Announcements

- Quiz-1 grades are out.
- Quiz-2 out tonight, due in 24 hours.

- Feedback from Pset1
  - **Start early.**
  - If you start working on the day of, post on piazza, not reasonable to expect TFs to respond.

Announcements

- Labs, solutions, slides -> course website
  - Updated all the slides with slido responses.

- Problem sets, quizzes -> on Gradescope.

Quiz-1

What are the issues shared by both Average Nearest Neighbor and K Nearest Neighbor algorithms? Choose all that apply.

- [x] Both algorithms are computationally intensive and suitable only for small datasets
- [x] Both algorithms might require vector quantization based on complexity of input features.
- [x] Both algorithms require a significant amount of memory for model storage
- [x] Both algorithms struggle with handling high-dimensional data

Last time

- Bagging and Random Forests
- Feature normalization
- Curse of dimensionality
  - PCA.

Goal: Pick a set of directions that leads to minimal information loss

Normalize your data → Covariance matrix → Eigenvalue decomposition using SVD

Recall: PCA Algorithm

Normalize features (ensure every feature has zero mean) and optionally scale feature

Compute "covariance matrix" Σ:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)}\right)\left(x^{i}\right)^{T}$$

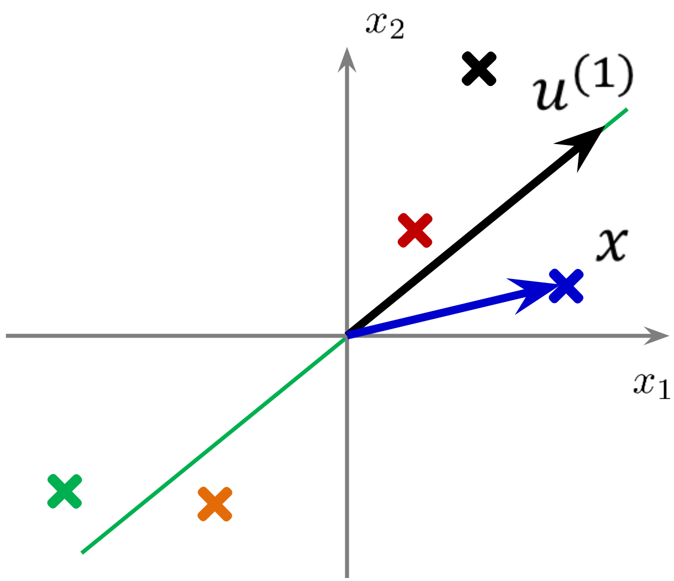
Compute its "eigenvectors":

$[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = \mathbf{svd}(\boldsymbol{\Sigma});$

$$\mathcal{U} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{d\times d}$$

Keep first s eigenvectors and project to get new features r

Principal Components Analysis



Find orthonormal basis vectors

$$\mathcal{U}^* = [u^{(1)} \quad \ldots \quad u^{(s)}], \text{ where s} \ll \text{d}$$

$$r_i = \mathcal{U}^{*T}(x_i - \mathrm{mean}(\{x\}))$$

Reconstructed data point

$$\tilde{x} = \sum_{i=1}^{s} r_i u^{(s)}$$

Cost function:

$$J = \frac{1}{N}\sum_{i=1}^{N}\left\|\tilde{x}^i - x^i\right\|^2$$

Want: $\min_{U} J$

Principal Components Analysis

Good for
Data visualization
Reduce compute / memory overhead

Principal Components Analysis

Good for	**Not to use for**
Data visualization	Tackling overfitting
Reduce compute / memory overhead	

Real world example

Mean image from Japanese Facial Expression dataset

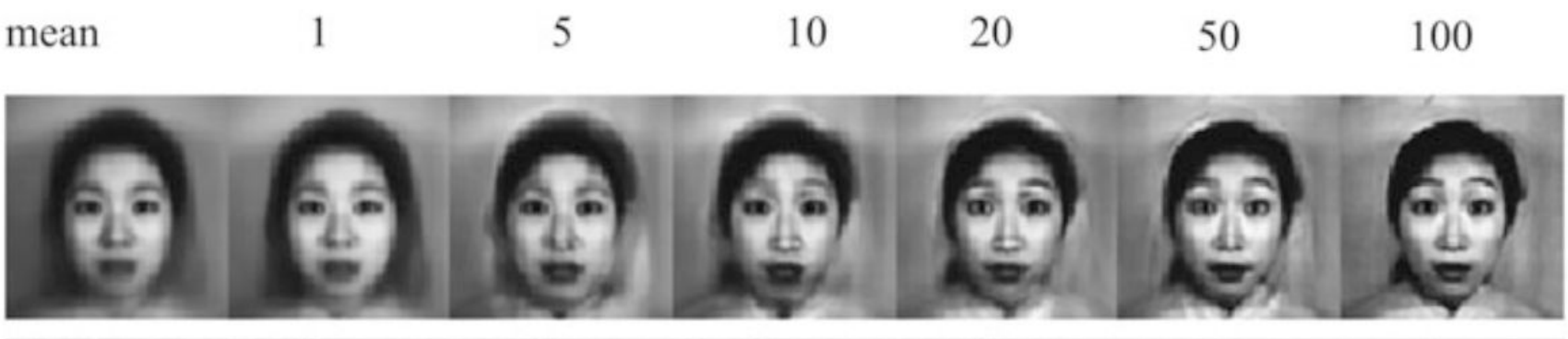
First sixteen principal components of the Japanese Facial Expression dat

Reconstructing original signal from PCs

Reconstructed data point

$$\tilde{x} = \sum_{i=1}^{s} r_i u^{(s)}$$

Sample Face Image



- archness of the eyebrows
- hair tied back
- nose shape changing as more PCs are added

Today

- Dimensionality reduction
  - PCoA
  - CCA
- Bag of Words
- Latent Semantic Analysis
- Canonical Correlation Analysis

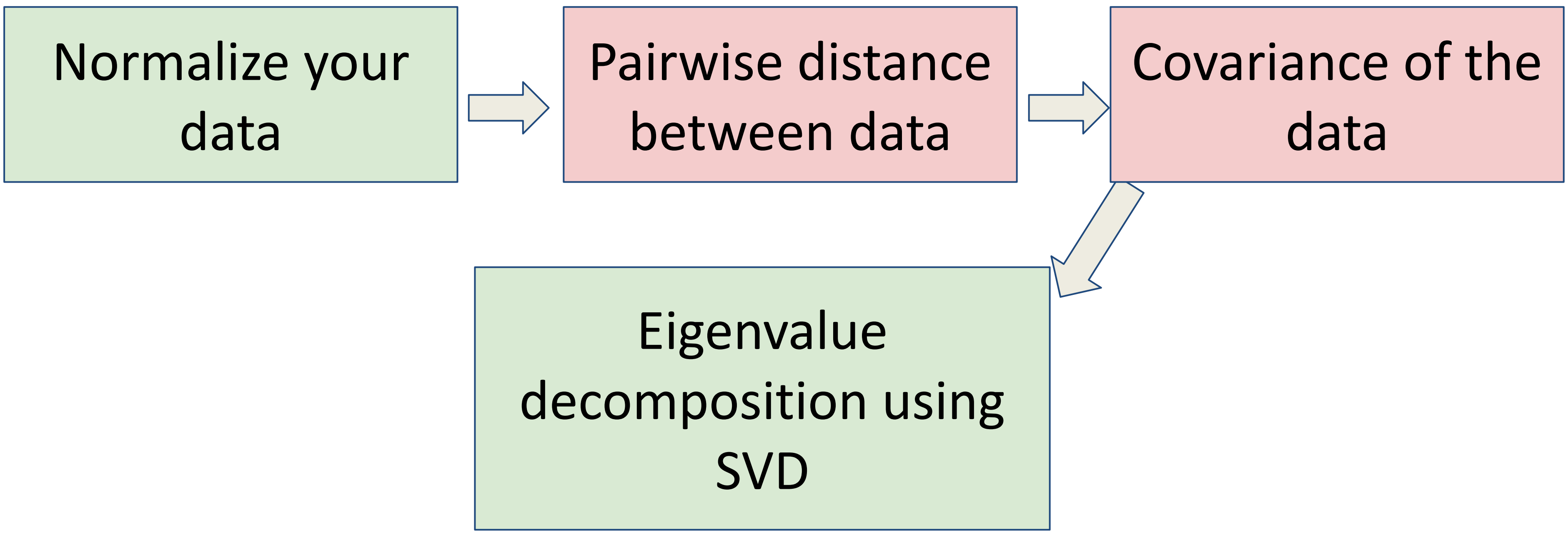
PCA

Normalize your data → Covariance of the data → Eigenvalue decomposition using SVD

Key observation: For high-dimensional data and fewer data points, covariances could be inaccurate.

Principal Coordinate Analysis (PCoA)

- Compared to PCA, focuses more on **distances between samples** (better for few sample, high-dim. data).



PCoA is a specific type of Multidimensional Scaling (MDS) techniques

ID	DBP	SBP	BMI	Chol
1	82	132	18	192
2	86	133	20	240
3	98	145	35	140
4	85	139	22	170
5	87	145	28	218

ID	DBP	SBP	BMI	Chol
1	-0.917	-1.086	-0.955	0.000
2	-0.262	-0.926	-0.665	1.222
3	1.703	0.990	1.504	-1.324
4	-0.426	0.032	-0.376	-0.560
5	-0.098	0.990	0.492	0.662
Mean	0	0	0	0
SD	1	1	1	1

Distance matrix

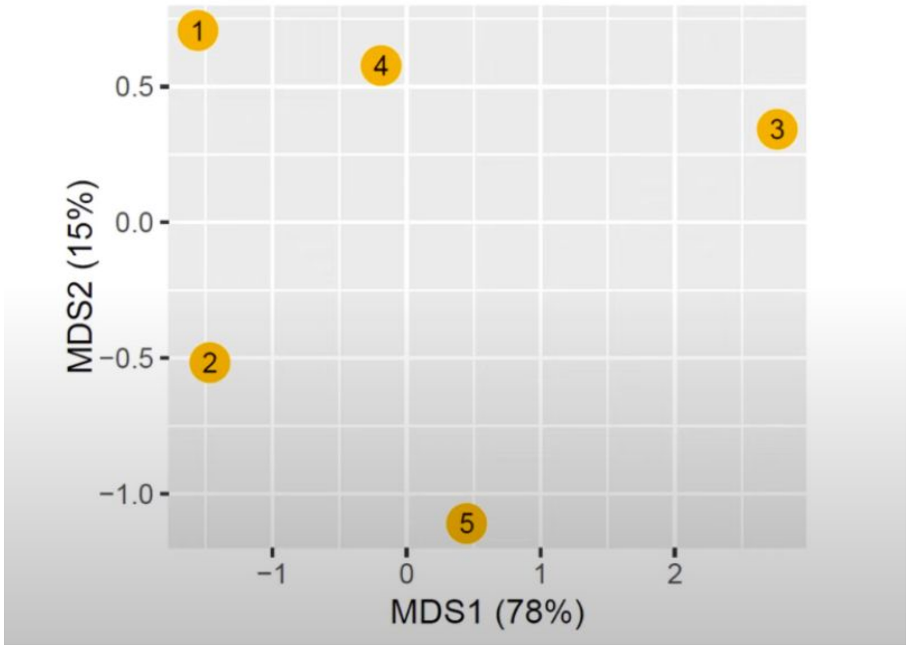
	1	2	3	4	5
1	0	1.426	4.356	1.463	2.741
2	1.426	0	4.327	2.051	2.314
3	4.356	4.327	0	3.093	2.866
4	1.463	2.051	3.093	0	1.809
5	2.741	2.314	2.866	1.809	0

SVD

What is captured in PCoA

Distance matrix

	1	2	3	4	5
1	0	1.426	4.356	1.463	2.741
2	1.426	0	4.327	2.051	2.314
3	4.356	4.327	0	3.093	2.866
4	1.463	2.051	3.093	0	1.809
5	2.741	2.314	2.866	1.809	0



PCA v/s PCoA

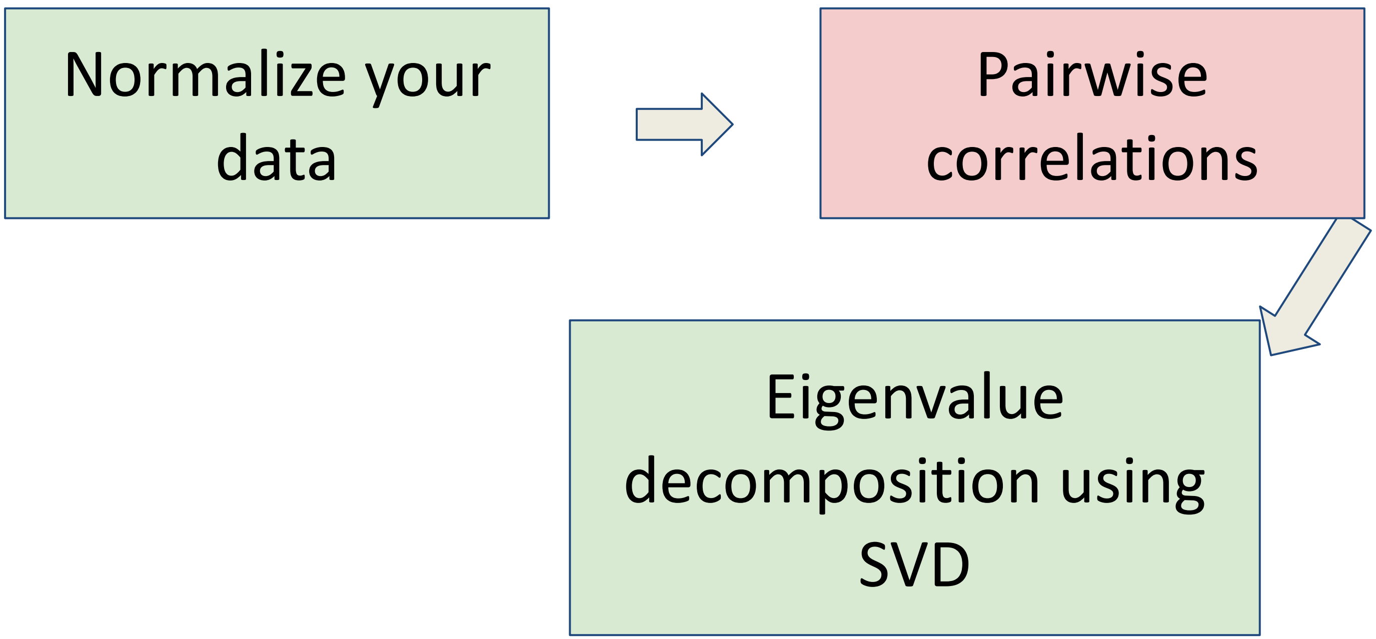
PCA	PCoA
A coordinate system that preserves variance	A coordinate system that preserves similarity

Today

- Dimensionality reduction
  - PCoA
  - CCA
- Bag of Words
- Latent Semantic Analysis

Canonical Correlation Analysis (CCA)

- CCA focuses more on correlations between samples.



Canonical Correlation Analysis (CCA)

- X and Y denote random variables represented as column vectors such that

$$X = (x_1, \ldots, x_n)^T \qquad Y = (y_1, \ldots, y_m)^T$$

- Define correlation matrix

$$\Sigma_{XY} = \mathrm{cov}(X, Y)$$

- Goal is to learn axes **u** and **v** such that:

$$\underset{u,v}{\mathrm{maximize}}\ \mathrm{corr}(\{u^T x, v^T y\})$$

- Let ($\mathbf{u^1}$,$\mathbf{v^1}$) are the first set of canonical axes.
- For ($\mathbf{u^2}$,$\mathbf{v^2}$): two constraints:
  - maximize correlations
  - be uncorrelated with ($\mathbf{u^1}$,$\mathbf{v^1}$)

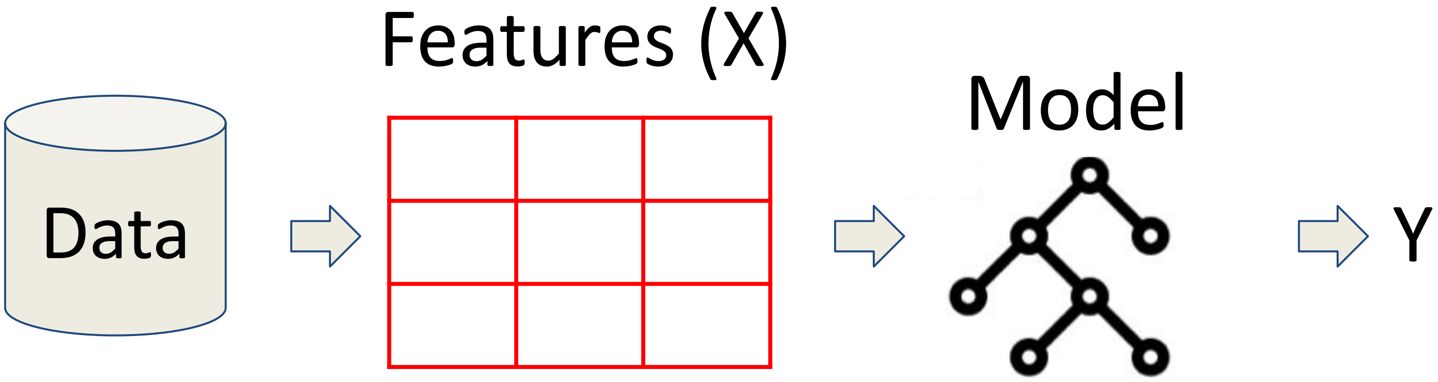
PCA v/s PCoA v/s CCA

Principal Component Analysis **(PCA)**	Principal Coordinate Analysis **(PCoA)**	Canonical correlation analysis **(CCA)**
A coordinate system that captures variance	A coordinate system that captures distance	A coordinate system that captures correlation

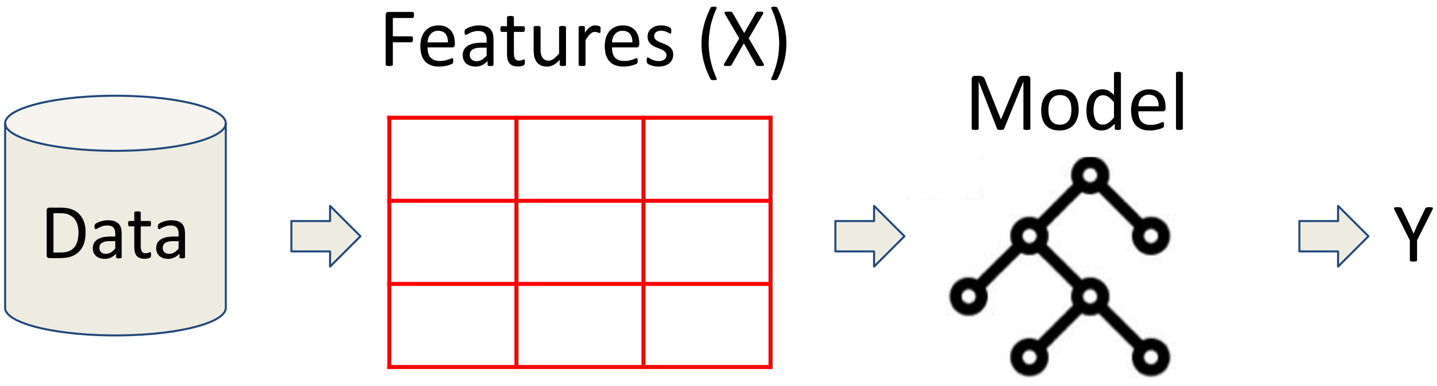
Today

- Dimensionality Reduction
- Feature representation
  - Bag of Words
- Latent Semantic Analysis

Standard ML pipeline



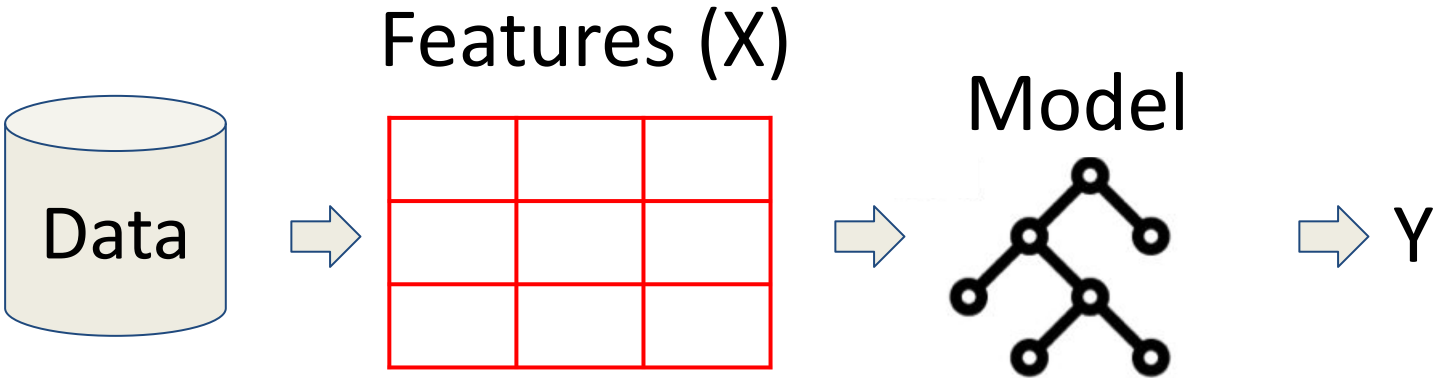
Standard ML pipeline



Example features

Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

Standard ML pipeline

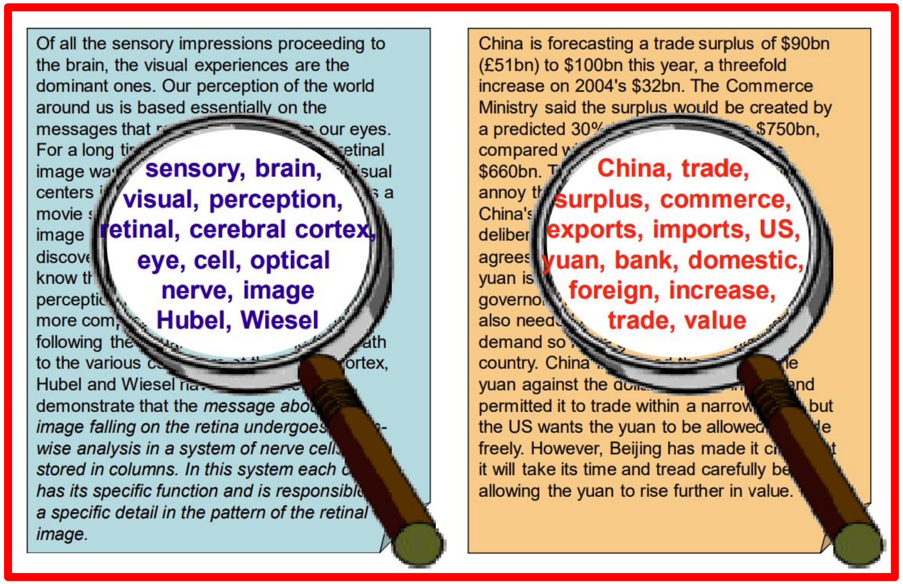


Example features

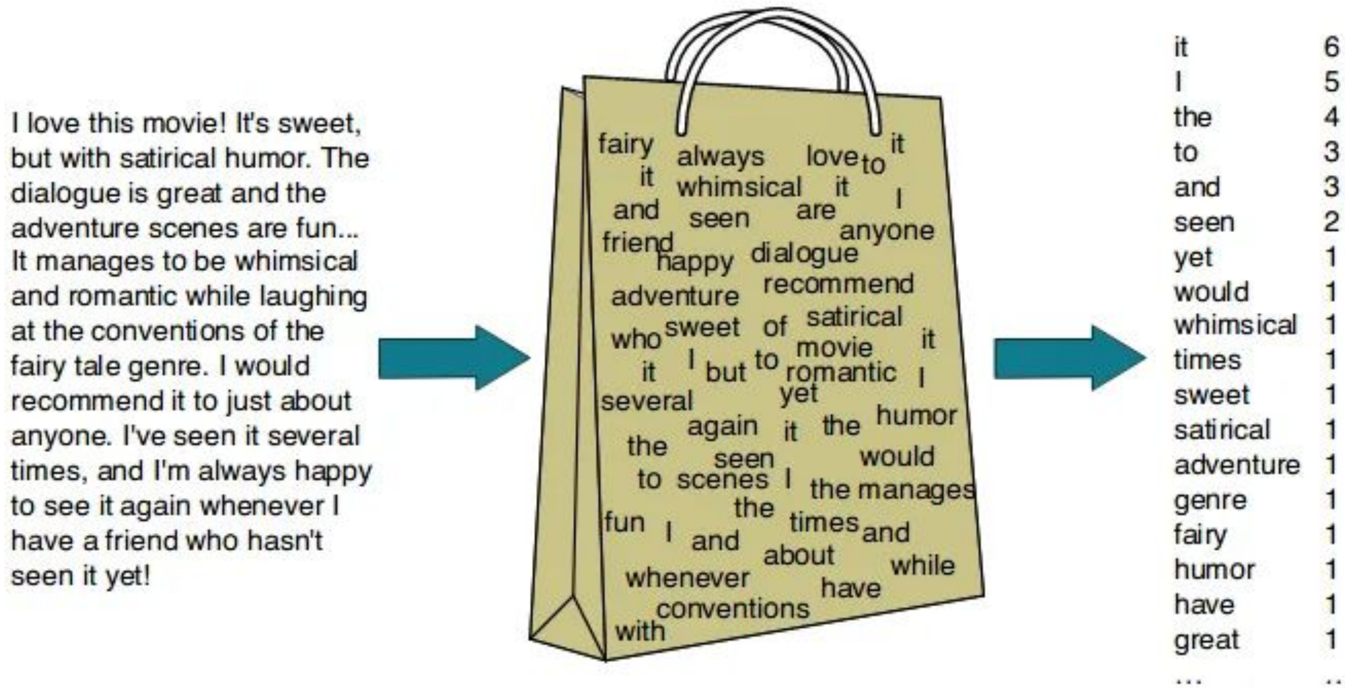
Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

Sunny = 0
Overcast = 1
Rainy = 2

How to represent text sentences as features?



Bag of words



Most fundamental representation of text sentences

BoW: Step1: Tokenization

- remove punctuations, make everything lowercase
- “The” with capital t and the token “the” with small t are mapped to the same token at vocab.

Image credit:mlarchive.com

BoW: Step-2: Vocabulary construction

Go through your entire dataset and index each unique word

the	br	fox	jum	ove	laz	dog	oov	whit	cat

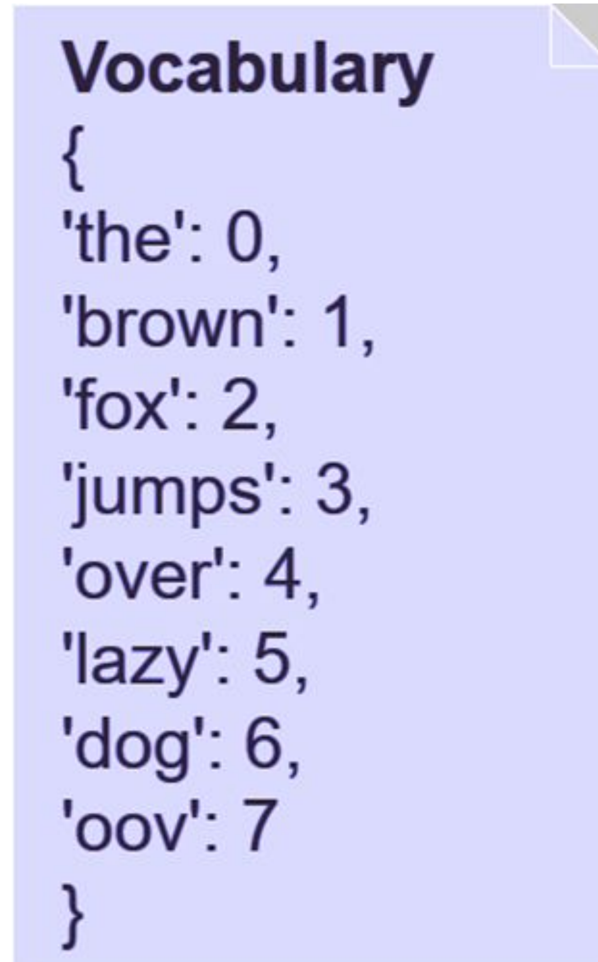


Image credit:mlarchive.com

BoW: Step-3: Featurize each sentence

Word	Appearance count	Index
the	2	0
brown	1	1
fox	1	2
jumps	1	3
over	1	4
lazy	1	5
dog	1	6
oov	0	7

the	br	fox	jum	ove	laz	dog	oov	whit	cat
2	1	1	1	1	1	1	0	0	0

Image credit:mlarchive.com

BoW: Step-3: Featurize each sentence

The White cat jumps over the lazy dog

the	br	fox	jum	ove	laz	dog	oov	whit	cat
2	0	0	1	1	1	1	0	1	1

The brown fox jumps over the lazy dog

the	br	fox	jum	ove	laz	dog	oov	whit	cat
2	1	1	1	1	1	1	0	0	0

Summary: BoW

1. Tokenize
2. Construct vocabulary
3. Featurize each sentence

Application#1: Sentence representation

The White cat jumps over the lazy dog

the	br	fox	jum	ove	laz	dog	oov	whit	cat
2	0	0	1	1	1	1	0	1	1

The brown fox jumps over the lazy dog

the	br	fox	jum	ove	laz	dog	oov	whit	cat
2	1	1	1	1	1	1	0	0	0

Application#2: Relevant document retrieval

- Let x_i, x_j be two vectors, their cosine distance would be

$$d_{ij} = \frac{x_i^T x_j}{\|x_i\| \, \|x_j\|}$$

- Sometimes referred to as cosine similarity
- $2 - d_{ij}$ is also sometimes used

What are some downsides of BoWs representation? Select all that apply.

What are some downsides of BoWs representation? Select all that apply.

Some words are over-represented leading to a skewed vector ✓

80%

Sparse feature representations lead to more memory usage

87%

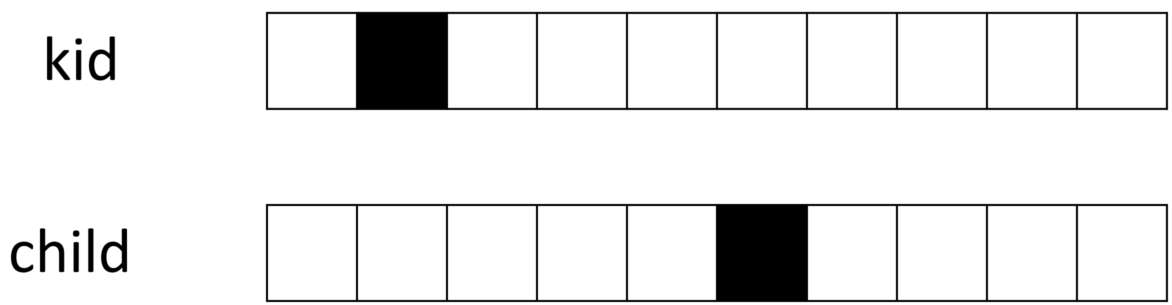
There is a loss of semantic meaning ✓

77%

There is a loss of spatial ordering ✓

72%

Challenge: Semantically Similar Words



Zipf's Law

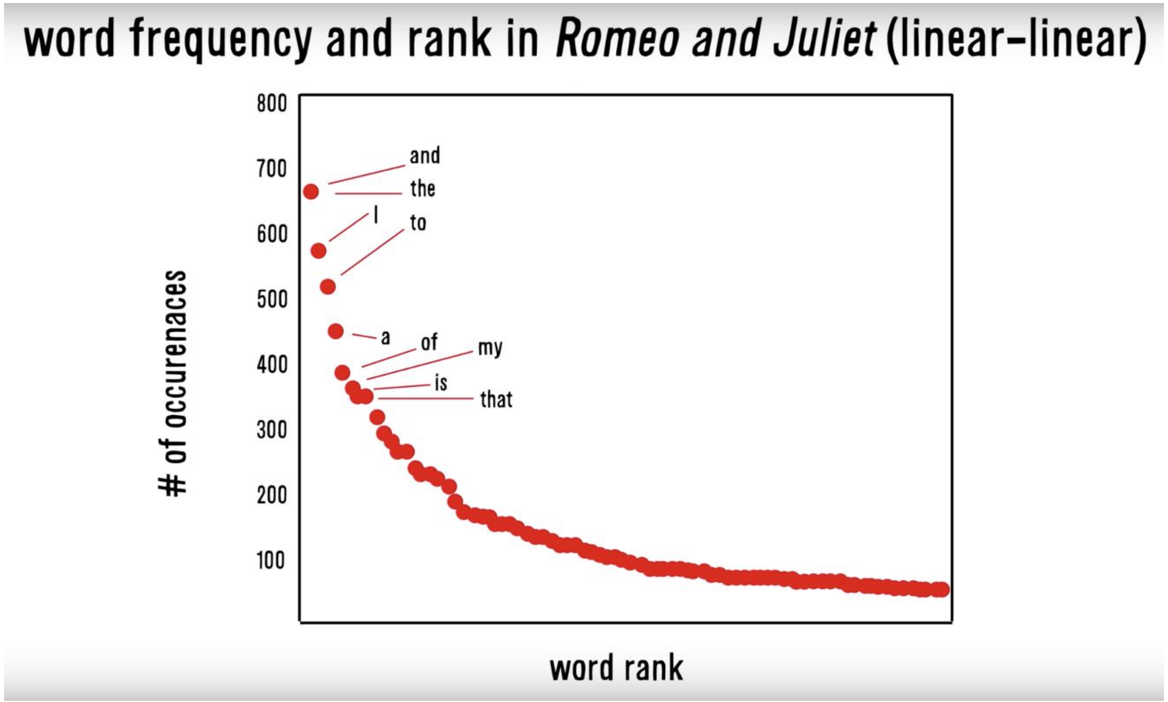
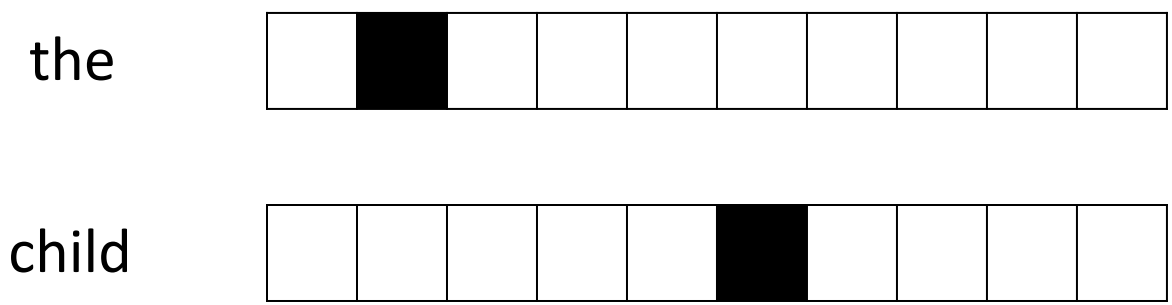


image credit:
https://medium.com/datadriveninvestor/zipfs-law-breakdown-application-in-app-development-5e9cda70cdc8

Challenge: Not all words are meaningful



Term frequency-inverse document frequency (TF-IDF)

- Let c_{ij} be the number of times the ith word appears in the jth document
- N documents
- N_i documents with at least one instance of the ith word

TF-IDF score:

$$c_{ij} \log \frac{N}{N_i}$$

Document-term matrix

- Let x_i be the bag-of-words vector for document i, the Document-term matrix is,

$$\mathcal{X} = \begin{bmatrix} x_1^T \\ \dots \\ x_N^T \end{bmatrix}$$

- $\mathcal{X}^T$ = term-document matrix

Latent Semantic Analysis

1. Take the SVD of $\mathcal{X} = \mathcal{U}\Sigma\mathcal{V}^T$
2. Let Σ_r be the matrix resulting from $\Sigma[r:, r:] = 0$ (note singular value ordering)
3. Return $X^{(r)} = \mathcal{U}\Sigma_r\mathcal{V}^T$

- Effectively smooths out word counts
- Typically followed by unit-normalizing each document

Latent Semantic Analysis

Pros
Addresses the curse of dimensionality
Hopefully the semantic closeness emerges (unsure) in this new feature space
SVD is super fast

Latent Semantic Analysis

Pros	Cons
Addresses the curse of dimensionality	Lacks interpretability
Hopefully the semantic closeness emerges (unsure) in this new feature space	
SVD is super fast	

What are the benefits of building a vocabulary on bigrams (e.g., {"brown fox", "blue sky"}) instead of unigrams (e.g., {brown, fox})?

What are the benefits of building a vocabulary on bigrams (e.g., {"brown fox", "blue sky"}) instead of unigrams (e.g., {brown, fox})?

Bigrams help capture the dependency with the nearby words building a more semantically meaningful vocabulary ✓

92%

Bigrams lead to an exponential increase in the vocabulary size.

3%

Bigrams contribute to significantly sparser feature vectors

5%

Why bother to study about sentence tokenization?

- Laid foundations for defining vector representations of words.
- Basic building blocks:
  - Tokenize words
  - Learn dependency between them.

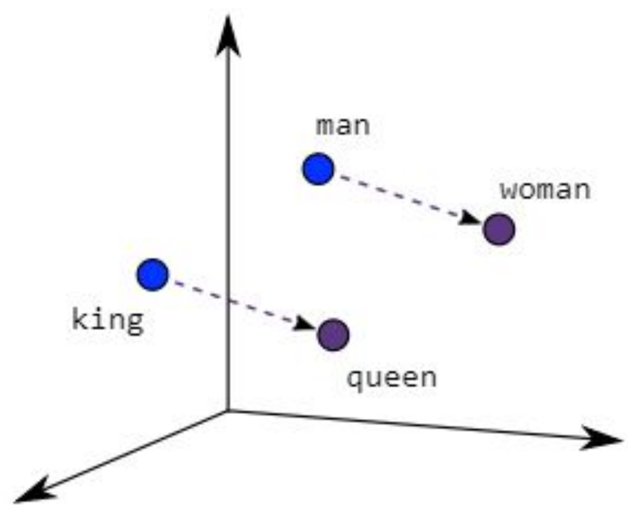
Why bother to study about sentence tokenization?

BoW
(unigrams,
bigrams, n-grams)

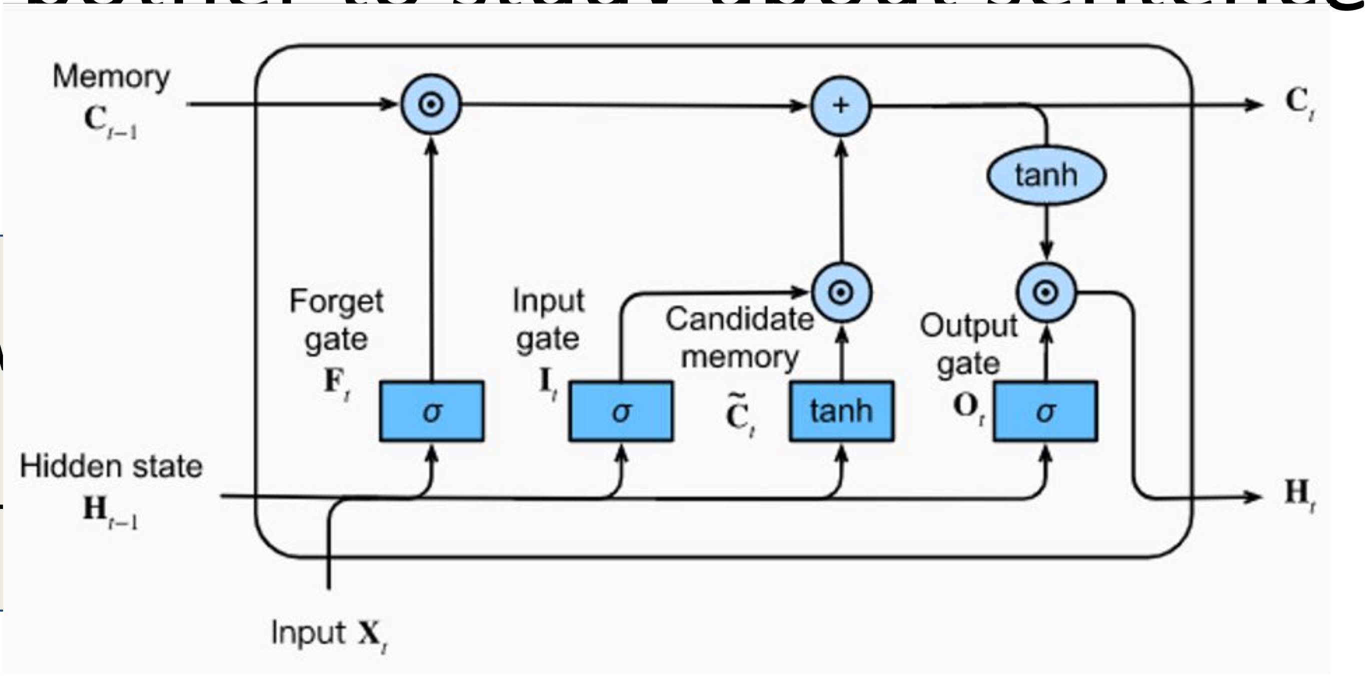
Why bother to study about sentence tokenization?

BoW
(unigrams, bigrams, n-grams)

word2vec
(Captures meaning of the word based on the surrounding words)



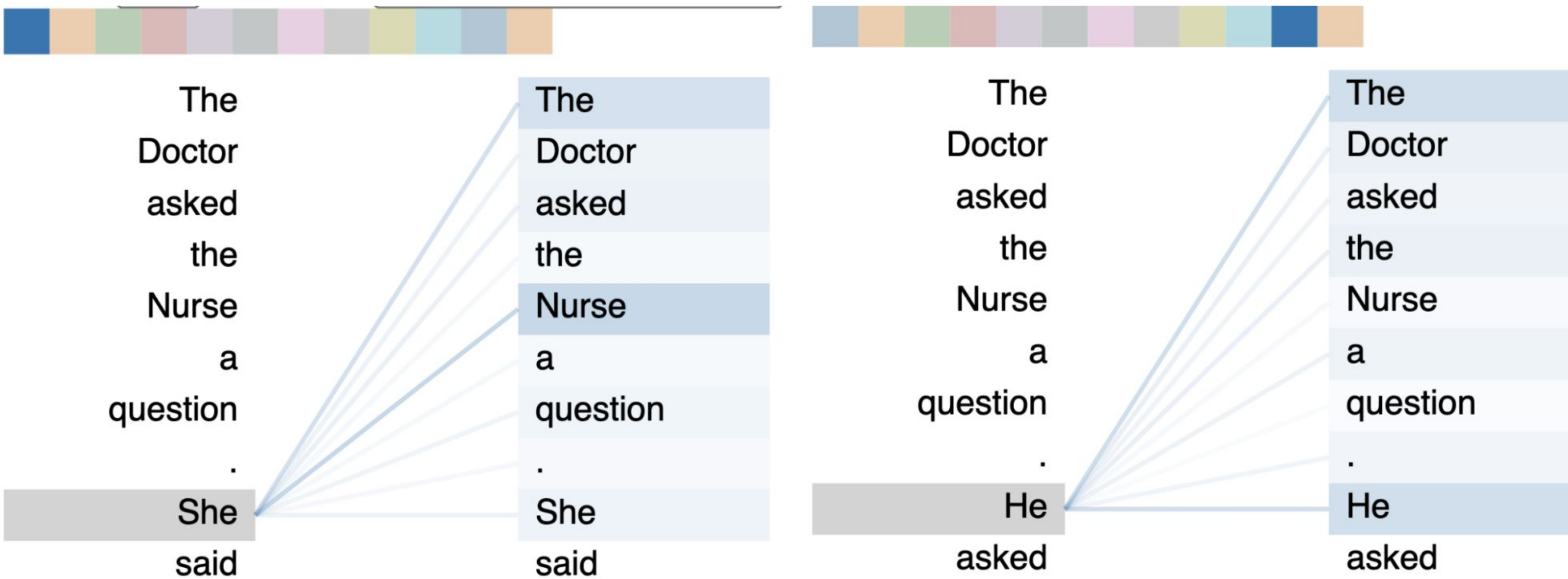
Why bother to study about sentence



BoW
(unigra
bigrams, n-

LSTM
(Has memory unit to remember relevant words)

Why bother to study about sentence tokenization?



Transformers

LSTM
(Has memory unit to remember relevant words)

Is context understanding a solved problem?

- **Active**: "A table without any bottle on it."
- **Passive**: “An empty table”
- **Random active**: “bottle table without it on a”
- **Random passive**: “ table empty an”

Is context understanding a solved problem?

- **Active**: "A table without any bottle on it."
- **Passive**: “An empty table”
- **Random active**: “bottle table without it on a”
- **Random passive**: “ table empty an”

	CLIP text similarity
(Active, passive)	0.9227
(Active, random passive)	0.9032
(random Active, passive)	0.8650
(random Active, random passive)	0.8780
(Active, random active)	0.9028
(Passive, random passive)	0.9624

Is context understanding a solved problem?

- **Active**: "A table without any bottle on it."
- **Passive**: “An empty table”
- **Random active**: “bottle table without it on a”

Takeaway: Most advanced text encoders still embed the sentence in a bag-of-words manner - leading to these high similarities

(Active, random passive)	0.9032
(random Active, passive)	0.8650
(random Active, random passive)	0.8780
(Active, random active)	0.9028
(Passive, random passive)	0.9624

How to represent images as features?

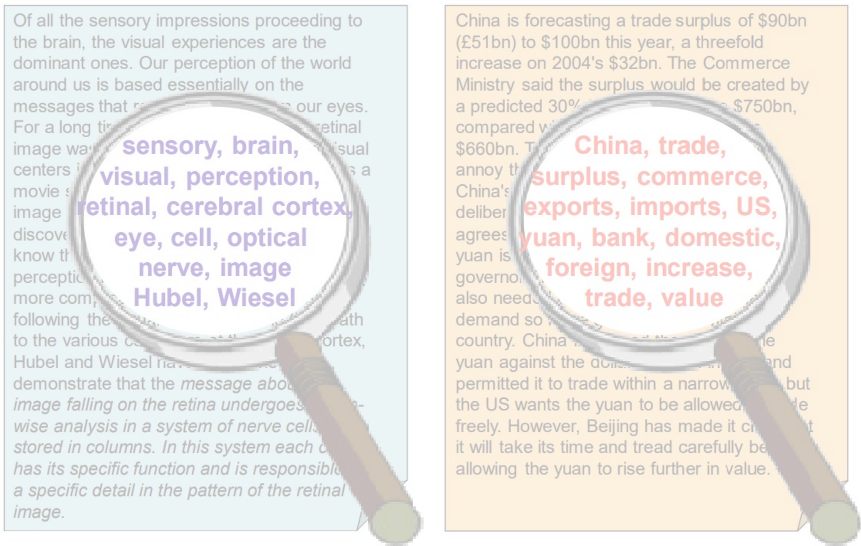


Image credit:Kristen Grauman

Visual bag of words

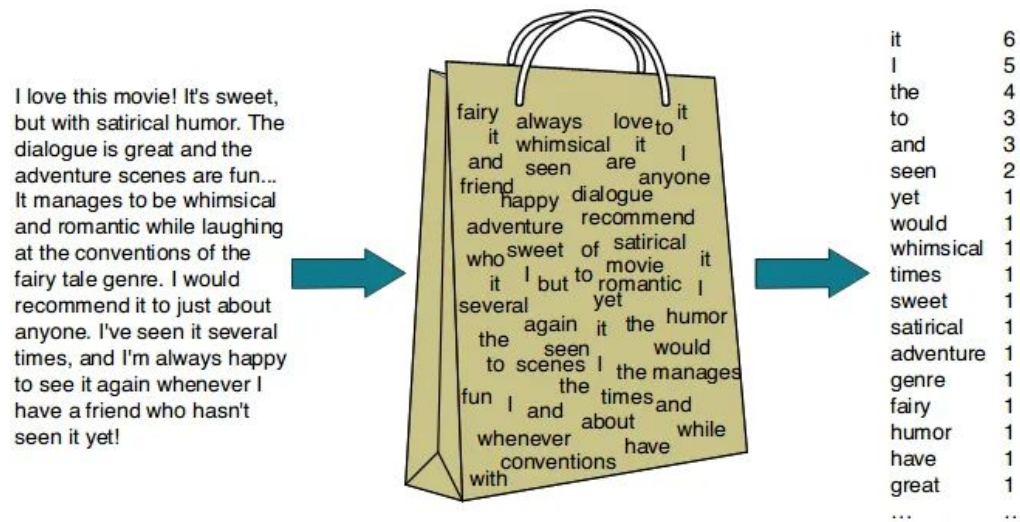


Image credit:Kristen Grauman

Visual BoW: Step1: Tokenization

- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)

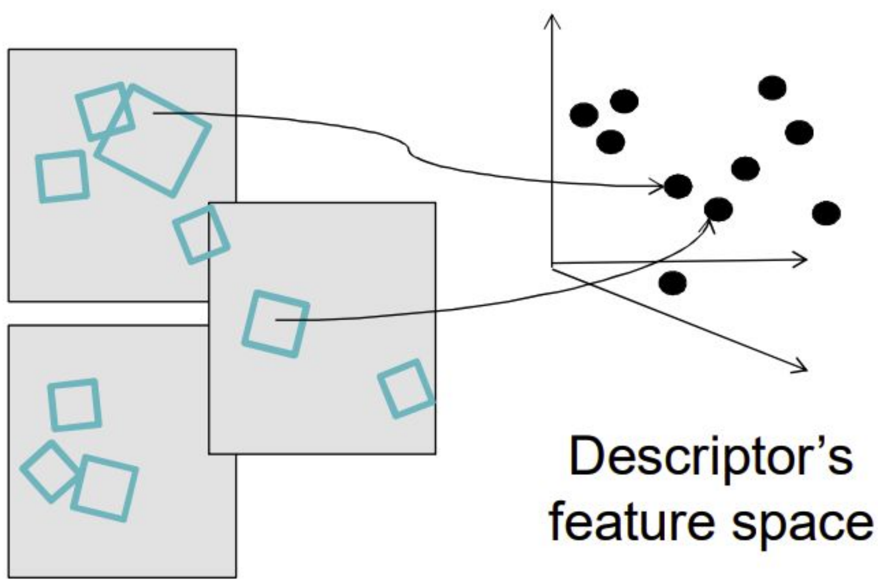
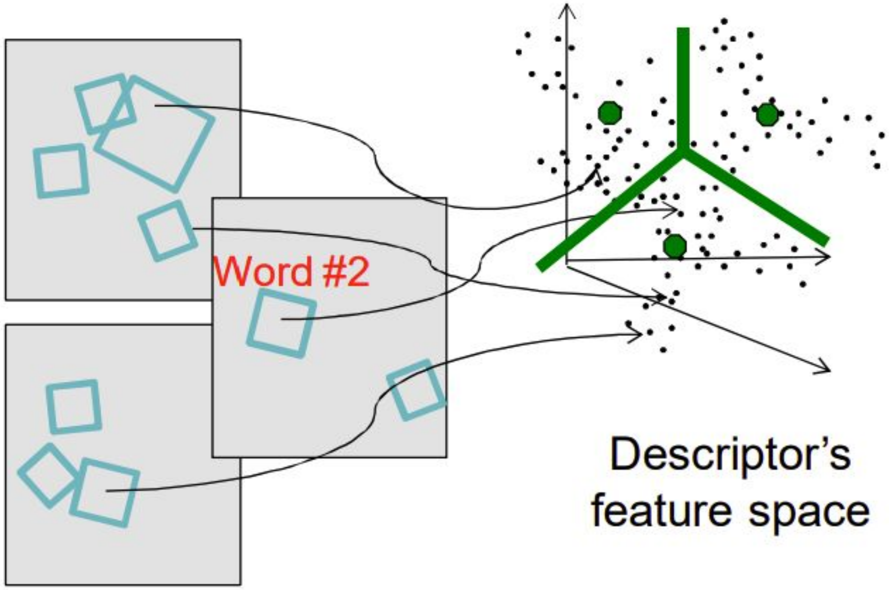


Image credit:Kristen Grauman

Visual words

- Map high-dimensional descriptors to tokens/words by quantizing the feature space



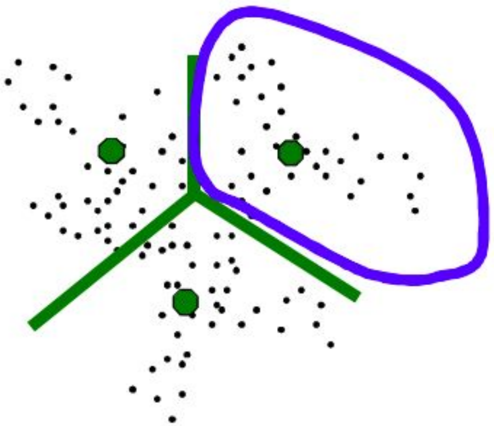
- Quantize via clustering, let cluster centers be the prototype "words"
- Determine which word to assign to each new image region by finding the closest cluster center.

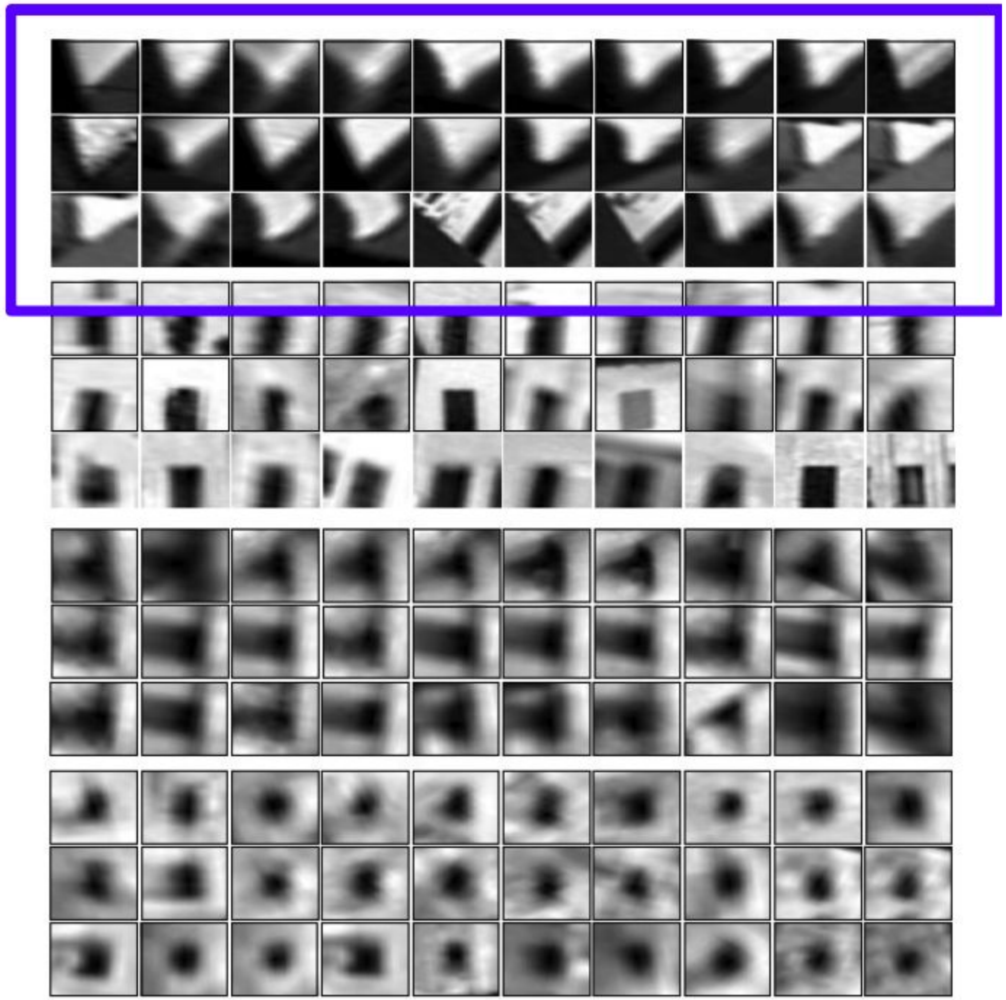
sten Grauman

Image credit:Kristen Grauman

Visual words

- Example: each group of patches belongs to the same visual word





Kristen Grauman

Figure from Sivic & Zisserman, ICCV 2003

BoW: Step-2: Vocabulary construction

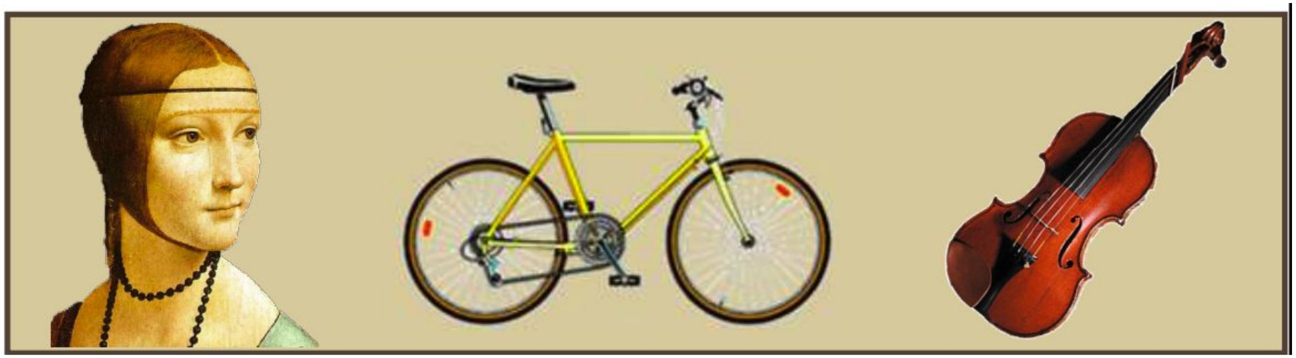


Image credit:Kristen Grauman

BoW: Step-3: Featurize each ~~sentence~~ image

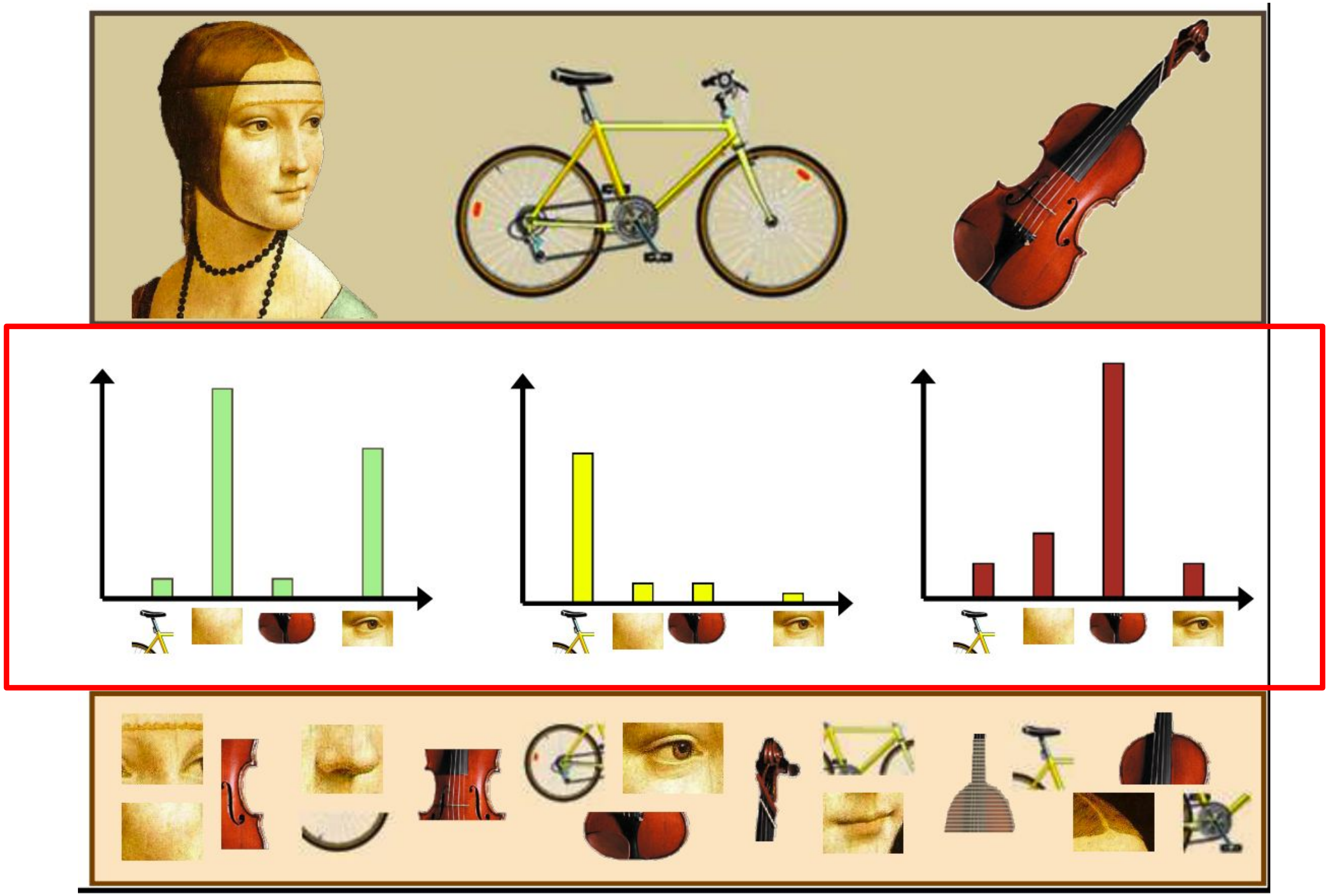
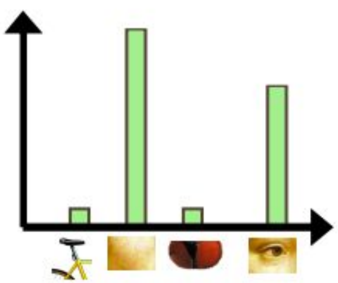


Image credit:Kristen Grauman

Comparing bags of words

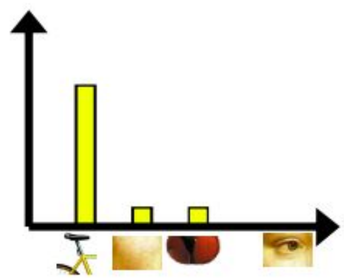
- Rank frames by normalized scalar product between their (possibly weighted) occurrence counts---*nearest neighbor* search for similar images.

[1 8 1 4]



$\vec{d}_j$

[5 1 1 0]



$\vec{q}$

$$sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

$$= \frac{\sum_{i=1}^{V} d_j(i) * q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} * \sqrt{\sum_{i=1}^{V} q(i)^2}}$$

for vocabulary of *V* words

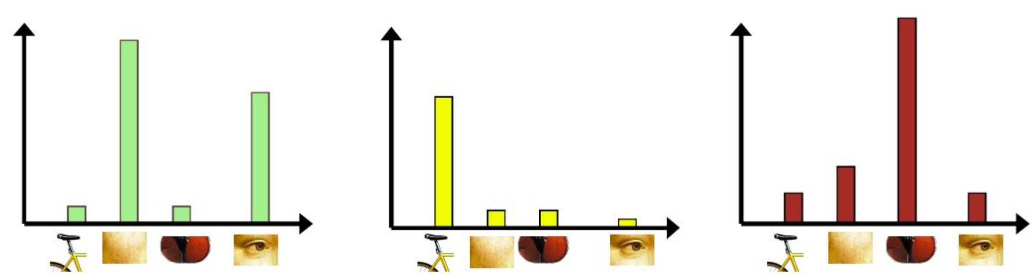
Slide credit:Kristen Grauman

Language vs vision

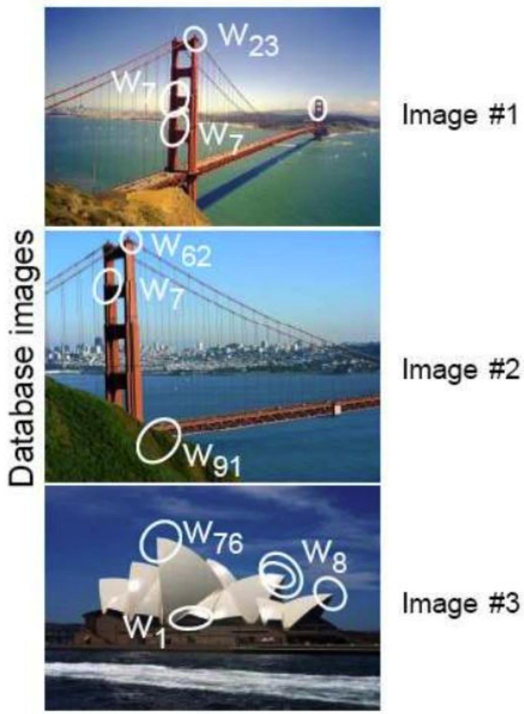
Vocabulary
{
'the': 0,
'brown': 1,
'fox': 2,
'jumps': 3,
'over': 4,
'lazy': 5,
'dog': 6,
'oov': 7
}

Feat representation

2	0	0	1	1	1	1	0	1	1



Application: Image retreival



Word #	Image #
1	3
2 …	
7	1, 2
8	3
9	
10 …	
91	2

…

Vocabulary

```
{
'the': 0,
'brown': 1,
'fox': 2,
'jumps': 3,
'over': 4,
'lazy': 5,
'dog': 6,
'oov': 7
}
```

- Database images are loaded into the index mapping words to image numbers

Image credit:Kristen Grauman

Application: Image retreival

New query image

Word #	Image #
1	3
2	
7	1, 2
8	3
9	
10	
…	
91	2

- New query image is mapped to indices of database images that share a word.

Image credit:Kristen Grauman

Learning between multiple datasets

Teddy bears shopping for groceries in ancient Egypt

Input

Generative Model

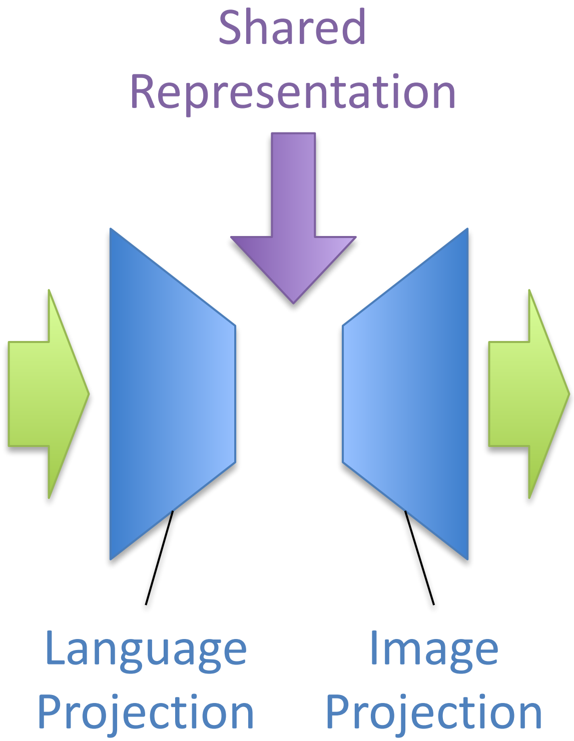
DALL·E 2

Output

Learning between multiple datasets

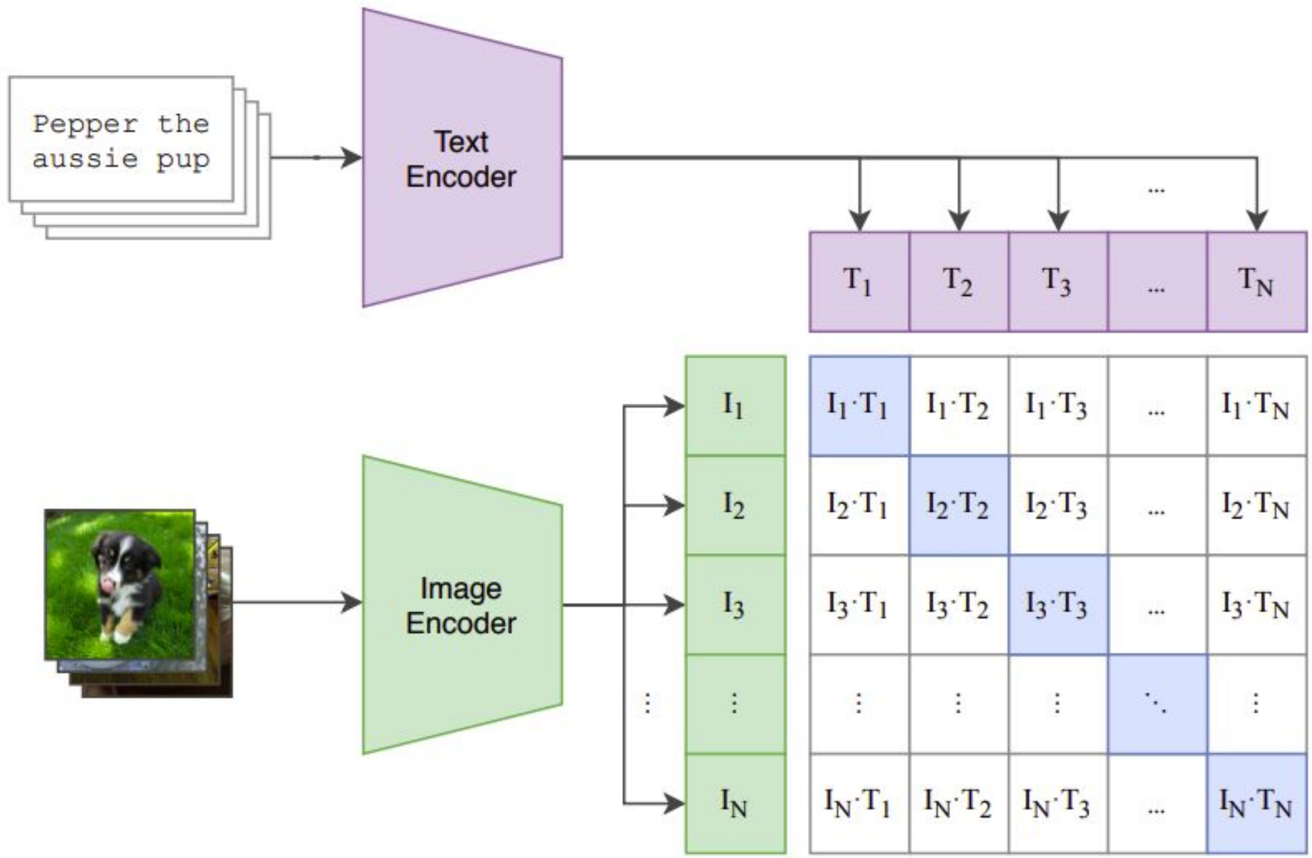
Teddy bears shopping for groceries in ancient Egypt

Input



Output

CLIP (Contrastive Language Image Pre-training)



Next Class

Clustering I: Agglomerative clustering, k-means

Reading: Forsyth Ch 8.1-8.2