PCA, Ctd. and Text and Natural Language

https://tinyurl.com/cis5190-11-2-2022

Osbert Bastani and Zachary G. Ives

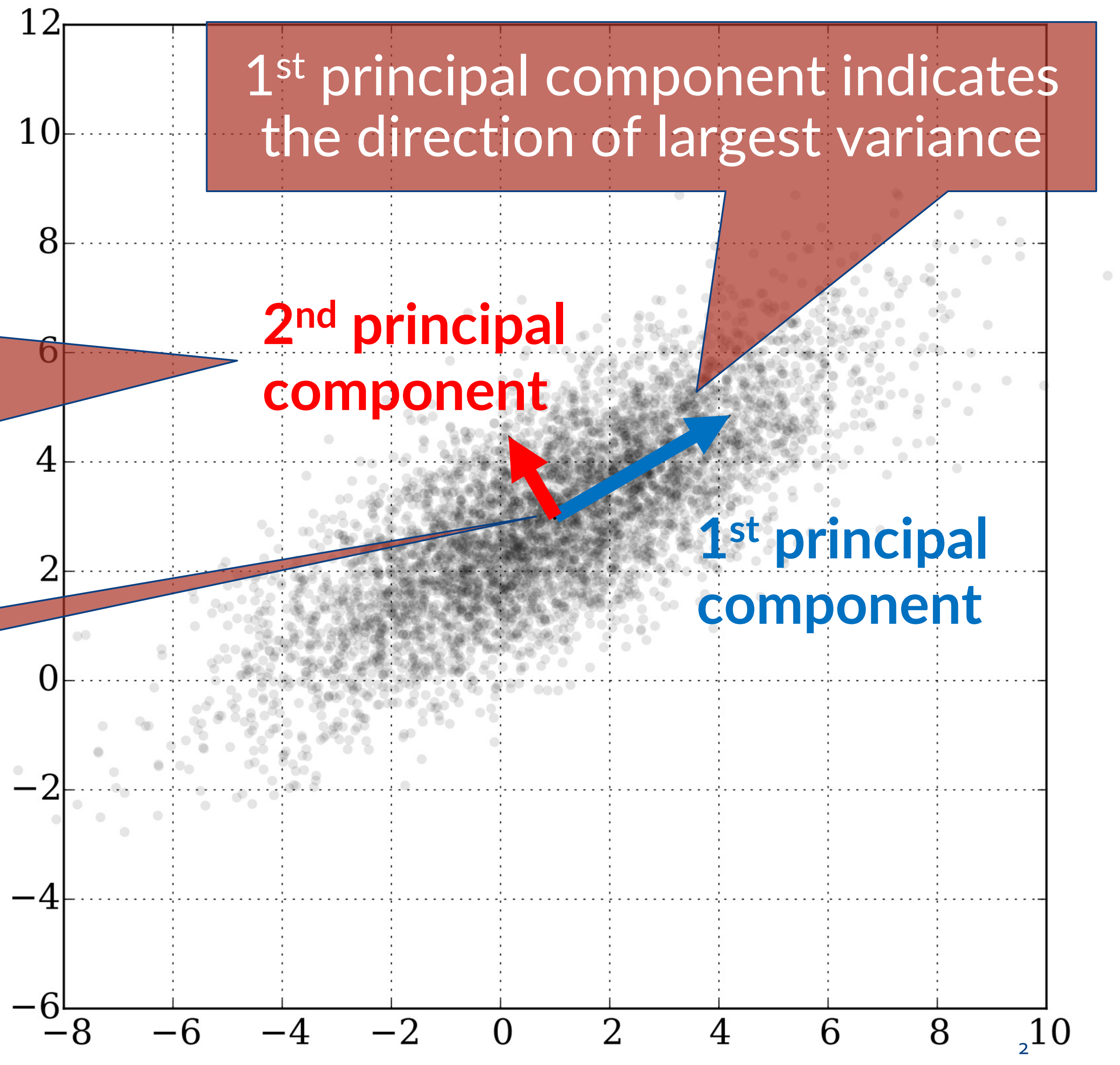
CIS 4190/5190 – Fall 2022

PCA on a 2D Gaussian Dataset

Each subsequent principal component:

- is orthogonal to all previous components
- indicates the direction of largest variance of the residuals

New axes (basis vectors) originate from the mean



PCA Algorithm

Given data $\{\boldsymbol{x}_1, \ldots \boldsymbol{x}_n\}$, compute covariance matrix C

- $\boldsymbol{X}$ is the $N \times D$ data matrix
- Compute data mean (average over all rows of $\boldsymbol{X}$)
- Subtract mean from each row of $\boldsymbol{X}$ (centering the data)
- Compute covariance matrix C $= \frac{\boldsymbol{X}^T\boldsymbol{X}}{n-1}$ (C is $D \times D$)

PCA basis vectors (new coordinate axes) are given by the eigenvectors of C

- Q, Λ = numpy.linalg.eig(C)
- $\{\boldsymbol{q}_d, \lambda_d\}_{d=1,\ldots,D}$ are the eigenvectors/eigenvalues of C

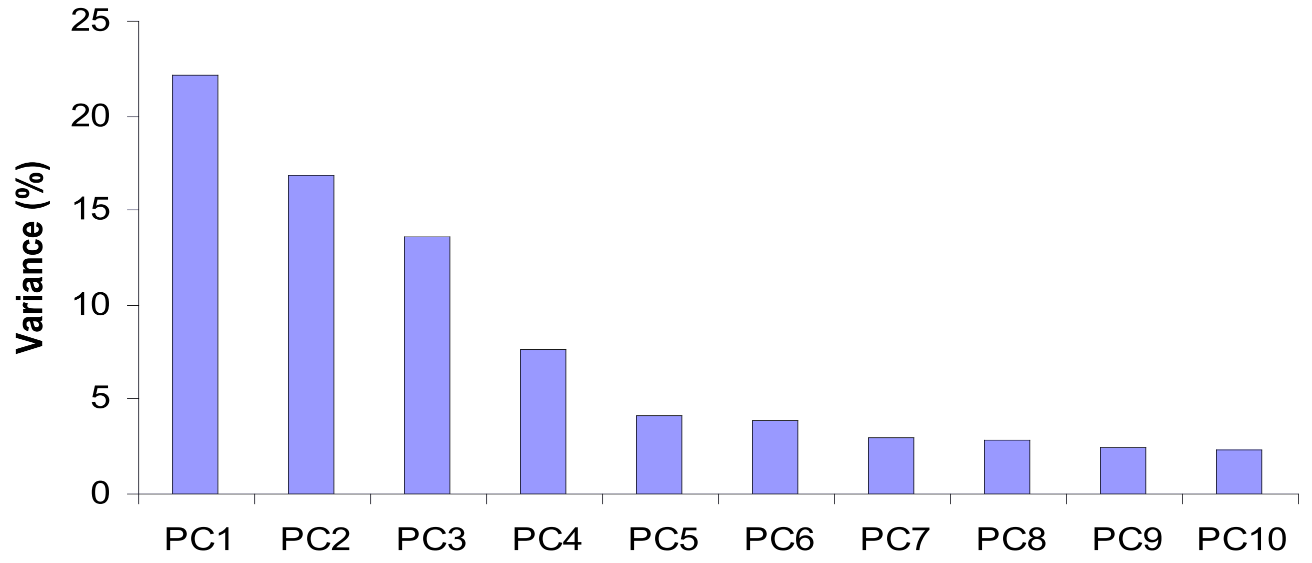
  $(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D)$

But there are *D* eigenvectors, so where is the dimensionality reduction?

- A: Larger eigenvalue $\Rightarrow$ "more important" eigenvectors

Dimensionality Reduction

- Can *ignore* the components of lesser significance



- You do lose some information, but if the eigenvalues are small, you don't lose much
  - choose only the first D' eigenvectors, based on their eigenvalues
  - final data set has only D' dimensions

Based on slide by Barnabás Póczos, UAlberta

Recap so far

- Want to reconstruct data approximately in a new coordinate space
- Must find axes of this coordinate space, because the weights on those axes are just projections
- Objective: axes with lowest reconstruction error
  - Same as axes with high variance projections
- Solution straight from linear algebra. Axes are eigenvectors of covariance matrix

- We now have the basic process – let's see how we use PCA to reduce dimensionality!

PCA

Each row of $\boldsymbol{Q}$ corresponds to a feature; keep only first D' columns of $\boldsymbol{Q}$

$$X = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & \cdots \\ 1 & 1 & 0 & 1 & 1 & \cdots \\ 0 & 0 & 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ 1 & 0 & 1 & 0 & 1 & \cdots \end{bmatrix}$$

$\boldsymbol{X}$ has D columns

$\boldsymbol{Q}$ is $D \times D$

$$Q = \begin{array}{cccccc} \begin{bmatrix} 0.34 & 0.23 & \cancel{-0.30} & \cancel{-0.23} & \cancel{\cdots} \\ 0.04 & 0.13 & \cancel{-0.40} & \cancel{0.21} & \cancel{\cdots} \\ -0.64 & 0.93 & \cancel{0.61} & \cancel{0.28} & \cancel{\cdots} \\ \vdots & \vdots & \cancel{\vdots} & \cancel{\vdots} & \cancel{\ddots} \\ -0.20 & -0.83 & \cancel{0.78} & \cancel{-0.93} & \cancel{\cdots} \end{bmatrix} \\ \boldsymbol{e}_1 \quad \boldsymbol{e}_2 \quad \cdots \quad \boldsymbol{e}_D \end{array}$$

$\boldsymbol{Q}$ is the eigenvectors of $\mathrm{C} = \frac{\mathrm{XX}^{\mathrm{T}}}{n-1}$; columns are ordered by importance (highest eigenvalues first)

PCA

- Each column of $\boldsymbol{Q}$ gives weights for a **linear combination** of the **original features**

$$Q = \begin{bmatrix} 0.34 & 0.23 & \cancel{-0.30} & \cancel{-0.23} & \cancel{\cdots} \\ 0.04 & 0.13 & \cancel{-0.40} & \cancel{0.21} & \cancel{\cdots} \\ -0.64 & 0.93 & \cancel{0.61} & \cancel{0.28} & \cancel{\cdots} \\ \vdots & \vdots & \cancel{\vdots} & \cancel{\vdots} & \cancel{\ddots} \\ -0.20 & -0.83 & \cancel{0.78} & \cancel{-0.93} & \cancel{\cdots} \end{bmatrix}$$

$$= 0.34 \times feature_1 + 0.04 \times feature_2 - 0.64 \times feature_3 + \cdots$$

PCA

Compute $\boldsymbol{x}.\boldsymbol{e}_d$ to get the new representation for each instance $\boldsymbol{x}$

$$X = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & \cdots \\ 1 & 1 & 0 & 1 & 1 & \cdots \\ 0 & 0 & 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ 1 & 0 & 1 & 0 & 1 & \cdots \end{bmatrix} \begin{array}{l} \\ \\ \boldsymbol{x}_3 \\ \\ \\ \end{array} \qquad \hat{Q} = \begin{bmatrix} 0.34 & 0.23 \\ 0.04 & 0.13 \\ -0.64 & 0.93 \\ \vdots & \vdots \\ -0.20 & -0.83 \end{bmatrix}$$

The new 2D representation for $\boldsymbol{x}_3$ is given by $[\widehat{x_{31}} = \boldsymbol{x}_3.\boldsymbol{e}_1,\ \widehat{x_{32}} = \boldsymbol{x}_3.\boldsymbol{e}_2]$:

$$\widehat{x_{31}} = 0.34(0) + 0.04(0) - 0.64(1) + \cdots$$
$$\widehat{x_{32}} = 0.23(0) + 0.13(0) + 0.93(1) + \cdots$$

The re-projected data matrix can be conveniently computed as $\hat{X} = X\hat{Q}$

Using PCA for Feature Reduction / Compression

We could use the PCA transformation of the data instead of the original features

Keep only $D' < D$ PCA features

PCA tries to retain most of the variance in the data

- So, we're reducing the dataset to features that retain meaningful variations of the data set

We may want to project "back" to the original feature space! Multiply by the transpose of the transformation matrix, and add back the mean:

$$\hat{X}\hat{Q}^T + \bar{X}$$

Eigenfaces

What happens when you compute the principal components of face images?

(1000 64×64 images)

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

What happens when you compute the principal components of face images?

"Eigenfaces": main directions of deviation from the mean face

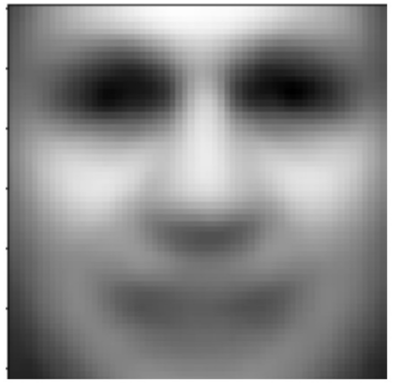
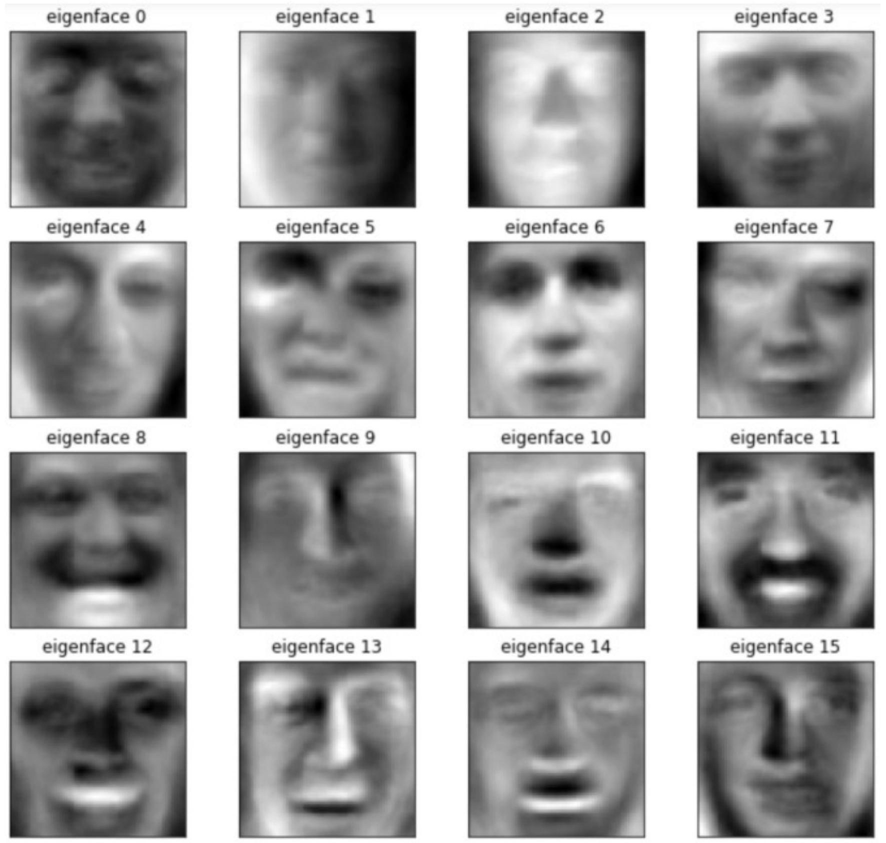


Figure #5: mean face



…

…

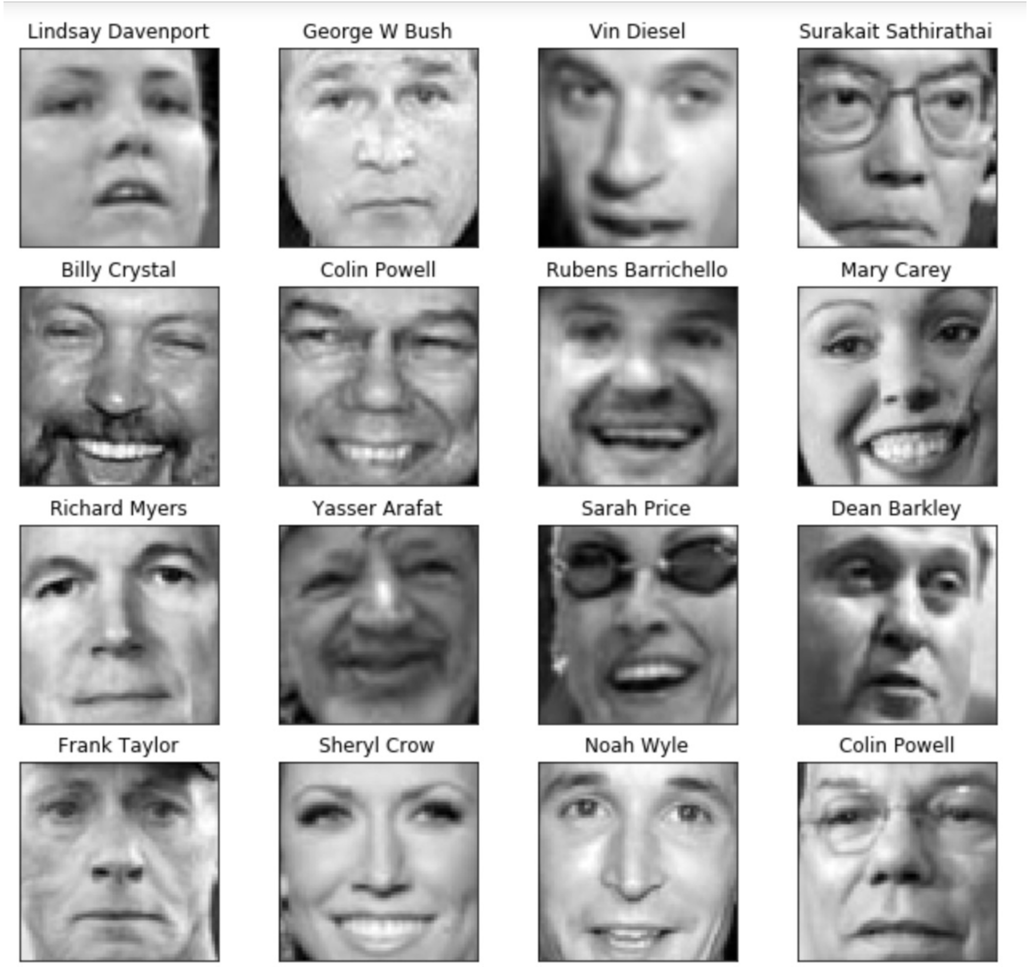
…

…

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

Let’s try reconstructing these faces with the eigenfaces now!

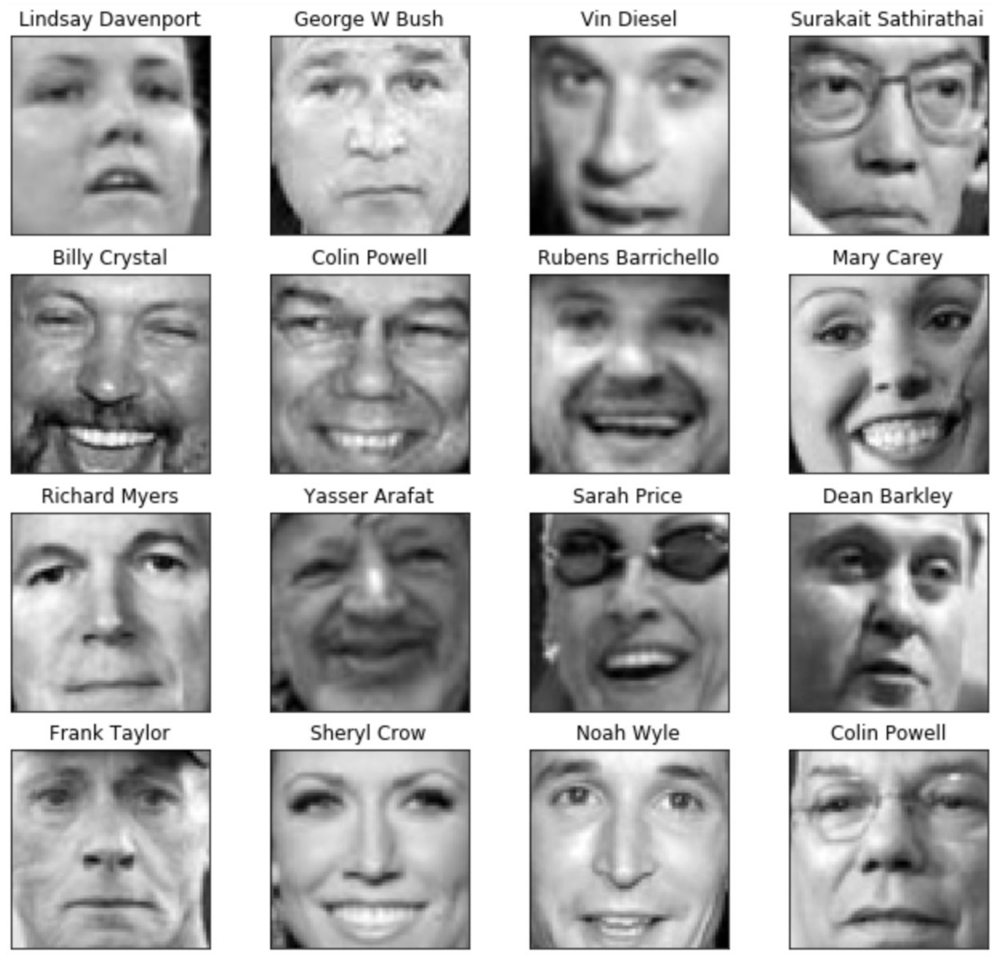


(1000 64×64 images)

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

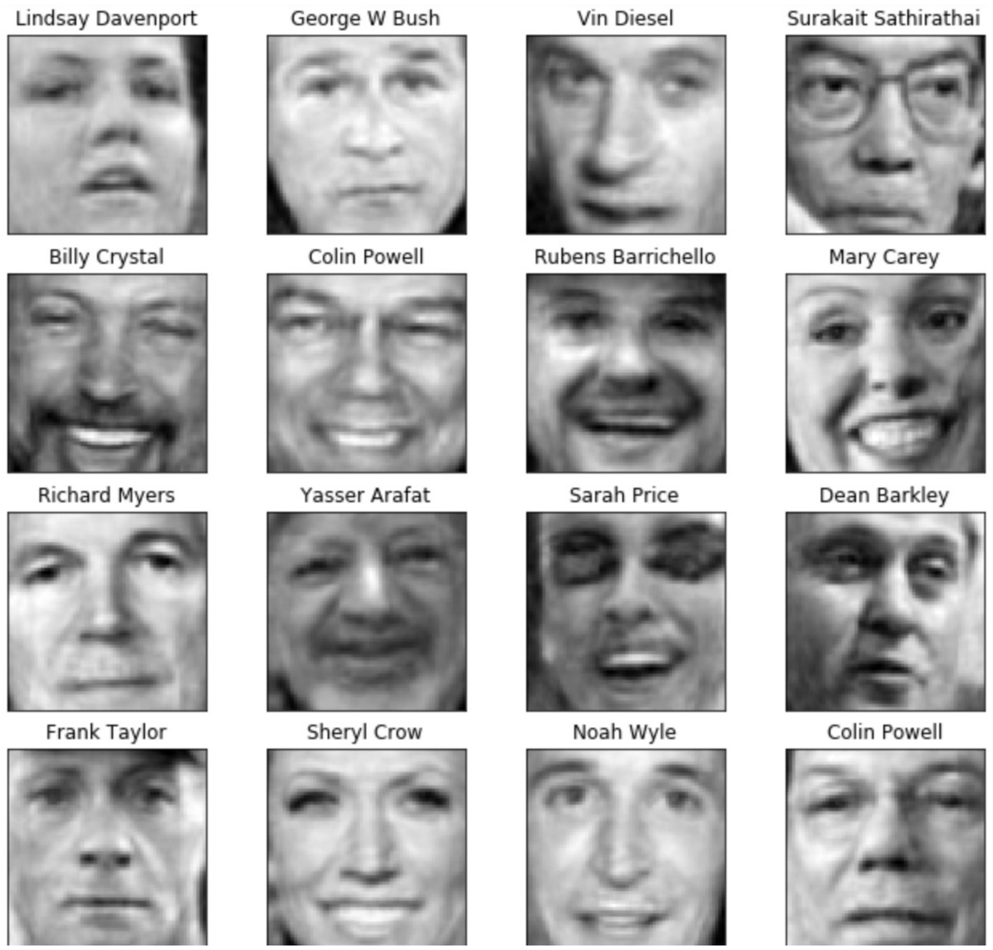
… with 1000 eigenvectors



https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

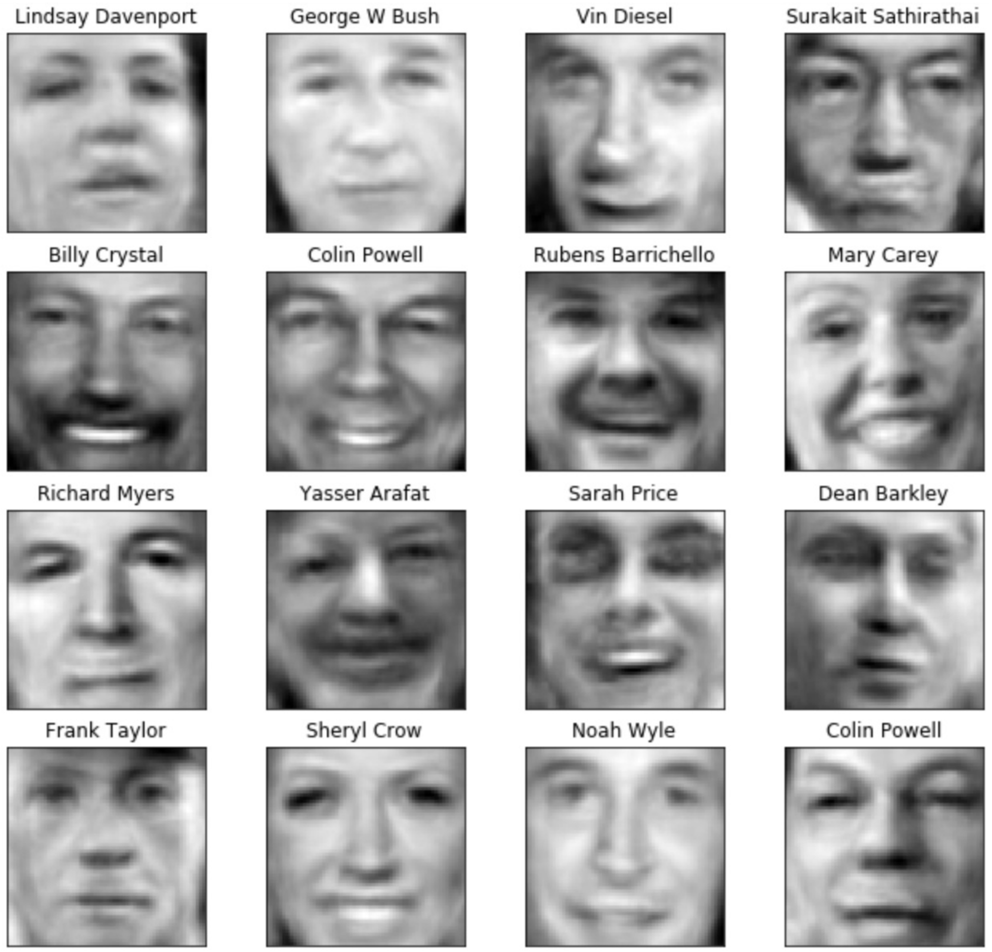
… with 250 eigenvectors



https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

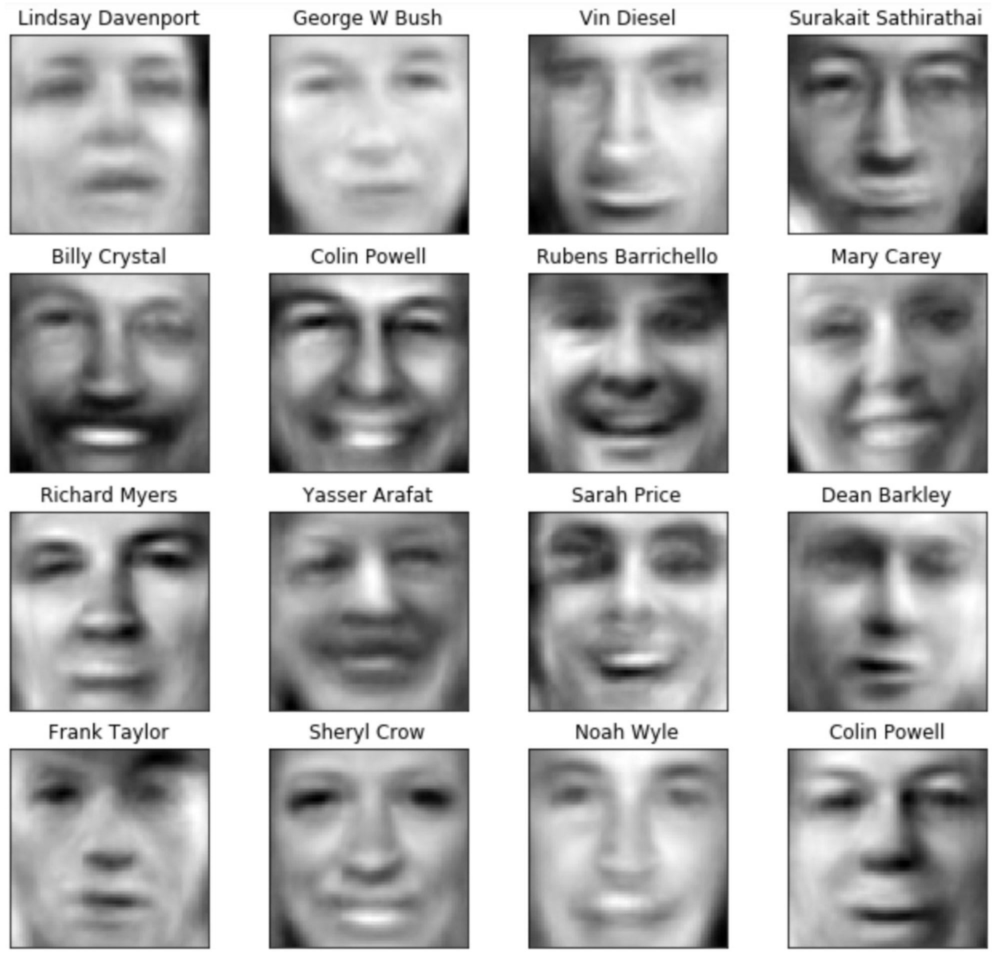
... with 100 eigenvectors



https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

… with 50 eigenvectors



https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

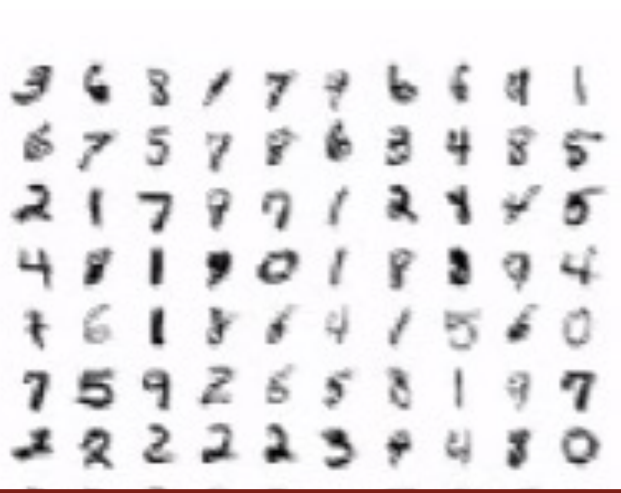
Utility of PCA

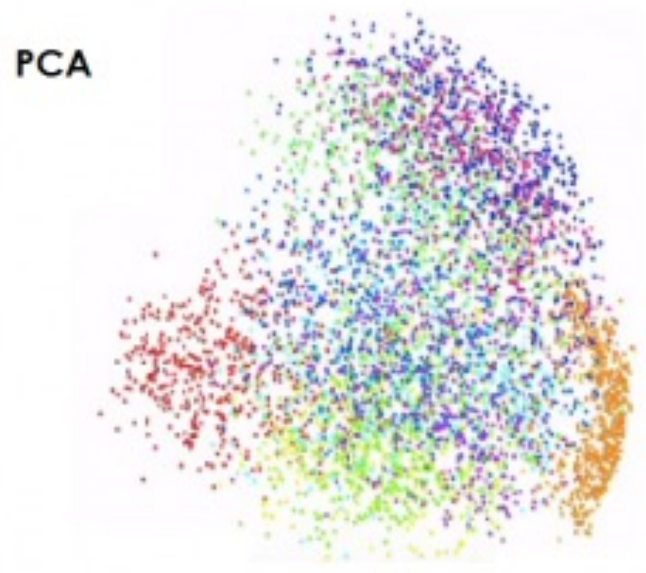
- PCA is often used as a preprocessing step for supervised learning
  - reduces dimensionality
  - eliminates multicollinearity

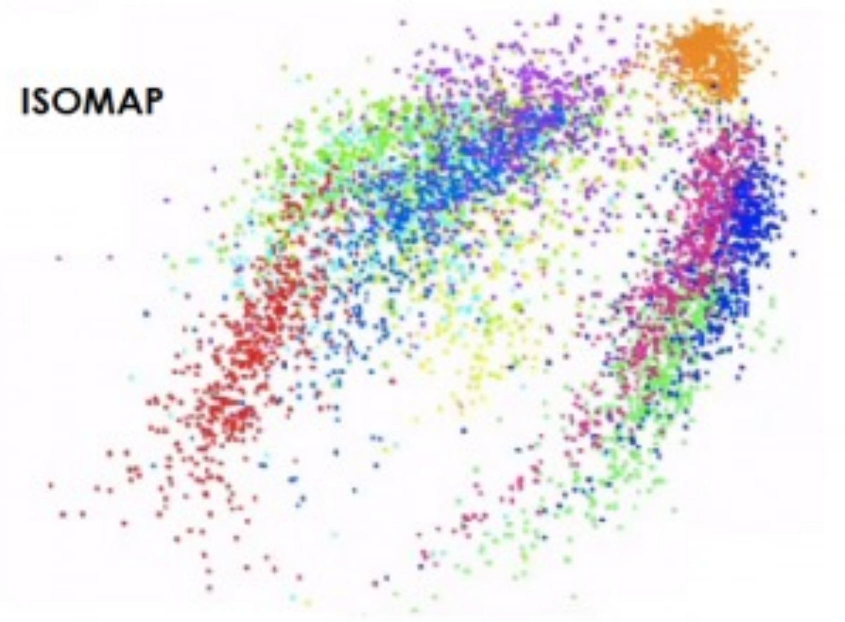
- Can back-project to the original feature space

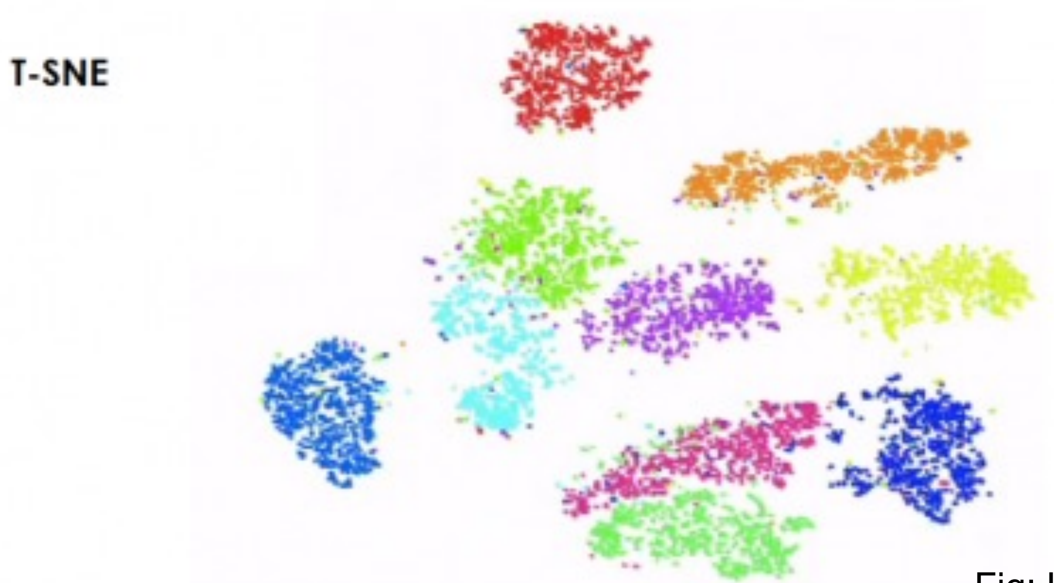
- Can also be used to aid in visualization

Beyond PCA: Non-linear dimensionality reduction









T-SNE and ISOMAP are more powerful methods that use nonlinear mappings, but:

- Require careful hyperparameter tuning
- Harder to optimize
- Not as easy to interpret, no easy projection back to original data

Fig: Laurens van der Maaten

Recap: Unsupervised Learning

Basic idea: reduce feature space to a much lower set of dimensions

- Clustering: find structural similarity, return one k-valued higher-level feature
- PCA: find orthonormal dimensions in order of most to least variance

- Can be useful for human inspection (visualization) as well as supervised ML

- Next: a very prominent datatype worthy of study on its own – text and documents!

Text and NLP

Goals of NLP

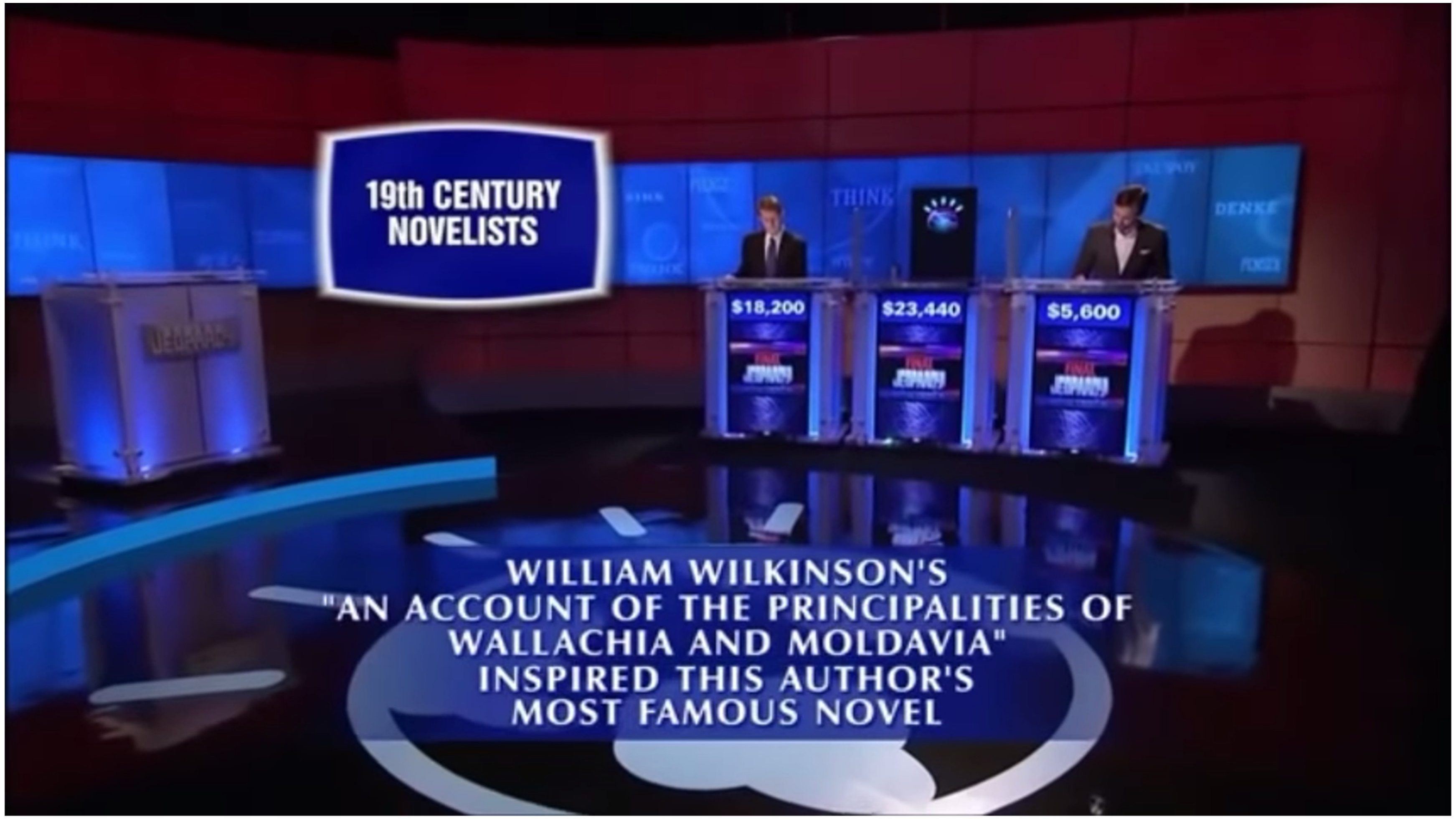
We would like to be able to understand text and extract all the same kinds of information in the same ways as humans might.

- recognize spam email
- search for webpages relevant to a search query
- read tweets and understand public sentiment on a topic
- read a novel and follow the plot
- hold a conversation with a person
- read a textbook and solve an exam
- recognize fake news articles
- translate from English to French

Language Understanding is Hard!

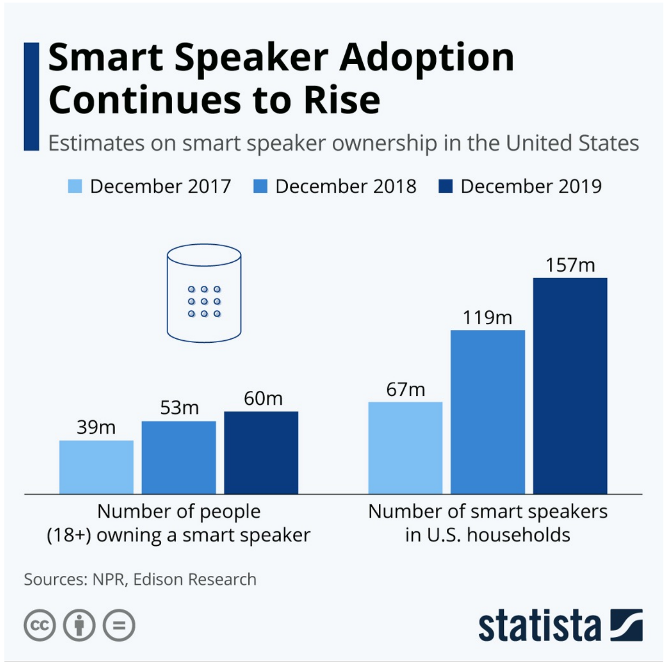
- Read a textbook on the Civil War and answer: "Did Abraham Lincoln have an iPhone?" (common sense)
- "Mary fought with Kate because she was a bad person". Answer: "Who was a bad person? Mary or Kate?" (ambiguity, requires long-term context)
- "The guitar didn't fit into the box because *it* was too *small*". Answer: "What was too small? The guitar or the box?" (ambiguity, requires common sense)

General Knowledge: IBM Watson Jeopardy! Challenge

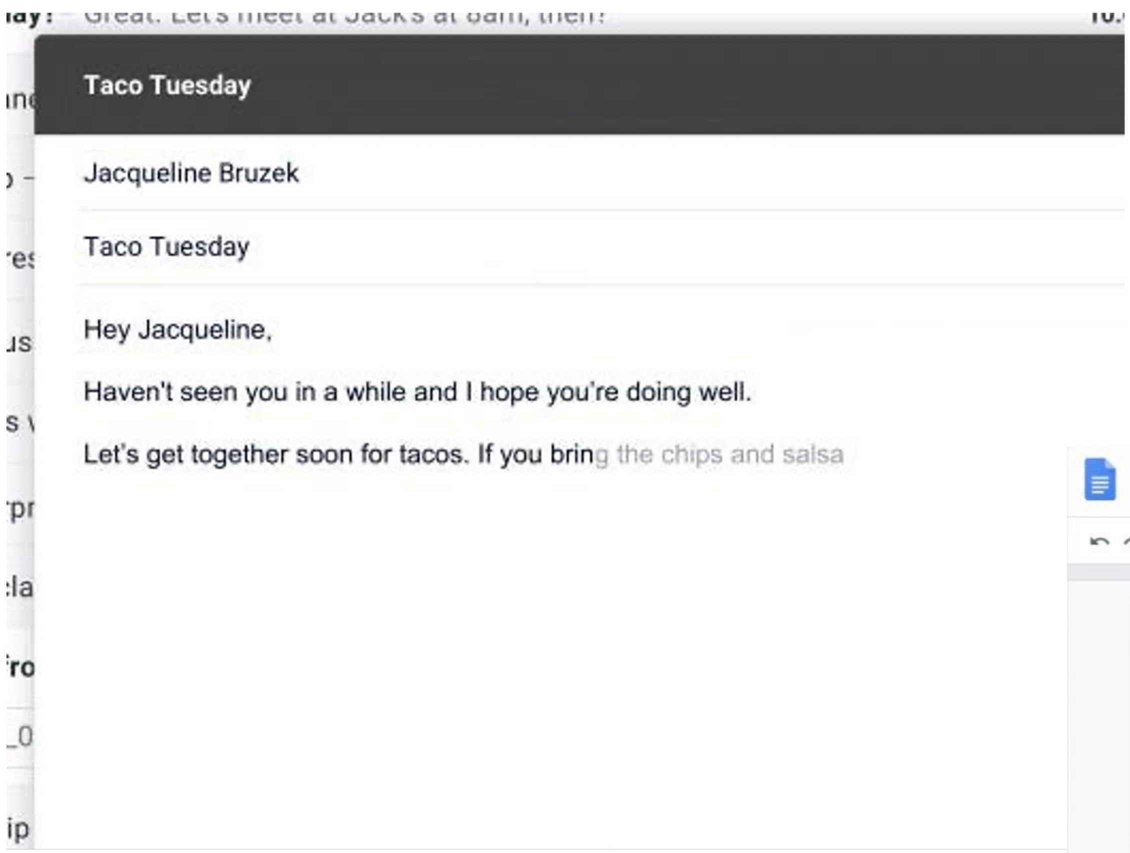

19th CENTURY NOVELISTS
$18,200
$23,440
$5,600
WILLIAM WILKINSON'S "AN ACCOUNT OF THE PRINCIPALITIES OF WALLACHIA AND MOLDAVIA" INSPIRED THIS AUTHOR'S MOST FAMOUS NOVEL

https://www.youtube.com/watch?v=Sp4q60BsHoY

Conversation: Smart Assistant Advancements



Writing: Smart Compose, Grammarly



Taco Tuesday

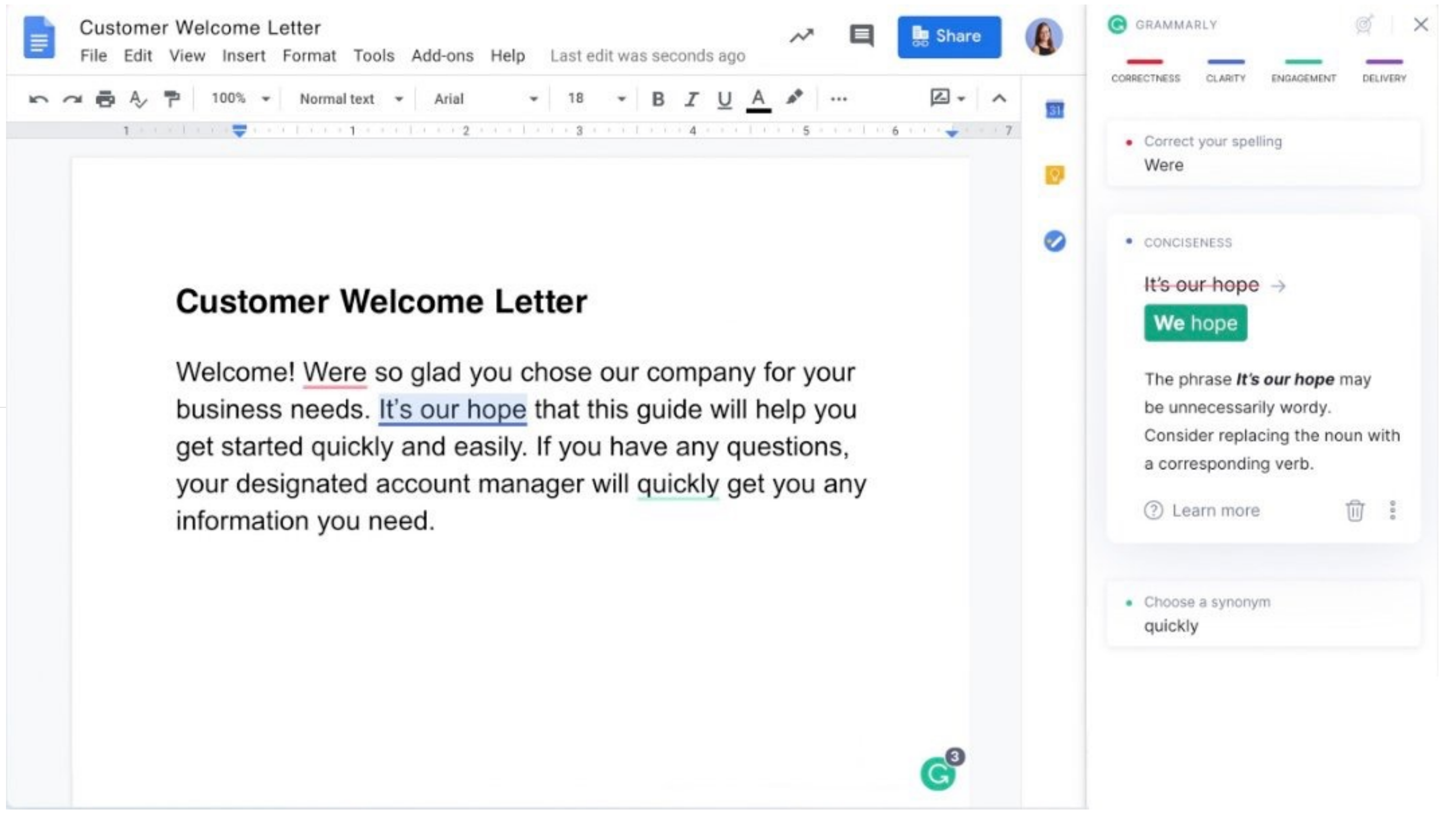
Jacqueline Bruzek

Taco Tuesday

Hey Jacqueline,

Haven't seen you in a while and I hope you're doing well.

Let's get together soon for tacos. If you bring the chips and salsa



Customer Welcome Letter
File Edit View Insert Format Tools Add-ons Help Last edit was seconds ago
Share
100% Normal text Arial 18
GRAMMARLY
CORRECTNESS CLARITY ENGAGEMENT DELIVERY

Customer Welcome Letter

Welcome! Were so glad you chose our company for your business needs. It's our hope that this guide will help you get started quickly and easily. If you have any questions, your designated account manager will quickly get you any information you need.

Correct your spelling
Were

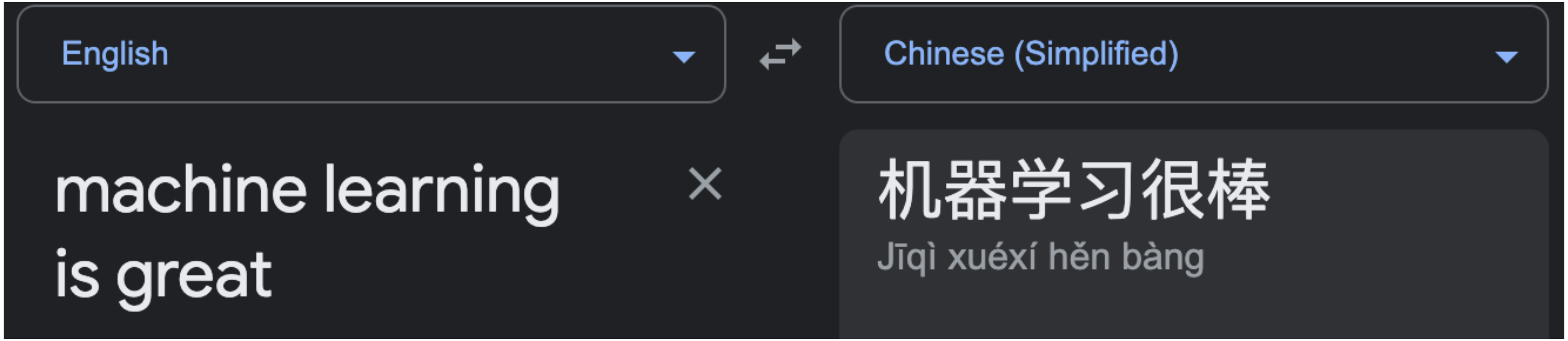
CONCISENESS
~~It's our hope~~ → **We hope**

The phrase ***It's our hope*** may be unnecessarily wordy. Consider replacing the noun with a corresponding verb.

Learn more

Choose a synonym
quickly

Translation

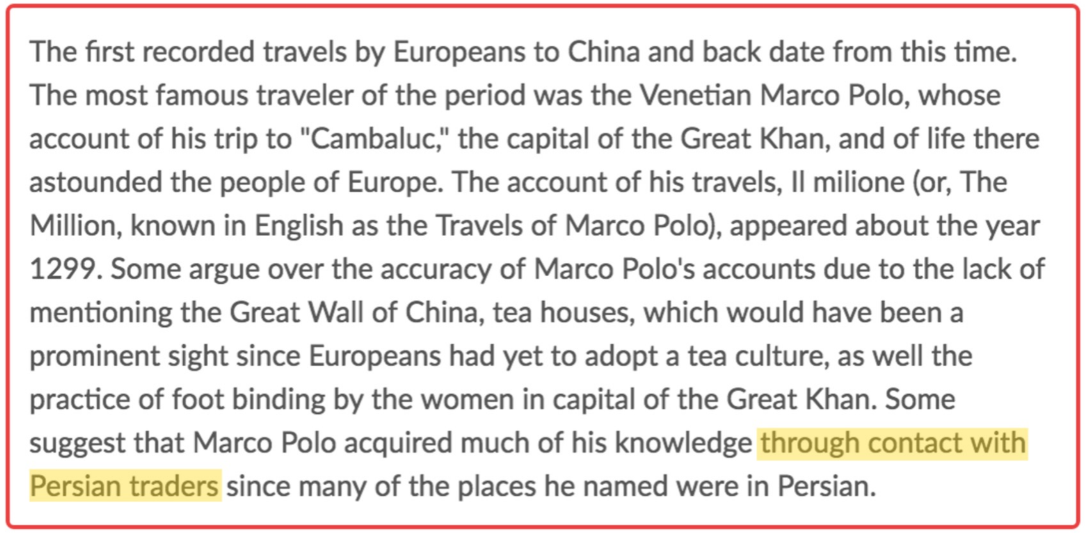


machine learning is great

机器学习很棒

Jīqì xuéxí hěn bàng

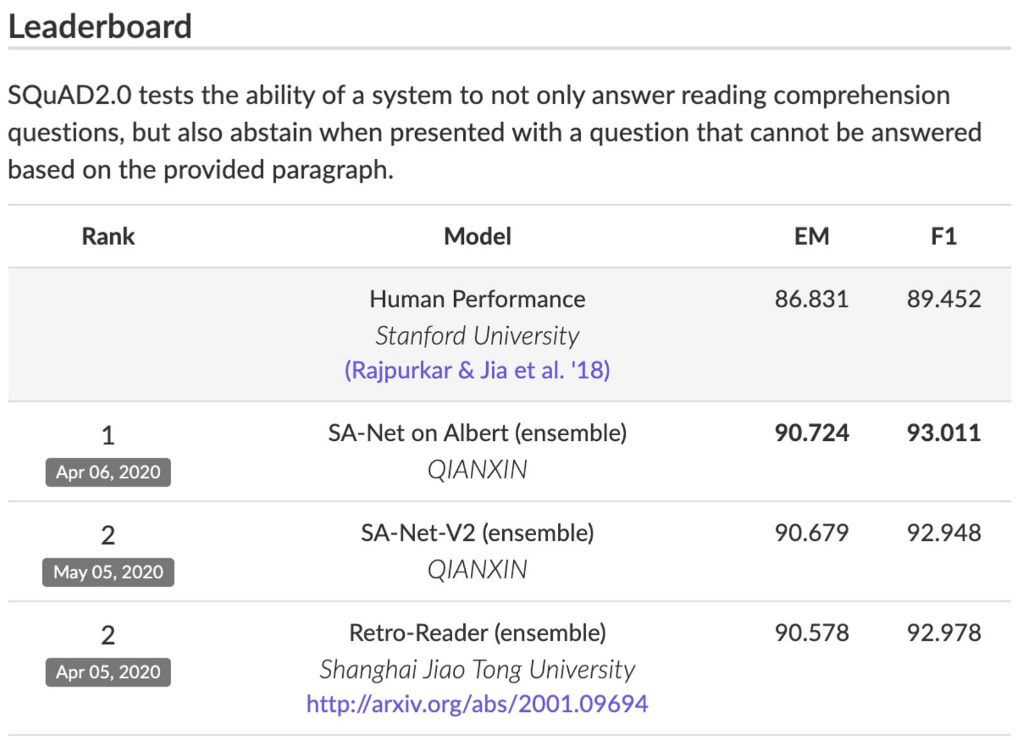
Comprehension: ML beats Humans at SQuAD



The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders



Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
2 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.578	92.978

Text Generation

Prompt

Write a creative ad for the following product to run on Facebook aimed at parents:

Product: Learning Room is a virtual environment to help students from kindergarten to high school excel in school.

Sample response

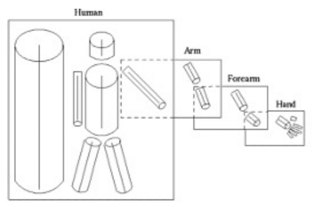
Are you looking for a way to help your child excel in school? Look no further than Learning Room! Our virtual environment is designed to help students from kindergarten to high school learn and grow. With our help, your child will be able to get ahead in school and achieve their academic goals.

A Basic NLP Pipeline

Hand-Defined Features for NLP

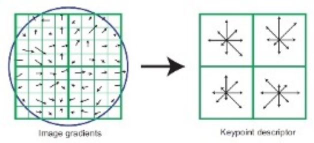
Similar history for NLP (with small lags)!

ML in Computer Vision



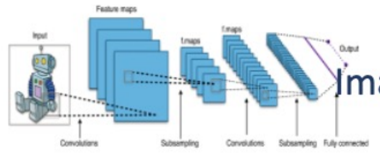
The *very* old: 1960's - Mid 1990's

Image → hand-def. features → hand-def. classifier



The old: Mid 1990's – 2012

Image → hand-def. features → learned classifier



The new: 2012 – ?

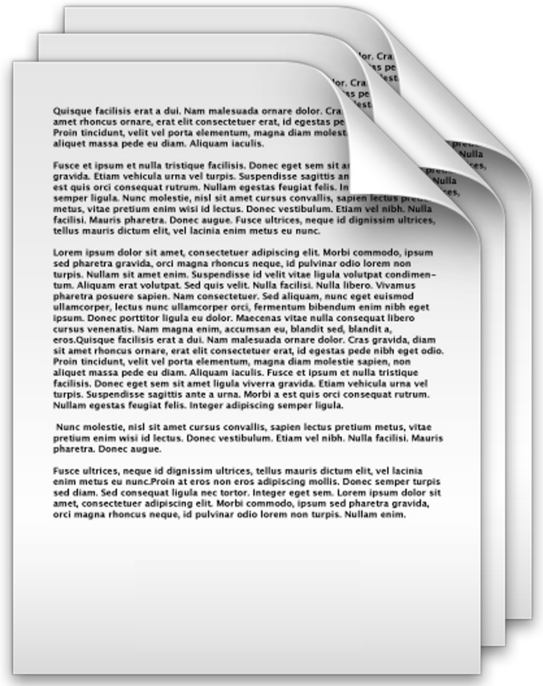
Image → jointly learned features + classifier with "deep" multi-layer neural networks

What would be a good representation of a document?

Bag of Words Representation

What is the ~~best~~ representation for documents?

simplest, yet useful



Idea: Treat each document as an *unordered set* of words

Lexicon: set of "all possible words". Two options:

- Union of words from all documents in the dataset
- Use Webster's dictionary, etc.

Bag of Words Representation

Represent document d as a vector of word counts $\boldsymbol{x}$

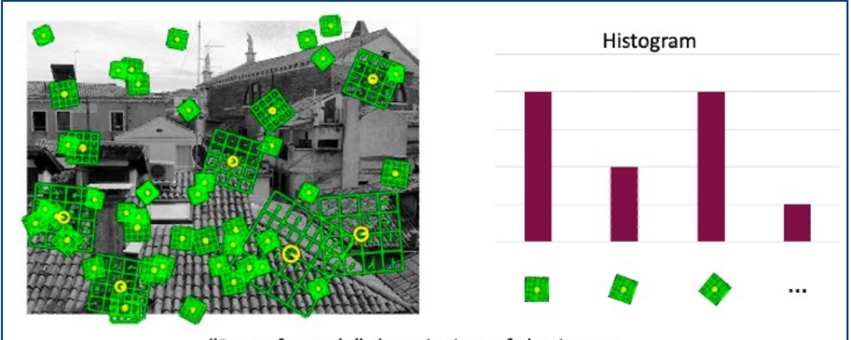
- x_j represents the count of word j in the document
  - $\boldsymbol{x}$ is sparse (few non-zero entries)

document d

number of times “abbey” occurred

aardvark	abacus	abandon	abase	abate	aberration	abbey	abbot	…	zoo
0	0	1	0	0	0	4	0	…	0

$\boldsymbol{x}$



“Bag-of-words” description of the image

We Lose Many Key Aspects, e.g.:

- Word senses (which have to come from *context*)
  “Took money out of the bank”
  “Got stuck on the river bank”
  “The pilot tried to bank the plane”

- Significance of some words vs others
  articles (“a”, “an”, “the”) vs unusual terms (“hagiography”)

Simple Improvements to Bag of Words

“n-grams”: Replace individual words with *n* consecutive words. E.g. 2-grams = count the number of times:

“I have a cat” ->

“I have” -- 1 occurrence, “have a” – 1 occurrence, “a cat” – 1 occurrence

“TF-IDF”: Down-weight words based on how commonly they occur in some dataset. e.g., words like “a”, “the” don’t count for as much as “Germany”, or “Olympics”.

Better Text Representations

Problems with Bag-of-Words Representations

Does not recognize that some word pairs are “more similar to each other” than others. E.g., “I have a dog”, “I have a cat”, and “I have a tomato” are all equally distant from each other, even though “dog” is much closer to “cat” than to “tomato”.

Need better representations of words

Does not recognize the sequential ordering of words. E.g., “Mary is faster than Jack” = “Jack is faster than Mary”

Need better representations of documents

Baseline Word Representation: One-Hot Encoding

Bag-of-Words (document) = $\sum_{\text{word } i \text{ in document}}$ onehot(word i)

- Onehot(“dog”) = [0, 0, 0, 1, 0, 0, 0]
- Onehot(“cat”) = [1, 0, 0, 0, 0, 0, 0]
- Onehot(“Tomato”) = [0, 0, 0, 0, 0, 1, 0]

Can we construct better vector representations of words (than one-hot encoding), where the distance between “dog” and “cat” is small but the distance between “dog” and “car” is large?

“**Vector Space Models**” :
Represent a word based on the statistics of its usage in a large collection of documents like Wikipedia entries / New York Times articles.

Term-Frequency Matrix

Count how many times a word (term) appears in each document in your dataset.

Note: columns are bag-of-words representations of the documents.

Wikipedia Article / Words	**Cat**	**Dog**	Apple Inc.	Apple (fruit)	Microsoft Inc.	…
a	377	370	842	231	286	…
the	929	787	1690	503	872	…
apple	0	0	1091	166	14	…
computer	0	0	88	0	36	…
fur	15	2	0	0	0	…
hair	6	6	0	0	0	…
…	…	…	…	…	…	…

Term-Frequency Matrix

Similar or related words appear keep co-occurring in documents.

Words \ Wikipedia Article	**Cat**	**Dog**	Apple Inc.	Apple (fruit)	Microsoft Inc.	…
a	377	370	842	231	286	…
the	929	787	1690	503	872	…
apple	0	0	1091	166	14	…
computer	0	0	88	0	36	…
fur	15	2	0	0	0	…
hair	6	6	0	0	0	…
…	…	…	…	…	…	…

Term-Frequency Matrix

Similar or related words appear keep co-occurring in documents.

Words \ Wikipedia Article	**Cat**	**Dog**	Apple Inc.	Apple (fruit)	Microsoft Inc.	…
a	377	370	842	231	286	…
the	929	787	1690	503	872	…
apple	0	0	1091	166	14	…
computer	0	0	88	0	36	…
fur	15	2	0	0	0	…
hair	6	6	0	0	0	…
…	…	…	…	…	…	…

Term-Frequency Matrix

Similar or related words appear keep co-occurring in documents.

Wikipedia Article / Words	**Cat**	**Dog**	Apple Inc.	Apple (fruit)	Microsoft Inc.	…
a	377	370	842	231	286	…
the	929	787	1690	503	872	…
apple	0	0	1091	166	14	…
computer	0	0	88	0	36	…
fur	15	2	0	0	0	…
hair	6	6	0	0	0	…
…	…	…	…	…	…	…

Term-Frequency Matrix

Idea 1: Represent each word by its corresponding row!

Representation of “apple”

Words \ Wikipedia Article	**Cat**	**Dog**	Apple Inc.	Apple (fruit)	Microsoft Inc.	…
a	377	370	842	231	286	…
the	929	787	1690	503	872	…
apple	0	0	1091	166	14	…
computer	0	0	88	0	36	…
fur	15	2	0	0	0	…
hair	6	6	0	0	0	…
…	…	…	…	…	…	…

Term-Term Matrix

Idea 2: Represent each word by the words in its neighborhood

“The **distributional hypothesis** in linguistics is derived from the semantic theory of language usage, i.e. words that are used and occur in the same contexts tend to purport similar meanings.”

“A word is characterized by the company it keeps" - John Firth

- For example, the words that frequently co-occur with “dog” in a sentence might be words like “play”, “pet”, “sleep”, “fur”, “feed”, etc.

What words do you think would co-occur with “cat”?

What words do you think would co-occur with “car”?

Term-Term Matrix

I have a pet cat.

I have a pet dog.

I have a pet car.

“pet”-“car” don’t co-occur near each other as often as “pet”- “cat” or “pet”- “dog”.

Term-Term Matrix

- Count how many times a word appears within the neighborhood "context" of another word (e.g., 4 words to the left/right).

Words \ Words	**pet**	**play**	tire	engine	run	…
dog	872	649	1	7	378	…
cat	789	831	5	0	285	…
car	12	4	290	927	562	…
…	…	…	…	…	…	…

Term-Term Matrix

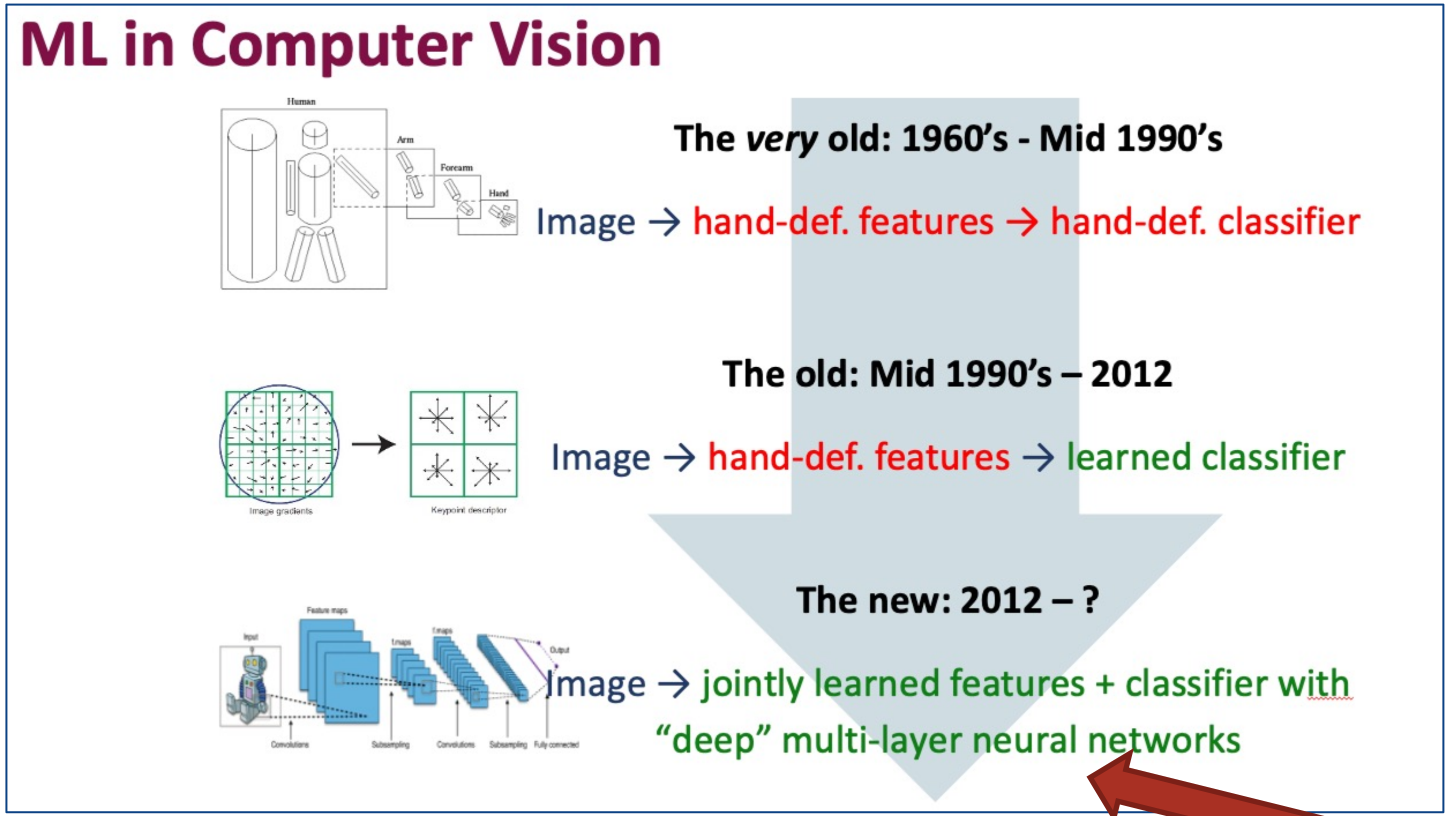
Similar words co-occur with the similar sets of words in the sentences.

Again, represent each word by its corresponding row.

Words / Words	**pet**	**play**	tire	engine	run	…
dog	872	649	1	7	378	…
cat	789	831	5	0	285	…
car	12	4	290	927	562	…
…	…	…	…	…	…	…

Unsupervised Learning of Compact Word Vectors

Learned Word Vectors


Human
Arm
Forearm
Hand
Image gradients
Keypoint descriptor
Input
Feature maps
f.maps
Output
Convolutions
Subsampling
Convolutions
Subsampling
Fully connected

ML in Computer Vision

The *very* old: 1960's - Mid 1990's

Image → hand-def. features → hand-def. classifier

The old: Mid 1990's – 2012

Image → hand-def. features → learned classifier

The new: 2012 – ?

Image → jointly learned features + classifier with "deep" multi-layer neural networks

How might we learn vector representations of words?

More Compact Representations

Word Vector Representation Sizes:

- Term-Frequency = # of documents
- Term-Term = # of words

These are huge vectors, likely with lots of zeros.

Can we get a more compact representation?

From Term-Term to Word2Vec

Idea: Train a neural network classifier to predict whether a word will co-occur in the context of another word.

The **classifier weights** will themselves encode vector space representations of words!

Word2Vec Training Data

- “The quick brown fox jumped over the lazy dog.”

Word	Context
quick	[the, brown]
brown	[quick, fox]
fox	[brown, jumped]
jumped	[fox, over]
…	…

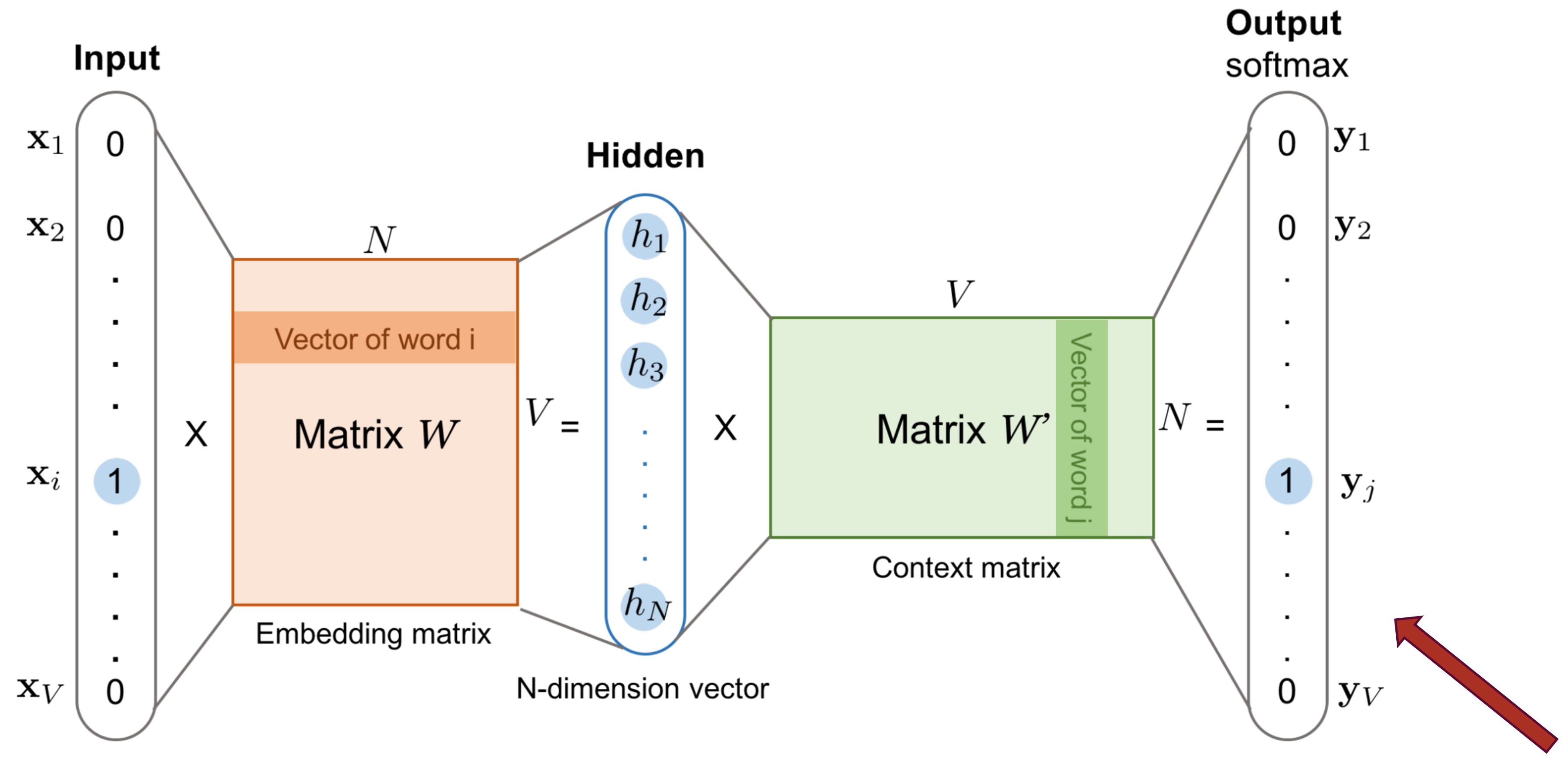
Word2Vec Training Data

- "The quick brown fox jumped over the lazy dog."

Input	Output
quick	the
quick	brown
brown	quick
brown	fox
fox	brown
…	…

Millions of training input-output pairs, from parsing huge, unlabeled datasets (e.g. all of Wikipedia)

Word2Vec Classifier

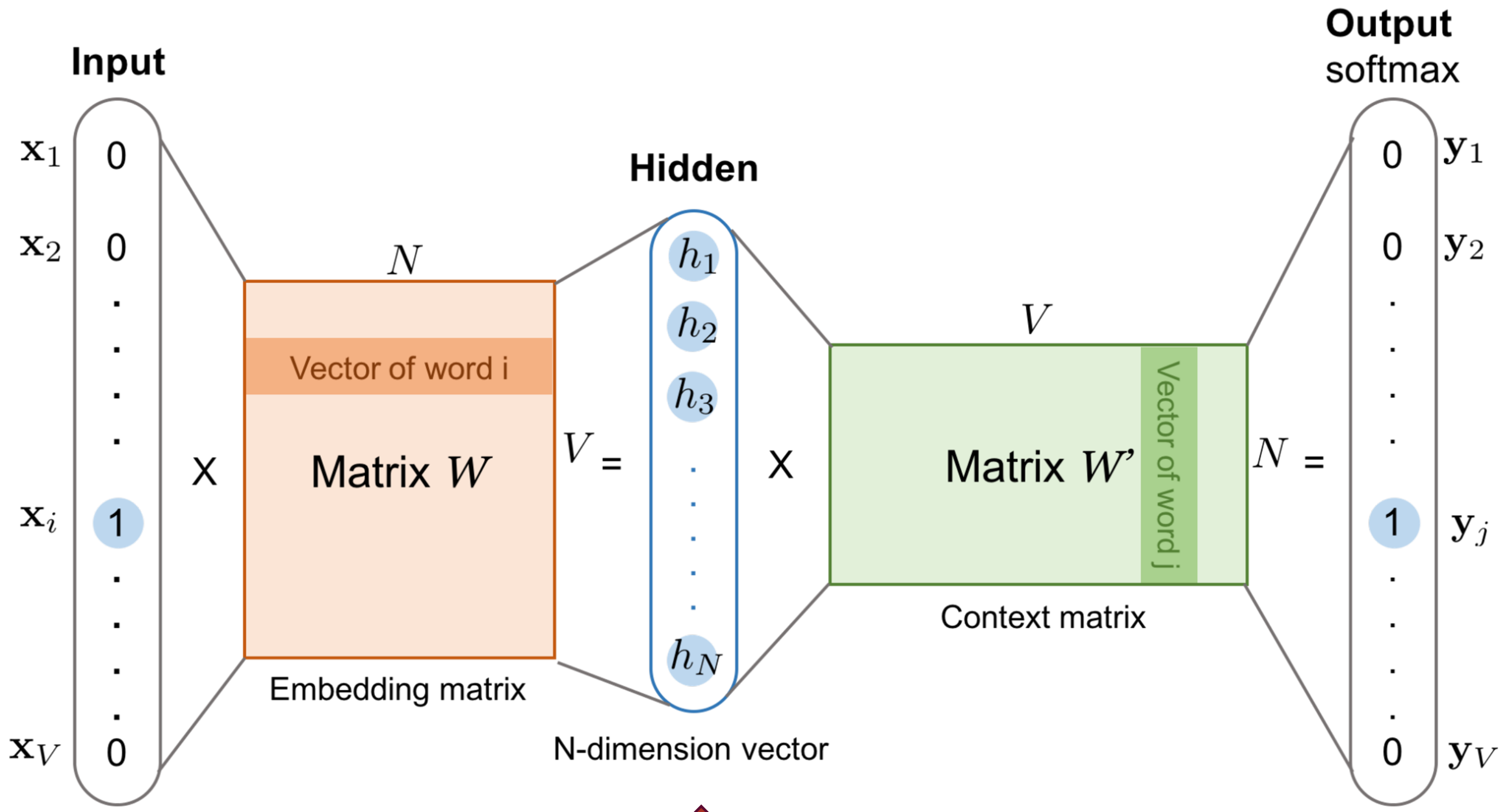


One-Hot Encoding
for
the Input Word

One-Hot Encoding
for
the Output Word

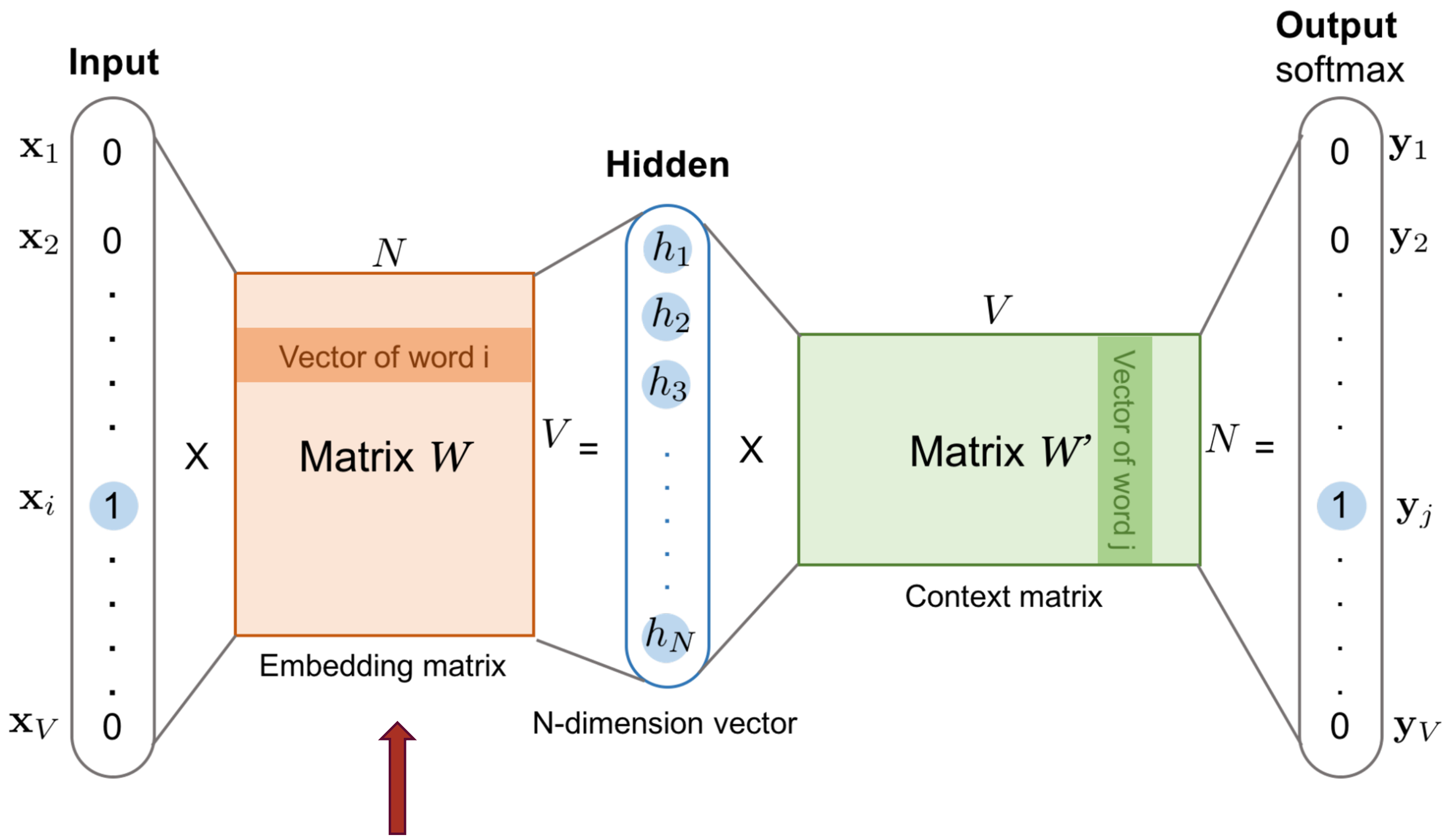
Source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word2Vec Classifier



***N* hidden units**
where *N* << *V*

Word2Vec Classifier



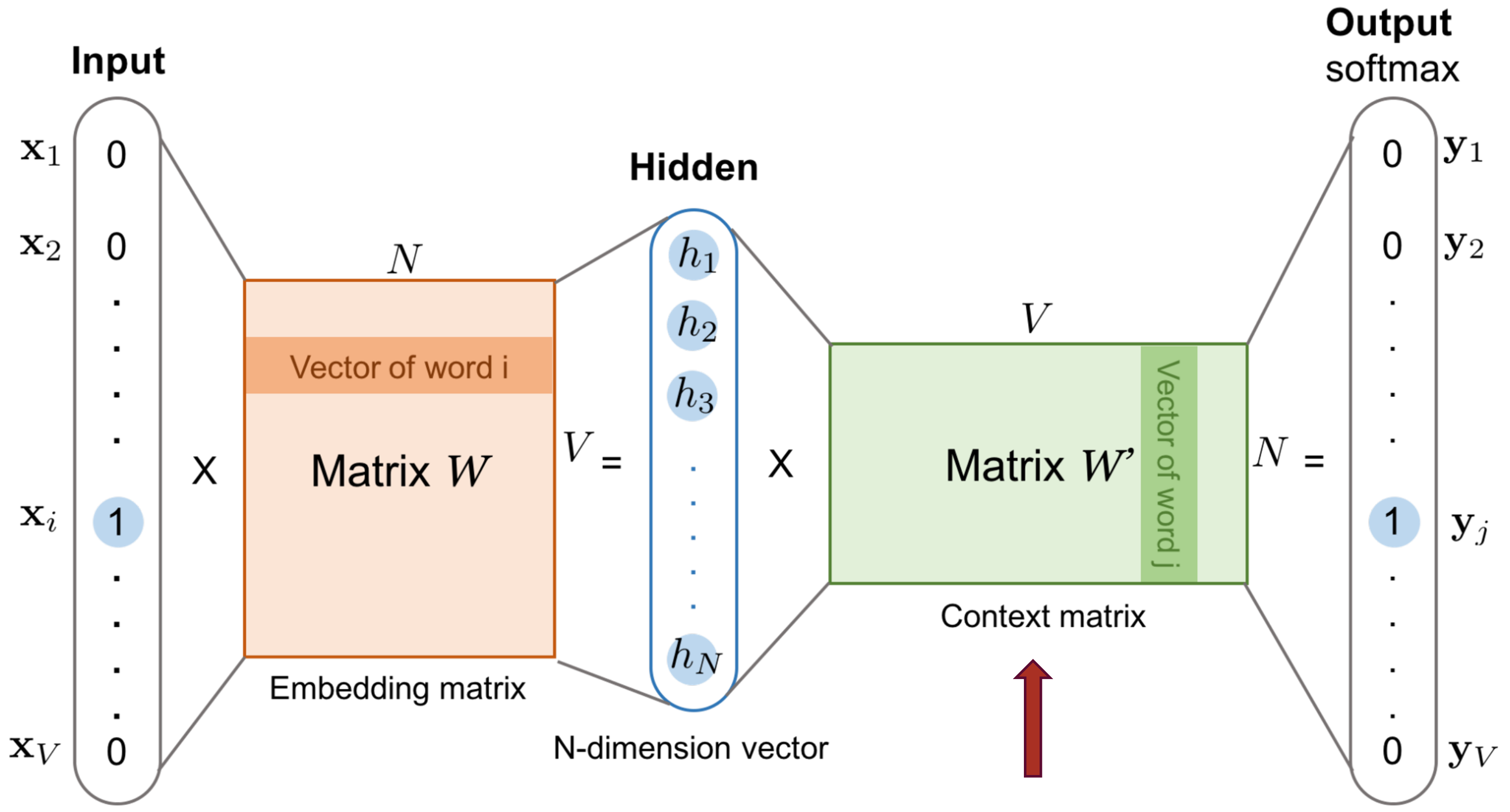
Has N columns, V (vocabulary size) rows.

Each row corresponds to a word.

i^{th} row = a vector representation for word #i

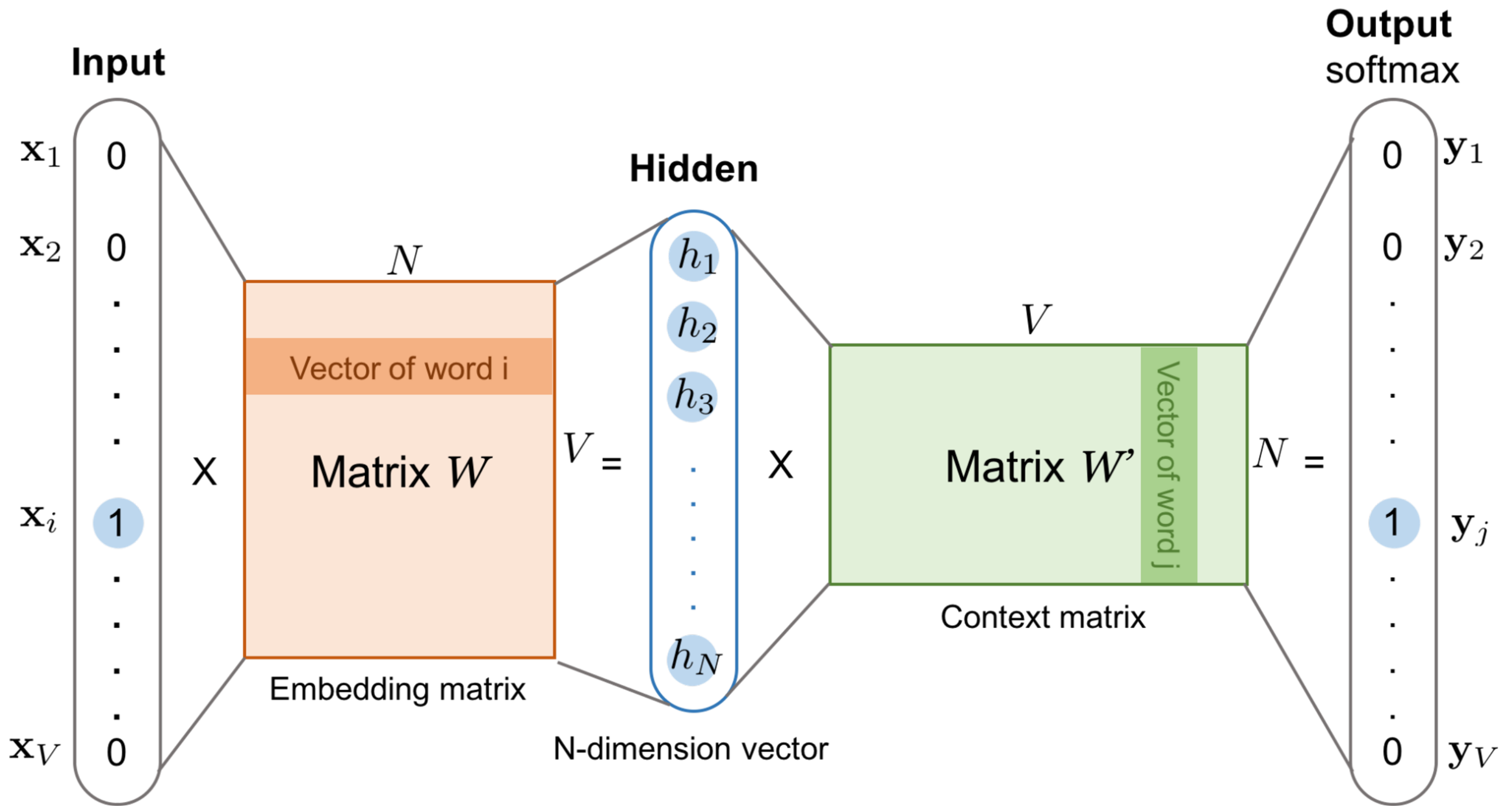
“Target Embedding”

Word2Vec Classifier



Has V (vocabulary size) columns, N rows.
Each column corresponds to a word.
i^{th} column = another vector representation for word #i
"Context Embedding"

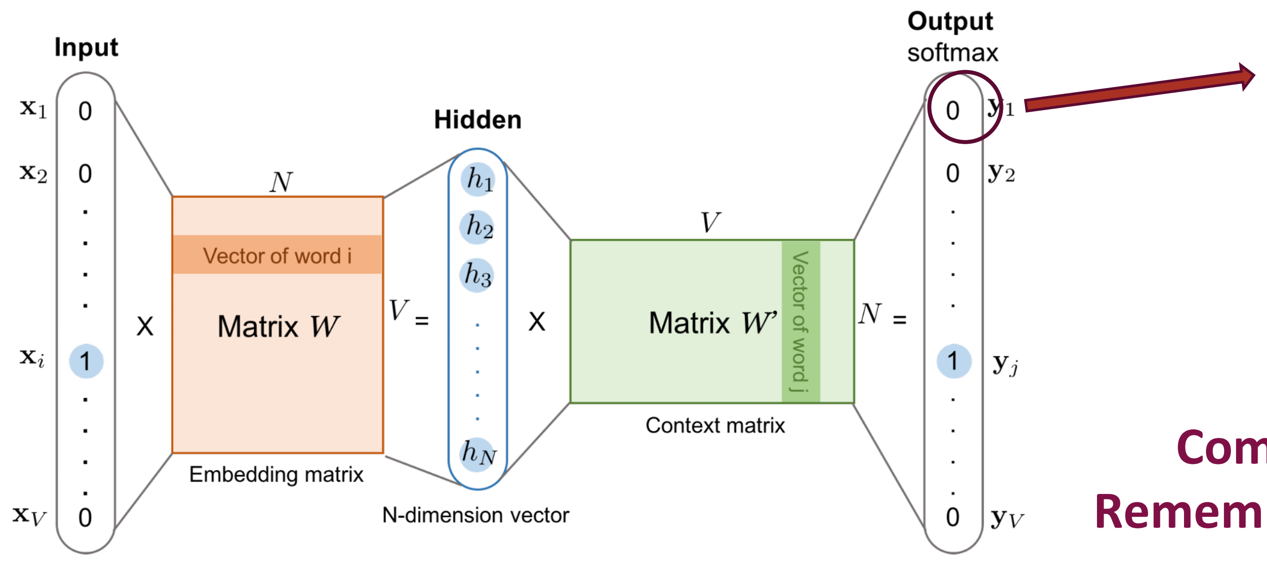
Word2Vec Classifier



After training, we can make our final word vector a concatenation of the two embeddings or just use one.

Word2Vec Training v1

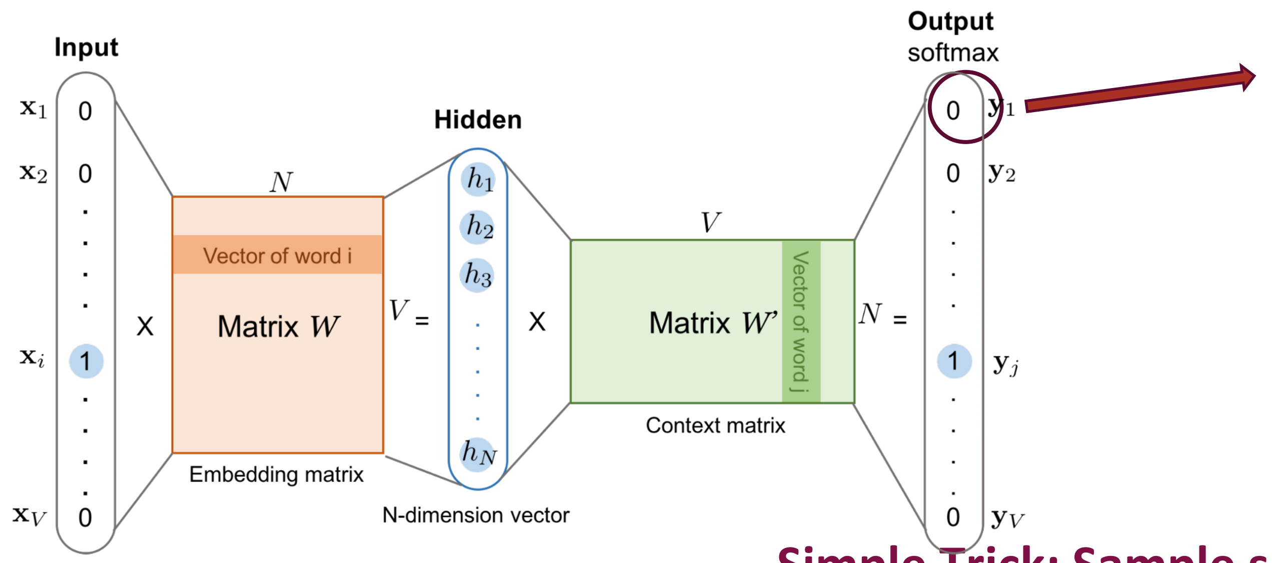
Standard softmax loss, then train the neural network.



$$p(w_o|w_{in}) = \frac{\exp(v'_{w_o} . v_{w_{in}})}{\sum_{k=1}^{V} \exp(v'_{w_k} . v_{w_{in}})}$$

Computing this denominator will be expensive. Remember that the vocabulary size V is of the order of millions of words!

Scaling Word2Vec Training



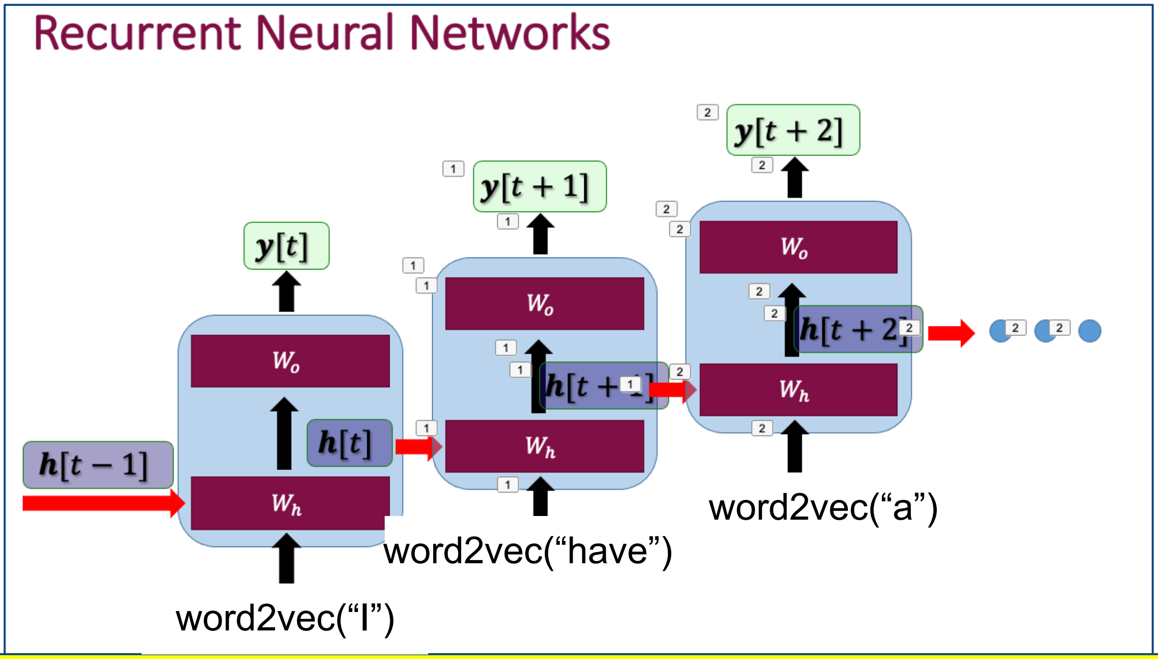
$$p(w_o|w_{in}) \approx \frac{\exp(v'_{w_o} . v_{w_{in}})}{\sum_{k=1}^{K} \exp(v'_{w_k} . v_{w_{in}})}$$

Simple Trick: Sample some random K-1<<V negative example words for each sample. e.g. K=2, 5, 20 etc.

Also means we need to update many fewer weights during each iteration of gradient descent.

From Words to Documents

- Sentence2Vec, Paragraph2Vec scale these Word2Vec ideas to learn direct embeddings for sentences / paragraphs.
- However, much more common to treat as a sequence of words, and represent each word by its word2vec-style representation:



"j'ai un chien"

"I have a dog"

Simple "sequence-to-sequence" models like these produced huge advances in machine translation in 2014.

Properties of Word2Vec

- Words that co-occur have vector representations that are close together (Euclidean distance).

Keep in mind this means that “sofa” and “couch” (synonyms) will be close together, but also things like “hot” and “cold” (antonyms) will also be close together.

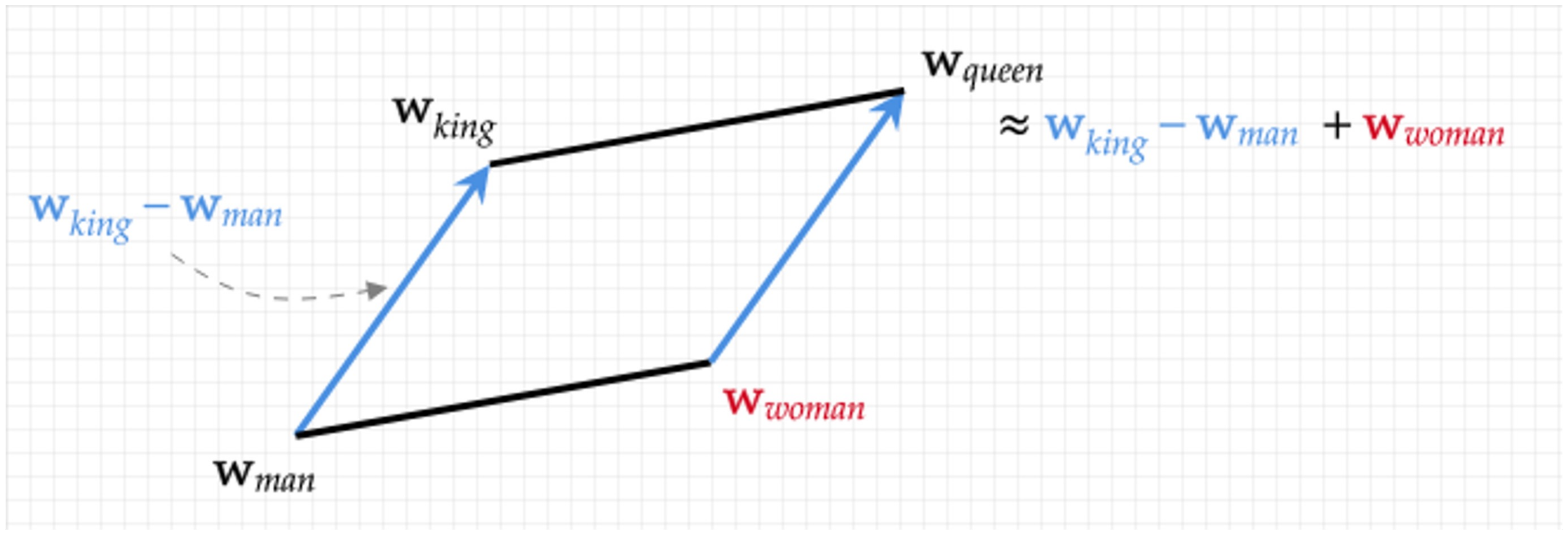
People both say “It’s ____ outside today.”

⍰ “hot” and “cold” co-occur with the same words often in sentences.

Properties of Word2Vec

- Vector operations (vector addition and vector subtraction) on word vectors often capture the semantic relationships of their words.

man : king :: woman: ?


$\mathbf{w}_{queen}$
$\approx \mathbf{w}_{king} - \mathbf{w}_{man} + \mathbf{w}_{woman}$
$\mathbf{w}_{king}$
$\mathbf{w}_{king} - \mathbf{w}_{man}$
$\mathbf{w}_{woman}$
$\mathbf{w}_{man}$

Source: https://www.ed.ac.uk/informatics/news-events/stories/2019/king-man-woman-queen-the-hidden-algebraic-struct

Use in Practice

Other word vector embeddings similar to word2vec:

- Continuous Bag of Words (CBOW)
- GLoVe

Available freely, and often used off-the-shelf:

- English word2vec weights trained on Google News data
- GloVe vectors trained on the Common Crawl dataset and a Twitter dataset.

If you have a lot of training data or a very different / niche domain (e.g., medical text), you might want to train your own word vectors on your dataset!