Word Representations in Vector Space.



**Skip-gram**

# CBOW model

- **Predict the current word** (the center word) based on the surrounding words (context words).
- Mikolov et. al. 2013. Efficient Estimation of Word Representations in Vector Space.



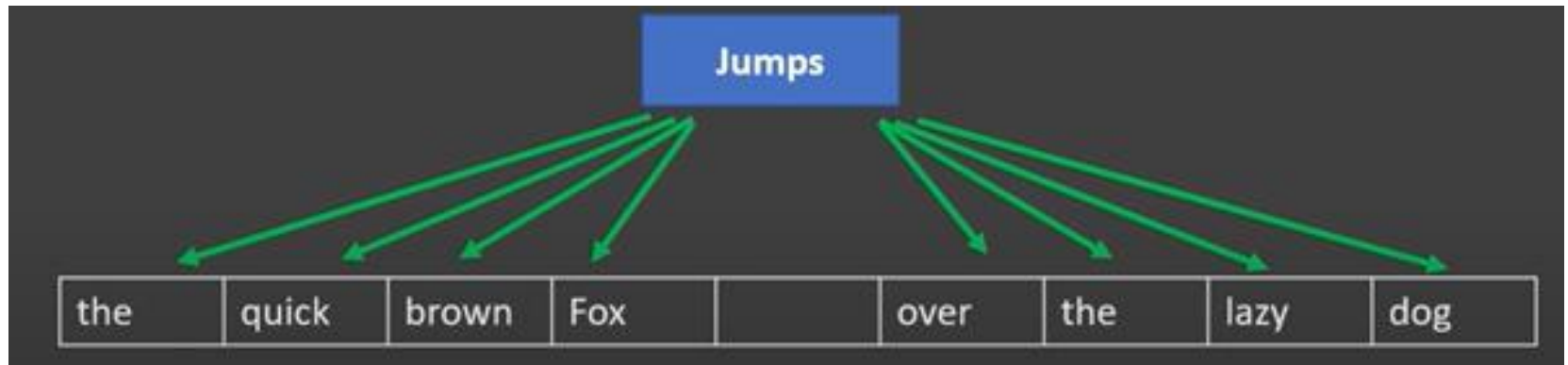
**CBOW**

# The skip-gram model

---

# Skip-gram

- It **predicts context words** from the target word.



# The model (1)

- Given a sliding window of a fixed size moving along a sentence:
  - the word in the middle is the “target”;
  - those on its left and right within the sliding window are the context words.



## The model (2)

Corpus

Training Data

The	man
-----	-----

 is here to → (the, man)

The	man	is
-----	-----	----

 here to → (man, the), (man, is)

The 

man	is	here
-----	----	------

 to → (is, man), (is, here)

The man 

is	here	to
----	------	----

 → (here, is), (here, to)

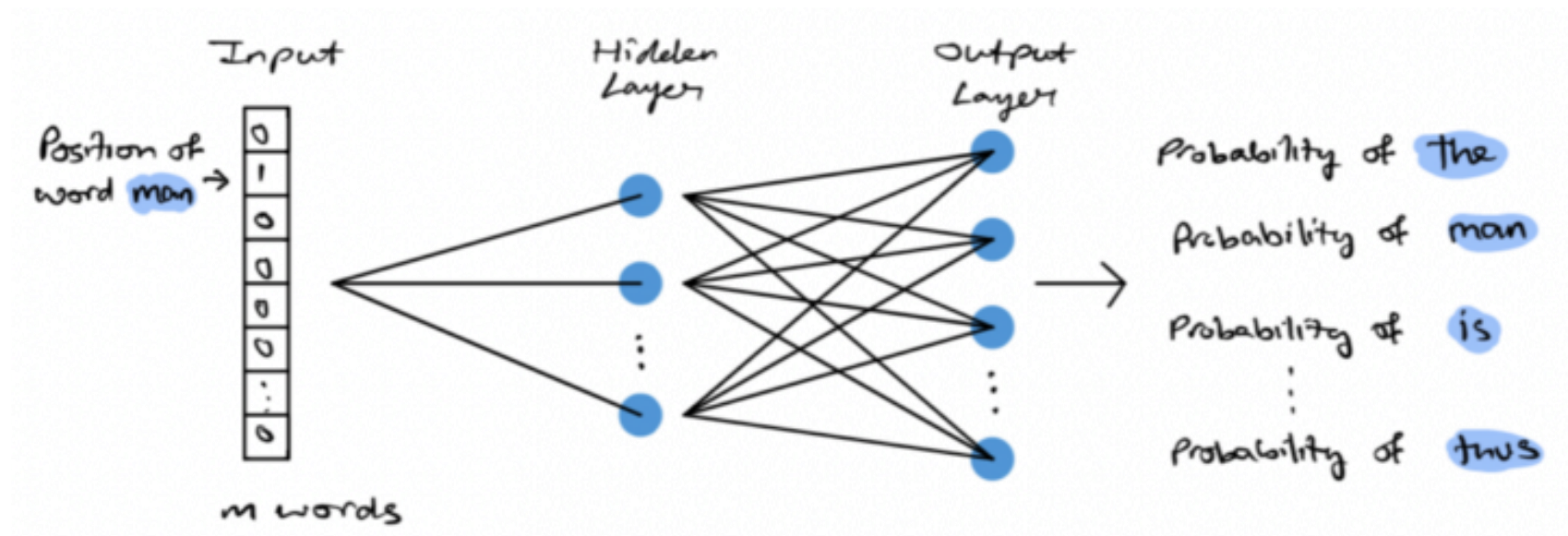
is	here	to
----	------	----

Window size

## The model (3)

- Given a sliding window of a fixed size moving along a sentence:
  - the word in the middle is the “target”;
  - those on its left and right within the sliding window are the context words.
- The skip-gram model **is trained to predict** the **probabilities** of a word being a context word for the given target.

# The model



- The hidden layer **is the word embedding** of size  $N$ .

# The CBOW model

---

# CBOW

- The **Continuous Bag-of-Words (CBOW)** is another similar model for learning word vectors.
- It **predicts the target word** from source context words.

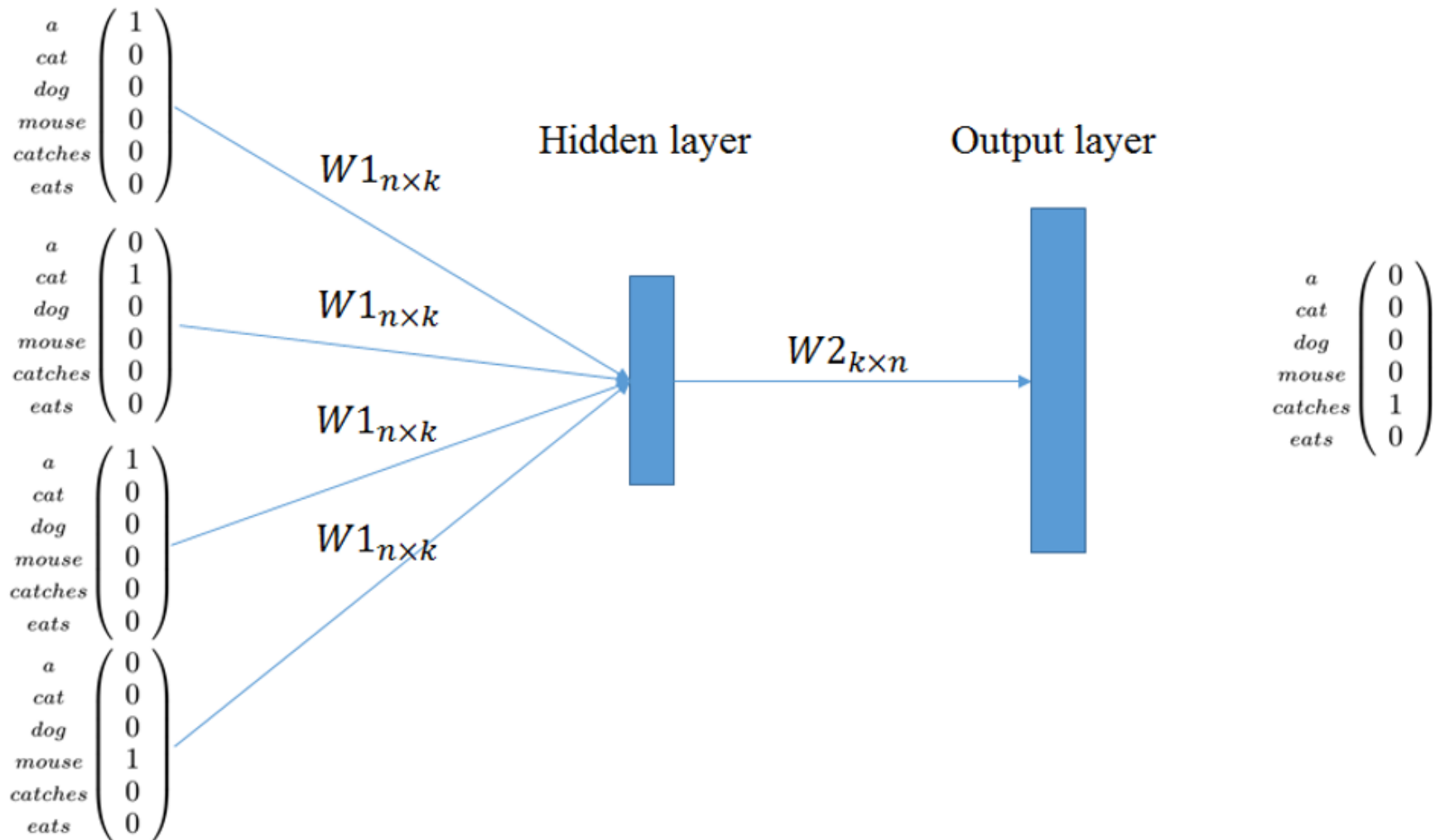


# The model (1)

## Positive Training Samples

Sentence	CBOW						
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	((quick, brown), the)			
The	quick	brown					
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	The	quick	brown	fox	((the, brown, fox), quick)		
The	quick	brown	fox				
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	The	quick	brown	fox	jumps	((the, quick, fox, jumps), brown)	
The	quick	brown	fox	jumps			
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	The	quick	brown	fox	jumps	over	((quick, brown, jumps, over), fox)
The	quick	brown	fox	jumps	over		

# The model (2)



# Skip-gram VS CBOW

- **Skip-gram**: works well also with a smaller amount of the training data, represents well even **rare words** or phrases.
- **CBOW**: several times faster to train than the skip-gram, slightly better accuracy for the **frequent words**.



# FastText (Origins, 2017)

## Enriching Word Vectors with Subword Information

Piotr Bojanowski\* and Edouard Grave\* and Armand Joulin and Tomas Mikolov

Facebook AI Research

{bojanowski, egrave, ajoulin, tmikolov}@fb.com

### Abstract

Continuous word representations, trained on large unlabeled corpora are useful for many natural language processing tasks. Popular models that learn such representations ignore the morphology of words, by assigning a distinct vector to each word. This is a limitation, especially for languages with large vocabularies and many rare words. In this paper, we propose a new approach based on the skipgram model, where each word is represented as a bag of character  $n$ -grams. A vector representation is associated to each character  $n$ -gram; words being represented as the sum of these representations. Our method is fast, allowing to train models on large corpora quickly and allows us to compute word representations for words that did not appear in the training data. We evaluate our word representations on

et al., 2010; Baroni and Lenci, 2010). In the neural network community, Collobert and Weston (2008) proposed to learn word embeddings using a feed-forward neural network, by predicting a word based on the two words on the left and two words on the right. More recently, Mikolov et al. (2013b) proposed simple log-bilinear models to learn continuous representations of words on very large corpora efficiently.

Most of these techniques represent each word of the vocabulary by a distinct vector, without parameter sharing. In particular, they ignore the internal structure of words, which is an important limitation for morphologically rich languages, such as Turkish or Finnish. For example, in French or Spanish, most verbs have more than forty different inflected forms. while the Finnish language has fifteen cases

# FastText: Main characteristics

- **Subword Embeddings**

- It breaks words down into smaller character ***n*-grams** (subwords) and learns embeddings for these subwords.
- This allows FastText to capture **morphological** and **syntactic information**, making it effective for handling out-of-vocabulary words and languages with rich morphology.

- **Efficiency**

- Designed for **efficient training** and **inference**.
- Its subword modeling **reduces the dimensionality** of the embedding space.
  - It stores embeddings for subwords and composes word embeddings from these subword representations.
- This can lead to significant memory savings, especially when dealing with large vocabularies, making it more memory and computationally efficient compared to some other embedding models.

# From WORD vectors to DOCUMENT vectors

- At this point we have **word embeddings**, such as those generated by methods like Singular Value Decomposition (SVD), GloVe, Word2Vec, or FastText.
- What about **DOCUMENT representation**?
- We can use word embeddings to represent a document by **combining the embeddings** of the individual words in the document.

# Average word embeddings

- Calculate the **mean** (average) of the word embeddings for all words in the document.
- This is a simple and effective method.
- It captures the overall semantic content of the document but **may lose some word order information**.

# Sum of word embeddings

- **Sum** the word embeddings for all words in the document.
- This representation also captures the semantics of the document but **might be sensitive to the length of the document**.

# Weighted average (TF-IDF-based)

- Use **TF-IDF** to assign weights to each word in the document.
- Calculate the weighted average of word embeddings based on the TF-IDF scores.
- This method gives **more importance to important or unique words** in the document.

# Doc2Vec (Paragraph Vector)

- Doc2Vec is an **extension of Word2Vec** that learns document embeddings.
- It can represent a document as a fixed-length vector, making it particularly useful when you want to capture the overall context of the document.
- It will be illustrated during the labs.
  - <https://radimrehurek.com/gensim/models/doc2vec.html>

# ISSUES AND SOLUTIONS

---

Towards Contextualized Word Embeddings



# Main Issues

- **Context independence**

- “Traditional” word embeddings are “context-independent”, which means that each word is represented by a **single static vector**.
- This fails to capture the various meanings of a word in different contexts.
  - For example, "bank" can refer to a financial institution or the side of a river, but a traditional embedding represents it with a single vector.

- **Out-of-Vocabulary (OOV) words**

- Traditional embeddings cannot handle out-of-vocabulary words, as they are limited to the words present in the training data.
  - In contrast, models like FastText, which use subword representations, can handle such words.

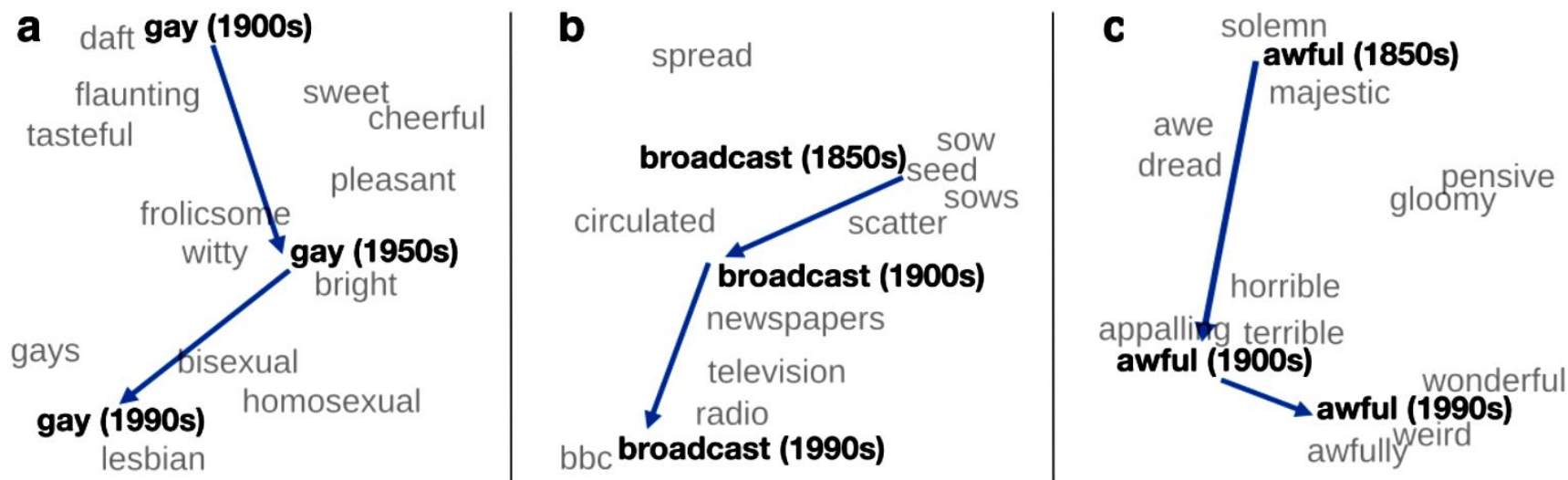
- **Lack of transparency**

- Traditional embeddings are not always transparent, and it can be difficult to interpret the meaning of individual dimensions or vectors.

**AND...**

# Changes in meaning...

- The word **gay** shifted from meaning “cheerful” or “frolicsome” to referring to homosexuality.
- In the early 20th century **broadcast** referred to “casting out seeds”; with the rise of television and radio its meaning shifted to “transmitting signals”.
- **Awful** underwent a process of pejoration, as it shifted from meaning “full of awe” to meaning “terrible or appalling”.



# Bias over time...

---

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

---

Tolga Bolukbasi<sup>1</sup>, Kai-Wei Chang<sup>2</sup>, James Zou<sup>2</sup>, Venkatesh Saligrama<sup>1,2</sup>, Adam Kalai<sup>2</sup>

<sup>1</sup>Boston University, 8 Saint Mary's Street, Boston, MA

<sup>2</sup>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. This raises concerns because their widespread use, as we describe, often tends to amplify these biases. Geometrically, gender bias is first shown to be captured by a direction in the word embedding. Second, gender neutral words are shown to be linearly separable from gender definition words in the word embedding. Using these properties, we provide a methodology for modifying an embedding to remove gender stereotypes, such as the association between the words *receptionist* and *female*, while maintaining desired associations such as between the words *queen* and *female*. Using crowd-worker evaluation as well as standard benchmarks, we empirically demonstrate that our algorithms significantly reduce gender bias in embeddings while preserving its useful properties such as the ability to cluster related concepts and to solve analogy tasks. The resulting embeddings can be used in applications without amplifying gender bias.

# Bias over time...

However, the embeddings also pinpoint sexism implicit in text. For instance, it is also the case that:

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{computer programmer}} - \overrightarrow{\text{homemaker}}.$$

In other words, the same system that solved the above reasonable analogies will offensively answer “man is to computer programmer as woman is to  $x$ ” with  $x=\text{homemaker}$ .

Extreme <i>she</i>		Extreme <i>he</i>		Gender stereotype <i>she-he</i> analogies	
1. homemaker	1. maestro	sewing-carpentry	registered nurse-physician	housewife-shopkeeper	
2. nurse	2. skipper	nurse-surgeon	interior designer-architect	softball-baseball	
3. receptionist	3. protege	blond-burly	feminism-conservatism	cosmetics-pharmaceuticals	
4. librarian	4. philosopher	giggle-chuckle	vocalist-guitarist	petite-lanky	
5. socialite	5. captain	sassy-snappy	diva-superstar	charming-affable	
6. hairdresser	6. architect	volleyball-football	cupcakes-pizzas	lovely-brilliant	
7. nanny	7. financier				
8. bookkeeper	8. warrior				
9. stylist	9. broadcaster	queen-king	sister-brother	Gender appropriate <i>she-he</i> analogies	
10. housekeeper	10. magician	waitress-waiter	ovarian cancer-prostate cancer	mother-father	
				convent-monastery	



# Bias over time...

<https://www.pnas.org/content/pnas/115/16/E3635.full.pdf>



## Word embeddings quantify 100 years of gender and ethnic stereotypes

Nikhil Garg<sup>a,1</sup>, Londa Schiebinger<sup>b</sup>, Dan Jurafsky<sup>c,d</sup>, and James Zou<sup>e,f,1</sup>

<sup>a</sup>Department of Electrical Engineering, Stanford University, Stanford, CA 94305; <sup>b</sup>Department of History, Stanford University, Stanford, CA 94305;

<sup>c</sup>Department of Linguistics, Stanford University, Stanford, CA 94305; <sup>d</sup>Department of Computer Science, Stanford University, Stanford, CA 94305;

<sup>e</sup>Department of Biomedical Data Science, Stanford University, Stanford, CA 94305; and <sup>f</sup>Chan Zuckerberg Biohub, San Francisco, CA 94158

Edited by Susan T. Fiske, Princeton University, Princeton, NJ, and approved March 12, 2018 (received for review November 22, 2017)

**Word embeddings are a powerful machine-learning framework that represents each English word by a vector. The geometric relationship between these vectors captures meaningful semantic relationships between the corresponding words. In this paper, we develop a framework to demonstrate how the temporal dynamics of the embedding helps to quantify changes in stereotypes and attitudes toward women and ethnic minorities in the 20th and 21st centuries in the United States. We integrate word embeddings trained on 100 y of text data with the US Census to show that changes in the embedding track closely with demographic and occupation shifts over time. The embedding captures societal shifts—e.g., the women’s movement in the 1960s and Asian immigration into the United States—and also illuminates how specific adjectives and occupations became more closely associated with certain populations over time. Our framework for temporal analysis of word embedding opens up a fruitful intersection between machine learning and quantitative social science.**

in the large corpora of training texts (20–23). For example, the vector for the adjective honorable would be close to the vector for man, whereas the vector for submissive would be closer to woman. These stereotypes are automatically learned by the embedding algorithm and could be problematic if the embedding is then used for sensitive applications such as search rankings, product recommendations, or translations. An important direction of research is to develop algorithms to debias the word embeddings (20).

In this paper, we take another approach. We use the word embeddings as a quantitative lens through which to study historical trends—specifically trends in the gender and ethnic stereotypes in the 20th and 21st centuries in the United States. We develop a systematic framework and metrics to analyze word embeddings trained over 100 y of text corpora. We show that temporal dynamics of the word embedding capture changes in gender and ethnic stereotypes over time. In particular, we quantify how specific biases decrease over time while other stereotypes increase. Moreover, dynamics of the embedding strongly

# Current trends in word embedding

- **Contextualized word embeddings:** a different embedding depending on context):
  - The nail hit the **beam** behind the wall.
  - They reflected a **beam** off the moon.
- **Tackling changes in meaning.**
- **Tackling bias over time.**

beam

1.

<b>n</b>	long thick piece of wood or metal or concrete, etc., used in construction
Types:	<a href="#">show 23 types...</a>
Type of:	<a href="#">piece</a> a separate part of a whole
	<a href="#">structural member</a> support that is a constituent part of any structure or building
<b>n</b>	a gymnastic apparatus used by women gymnasts
Synonyms:	<a href="#">balance beam</a>
Type of:	<a href="#">exerciser</a> , <a href="#">gymnastic apparatus</a> sports equipment used in gymnastic exercises

2.

<b>n</b>	a column of light (as from a beacon)
Synonyms:	<a href="#">beam of light</a> , <a href="#">irradiation</a> , <a href="#">light beam</a> , <a href="#">ray</a> , <a href="#">ray of light</a> , <a href="#">shaft</a> , <a href="#">shaft of light</a>
Types:	<a href="#">show 7 types...</a>
Type of:	<a href="#">light</a> , <a href="#">visible light</a> , <a href="#">visible radiation</a> (physics) electromagnetic radiation that can produce a visual sensation
<b>n</b>	a group of nearly parallel lines of electromagnetic radiation