UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Natural Language Processing

Vector Space Models

LESSON 10

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Some of the slides are taken from DeepLearning.AI and from the slides accompanying the textbook by D. Jurafsky

Distributional Hypothesis

- The role of context is important in the similarity of words
  - Words that occur in similar contexts tend to have similar meanings
- Distributional hypothesis
  - The link between similarity in how words are distributed and similarity in what they mean
    - Observation: words that are **synonyms** tend to occur in the same environment, with the amount of meaning difference between words "corresponding roughly to the amount of difference in their environment"

Word Similarity

- Words with similar meanings. Not synonyms, but sharing some element of meaning
  - *car, bicycle*
  - *cow, horse*

Motivation behind vector space models

- Suppose having two questions
  - Identical words except for the last ones
  - Different meaning

Where are you heading?

Where are you from?

Different meaning

- Consider two more questions
  - Completely different words
  - Same meaning

What is your age?

How old are you?

Same Meaning

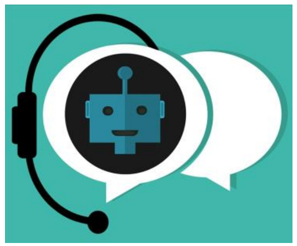
- Vector space models may help to
  - identify whether the first or second pair of questions are similar in meaning even if they do not share the same words
  - identify similarity for a question answering, and summarization
  - Allow capturing dependencies between words

Vector space models applications

- *You eat cereal from a bowl*
  - words cereal and bowl are related
- *You buy something and someone else sells it*
  - The second half of the sentence is dependent on the first half
- Vectors-based models capture these and many other types of relationships among different sets of words

Information Extraction

Machine Translation



Chatbots

Fundamental concept

- "You shall know a word by the company it keeps" (J.R. Firth, 1957)

- Vector space models
  - Represent words and documents as vectors
  - Representation that captures relative meaning
    - by identifying the context around each word in the text -> this captures the relative meaning!

What does *ongchoi* mean?

- Suppose you see these sentences:
  - Ongchoi is delicious **sautéed with garlic**
  - Ongchoi is superb **over rice**
  - Ongchoi **leaves** with salty sauces
- And you've also seen these:
  - …spinach **sautéed with garlic over rice**
  - Chard stems and **leaves** are **delicious**
  - Collard greens and other **salty** leafy greens
- Conclusion:
  - Ongchoi is a leafy green like spinach, chard, or collard greens
    - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

Vector Space Models

Word by Doc and Word by Word

Co-occurrence matrix

- Vector or distributional models of meaning are generally based on a co-occurrence matrix
- Vectors can be constructed from the co-occurrence matrix
  - A way of representing how often words co-occur
  - Co-occurrence -> Vector representation
- Depending on the task at hand, several possible designs exist
- Let's start with a *word-by-document* design
  - Each row represents a word in the vocabulary and each column represents a document from some collection

Word by Document Design

- Number of times a word occurs within a certain category

Corpus

	Entertainment	Economy	Machine Learning
data	500	6620	9320
film	7000	4000	1000

- For instance, entertainment category vector is v = [500,7000]
  - Categories can also be compared by doing a simple plot

Word-document matrix

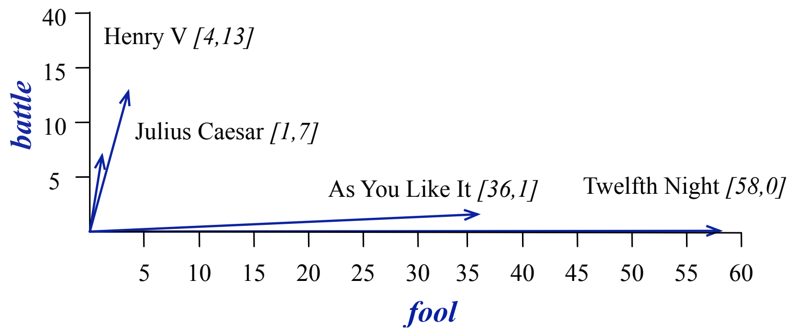
- Each document is represented as a vector of words

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Vectors are the basis of information retrieval

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Vectors are similar for two comedies
  - But comedies are different than the other two
    - Comedies have more fools and wit and fewer battles



Word meaning: Words can be vectors too!

- Battle is "*the kind of word that occurs in **Julius Caesar** and **Henry V***"
- Fool is "*the kind of word that occurs in comedies, especially **Twelfth Night***"

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Word-by-Word Design

- The co-occurrence of two different words is the number of times that they appear in a corpus together within a given word distance k (context)
- Assume that you are trying to construct a vector that will represent a certain word
- Create a matrix where each row and column corresponds to a word in the vocabulary
- Keep track of number of times they occur together within a certain distance k

I like simple data

I prefer simple raw data

k=2

	simple	raw	like	I
data	2	1	1	0

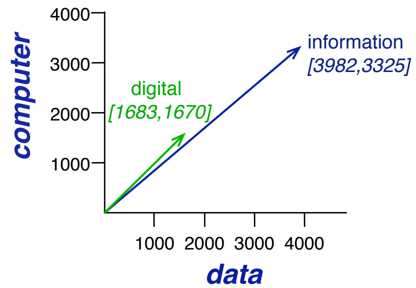
n

Word-Word matrix

- Two words are similar in meaning if their context vectors are similar

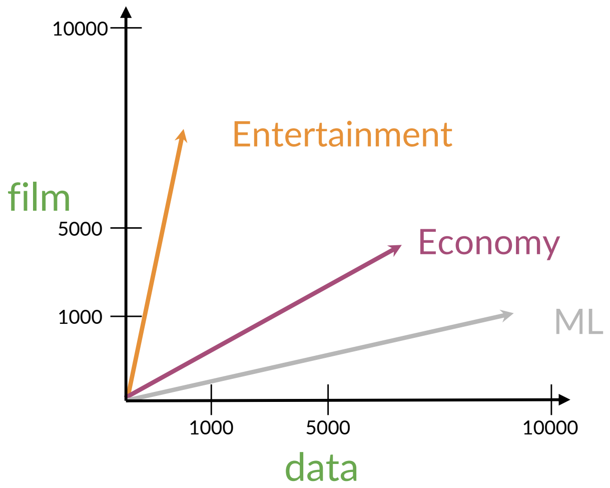
is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	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	...



Vector Space

- A representation of the words data and film is taken from the rows of the table
- Alternatively, the representation for every category of documents could be taken from the columns
  - 2D vector space

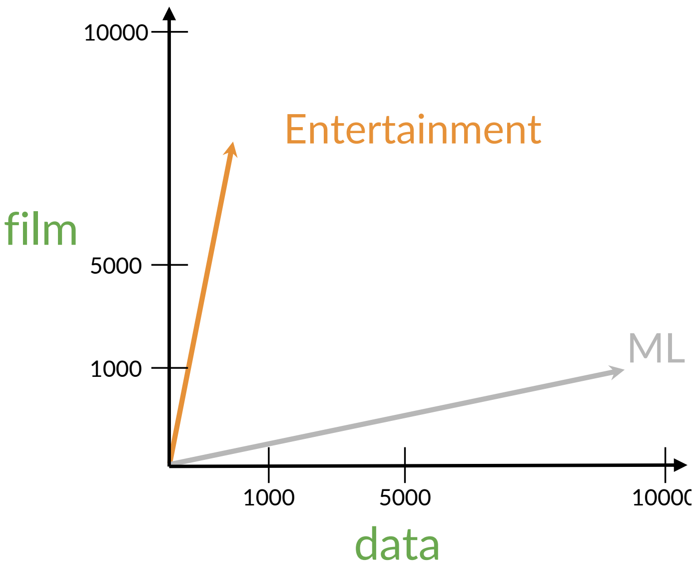


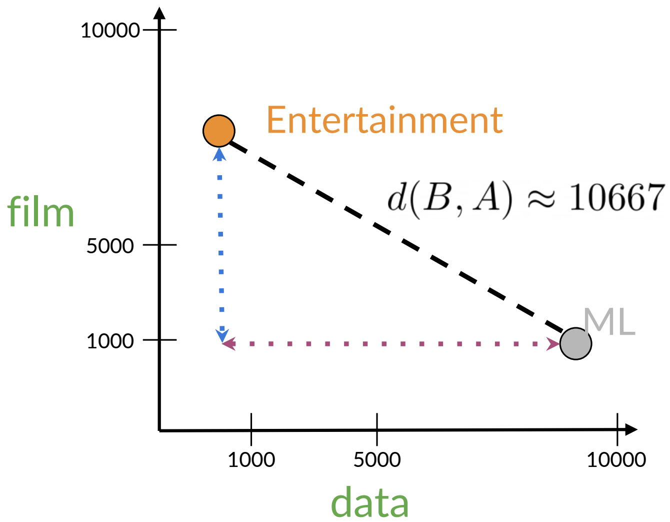
	Entertainment	Economy	ML
data	500	6620	9320
film	7000	4000	1000

Measures of “similarity:”
Angle
Distance

Comparing vectors: Euclidean distance

- Corpus A: Entertainment and Corpus B: Machine-Learning





Corpus A: (500,7000)

Corpus B: (9320,1000)

$$d(B, A) = \sqrt{(B_1 - A_1)^2 + (B_2 - A_2)^2}$$

$$c^2 = a^2 + b^2$$

$$d(B, A) = \sqrt{(-8820)^2 + (6000)^2}$$

Euclidean distance for n-dimensional vectors

- Example:
  - Euclidean distance between the vector *v* of the word ice cream and the vector representation *w* of the word the boba

	data	boba ($\vec{w}$)	ice-cream ($\vec{v}$)
AI	6	0	1
drinks	0	4	6
food	0	6	8

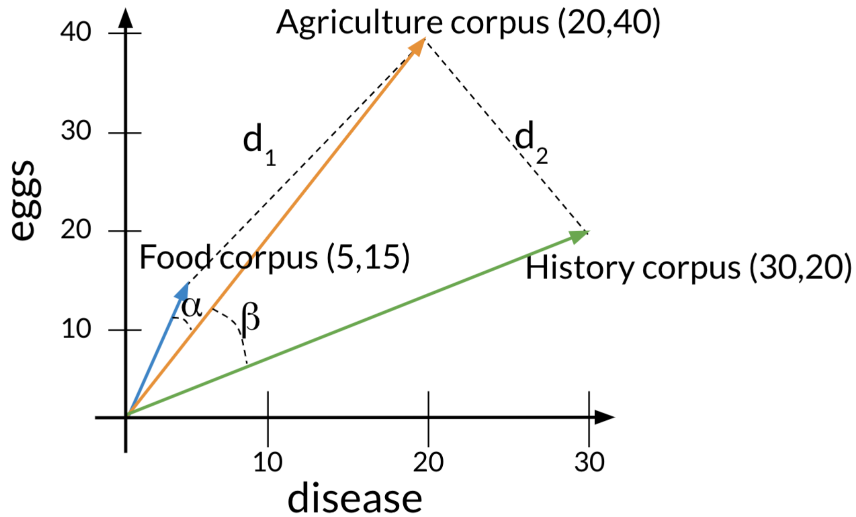
$$= \sqrt{(1-0)^2 + (6-4)^2 + (8-6)^2}$$

$$= \sqrt{1+4+4} = \sqrt{9} = 3$$

$$d(\vec{v}, \vec{w}) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2}.$$

Comparing vectors: Cosine similarity

- Euclidean distance is not always accurate
  - For instance, when comparing large documents to smaller ones
- Cosine similarity when corpora are of different sizes



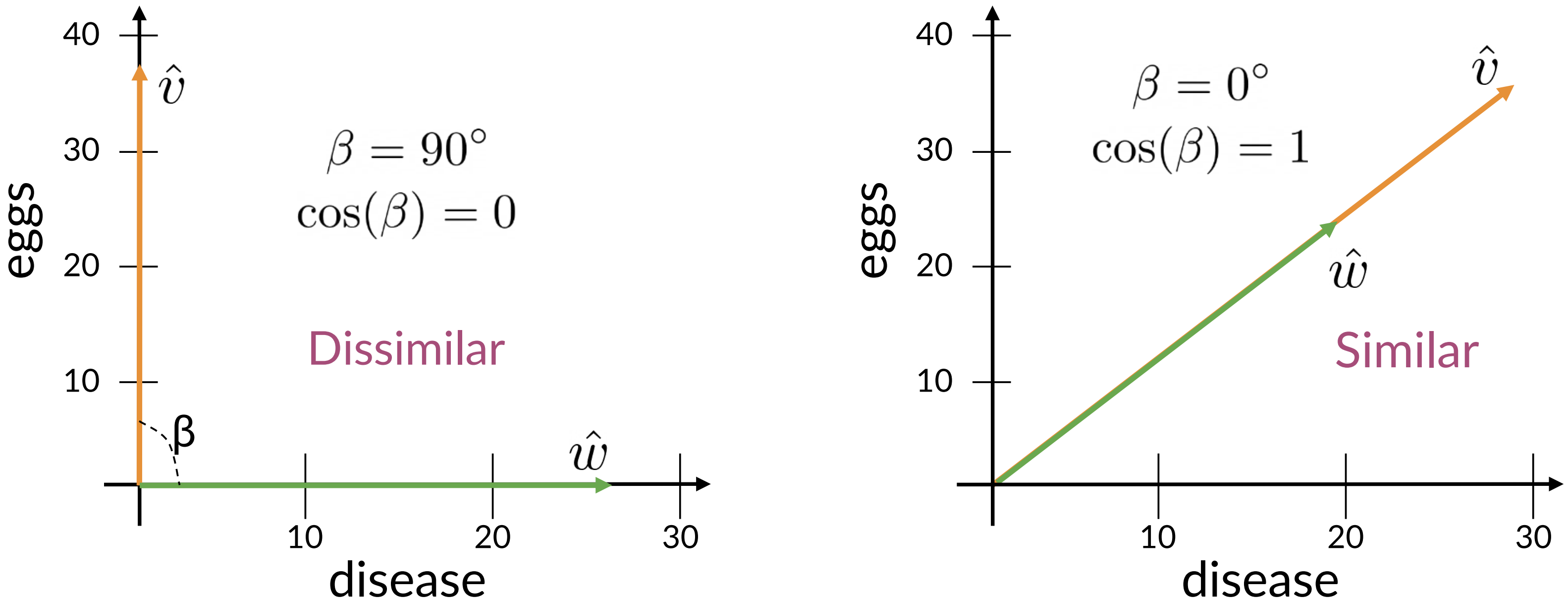
Euclidean distance: $d_2 < d_1$

Angles comparison: $\beta > \alpha$

The cosine of the angle between the vectors

$$\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\|\|\hat{w}\|}$$

Cosine Similarity



Raw Frequency is not the best representation

- The co-occurrence matrices we have seen represent each cell by word frequencies
- Frequency is useful
  - If sugar appears a lot near apricot, that's useful information
- But overly frequent words like the, it, or they are not very informative about the context
- It's a paradox: How can we balance these two conflicting constraints?

Two common solutions for word weighting

- tf-idf
  - tf-idf value for word **t** in document **d**: $w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$
  - Words like *the* or *it* have very low idf
- PMI (Pointwise mutual information):
  - $\textbf{PMI}(\boldsymbol{w_1}, \boldsymbol{w_2}) = \boldsymbol{log}\frac{\boldsymbol{p(w_1,w_2)}}{\boldsymbol{p(w_1)p(w_2)}}$
  - See if words like *good* appear more often with *great* than we would expect by chance

Term frequency

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

- Instead of using raw count, we squash a bit:

- $\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1)$

Document frequency

- df_t *is* the number of documents *t* occurs in.
- (note this is not collection frequency: total count across all documents)
- "*Romeo*" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse Document Frequency (IDF)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

- N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

What is a document?

- Could be a play or a Wikipedia article
- For the purpose of the tf-idf, documents can be anything
  - We often call each paragraph a document

Final tf-idf weighted value for a word

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

- Raw counts

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Tf-idf

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Pointwise Mutual Information

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

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

- PMI between two words: (Church & Hanks 1989)
  - Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora
    - Imagine w1 and w2 whose probability is each 10^{-6}
    - Hard to be sure p(w1,w2) is significantly different than 10^{-12}
- So we just replace negative PMI values by 0
- Positive PMI (**PPMI**) between word1 and word2:

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

Computing PPMI on a word-word matrix

- Matrix F with W rows (words) and C columns (contexts) f_{ij} is # of times w_i occurs in context c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

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

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Computing PPMI on a word-document matrix

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

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

- p(w=*information*,c=*data*) = 3982/111716 = .3399
- p(w=*information*) = 7703/11716 = .6575
- p(c=*data*) = 5673/11716 = .4842

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N} \qquad p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	**data**	**result**	**pie**	**sugar**	**p(w)**
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

Computing PPMI on a word-document matrix

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

	p(w,context)					p(w)
	computer	**data**	**result**	**pie**	**sugar**	**p(w)**
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

- PMI(*information*,*data*) = $\log_2$ (.3399 / (.6575*.4842)) = .0944

Resulting PPMI matrix (negatives replaced by 0)

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

Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values
- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

PPMI: Rare context words given a higher probability

- Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

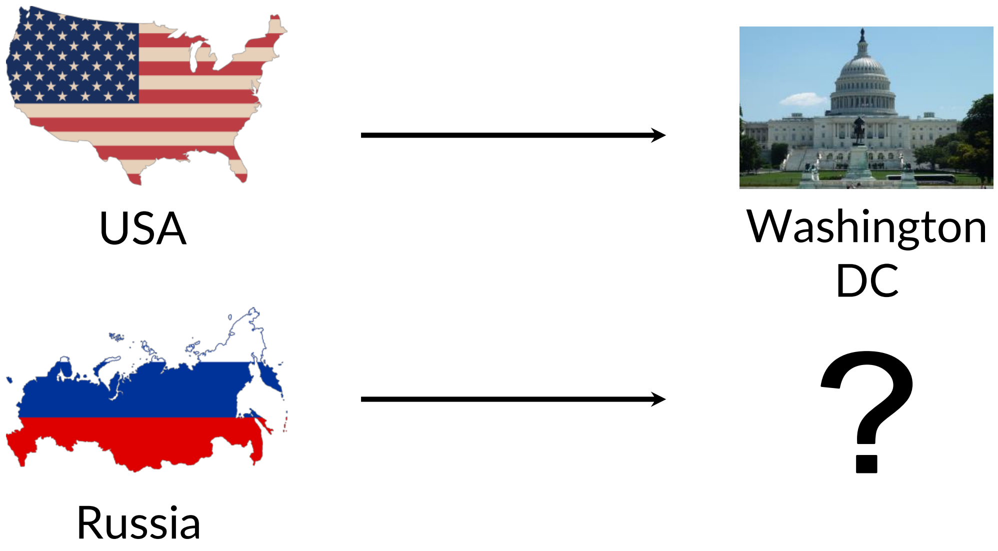
- This helps because $P_\alpha(c) > P(c)$ for rare c
- Consider two events, P(a) = .99 and P(b)=.01

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75}+.01^{.75}} = .97 \quad P_\alpha(b) = \frac{.01^{.75}}{.01^{.75}+.01^{.75}} = .03$$

Manipulating Words in Vector Spaces

Manipulating word vectors

- Suppose we have a vector space with countries and their capital cities
  - We know that Washington is the capital of the USA
  - We don't know the capital of Russia
  - Goal: infer the capital of Russia by the known relationship

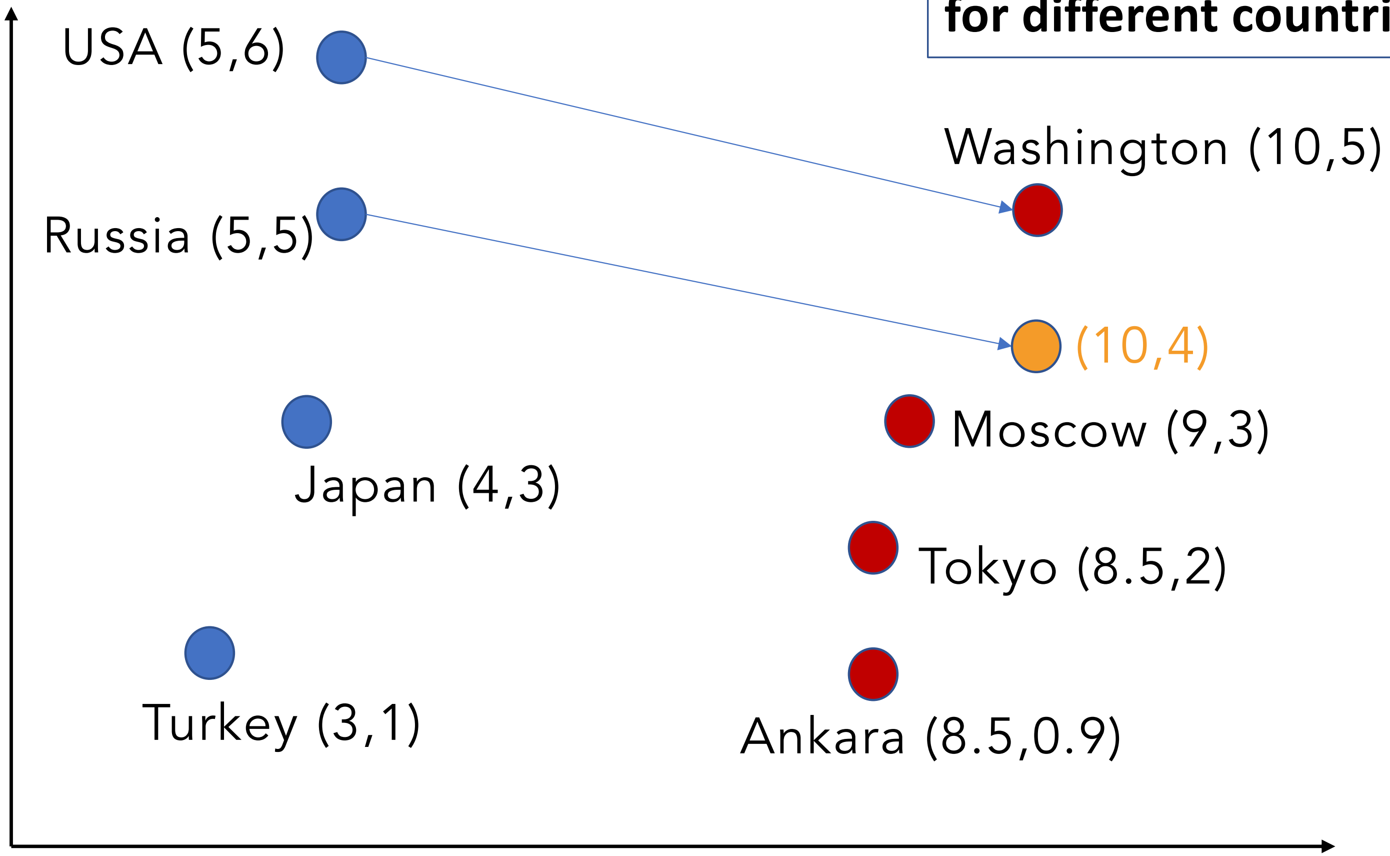


Manipulating word vectors

- Word vectors can be used to extract patterns and identify certain structures in our text
- Example
  - One can use the word vector for Russia, USA, and Washington DC to compute a vector that would be very similar to Moscow
  - Then use the cosine similarity of the **vector** with all the other word vectors and find that the vector of Moscow is the closest

Manipulating word vectors

2D vector space with different representations for different countries and capitals cities



Washington – USA = [5 -1]

Russia + [5 -1] = [10 4]

Moscow

Visualization of word vectors

- With vectors of very high dimensions PCA (or other dimensionality reduction techniques) can be used to plot vectors in 2D or 3D spaces

$d > 2$

oil	0.20	...	0.10
gas	2.10	...	3.40
city	9.30	...	52.1
town	6.20	...	34.3

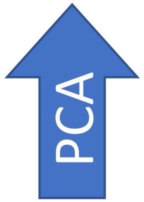
How can you visualize if your representation captures these relationships?

oil & gas

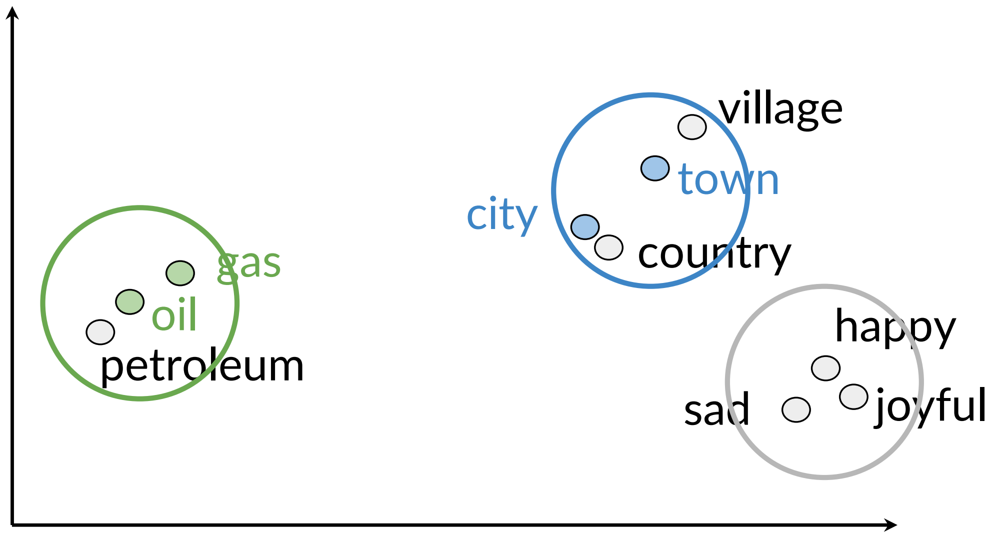
town & city

Visualization of word vectors

	$d = 2$	
oil	2.30	21.2
gas	1.56	19.3
city	13.4	34.1
town	15.6	29.8



	$d > 2$		
oil	0.20	...	0.10
gas	2.10	...	3.40
city	9.30	...	52.1
town	6.20	...	34.3



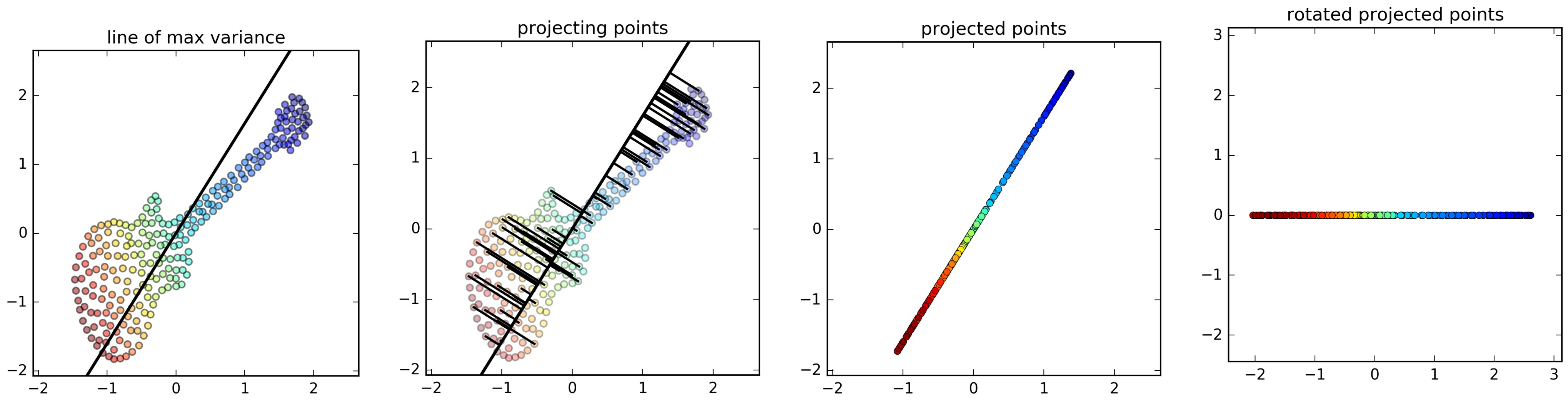
- Relationship between the words oil and gas and city and town
- In 2D space they appear to be clustered with related words
  - You can even find other relationships among your words that you didn't expect

Principal Component Analysis

- An unsupervised, deterministic algorithm used for feature extraction as well as visualization
- Applies a linear dimensionality reduction technique where the focus is on keeping the dissimilar points far apart in a lower-dimensional space
- Transforms the original data to new data by preserving the variance in the data using eigenvalues

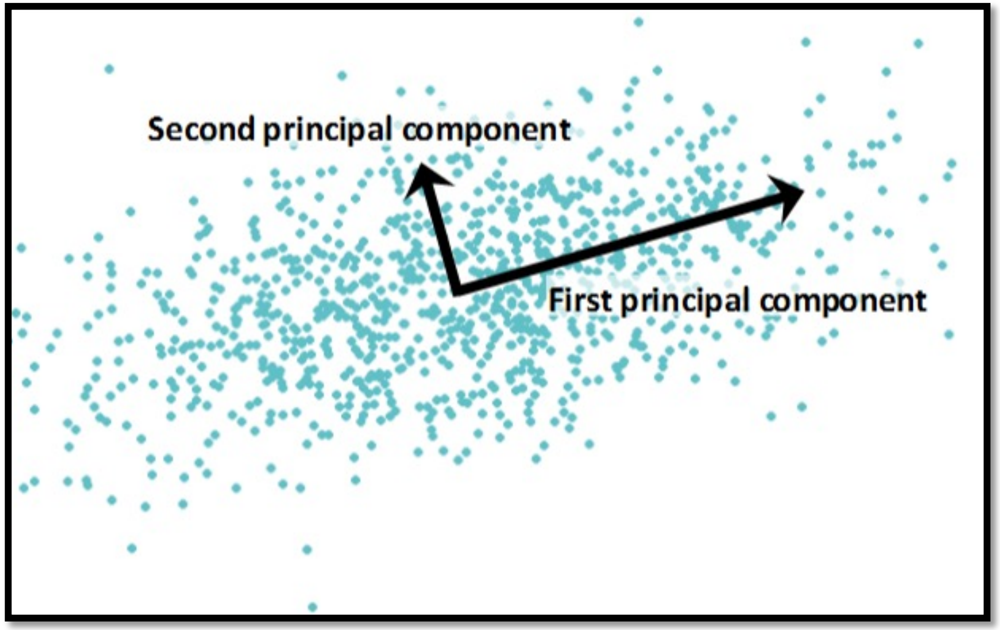
Principal component analysis (PCA)

- Here is how PCA proceeds

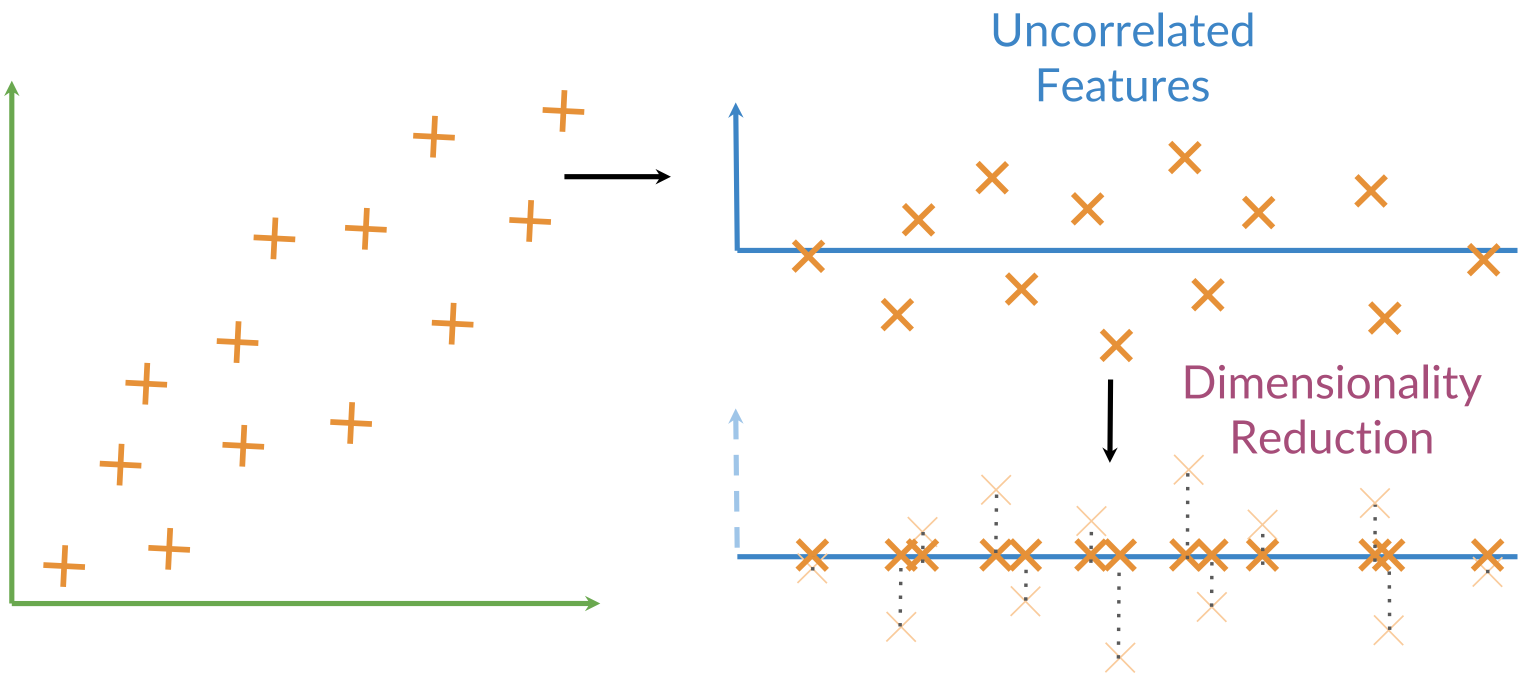


Principal Component Analysis

- Finds a new coordinate system such that few new axes captures the greatest variance
- Define lower-dimensional space for data
- Note
  - Original dimensions have a natural interpretation
    - E.g., Income, age, occupation, etc
  - New dimensions more difficult to interpret!
  - In general, there are as many principal components as original features

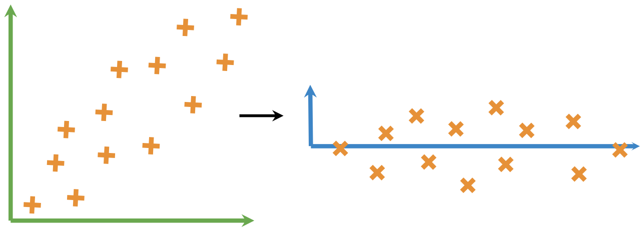


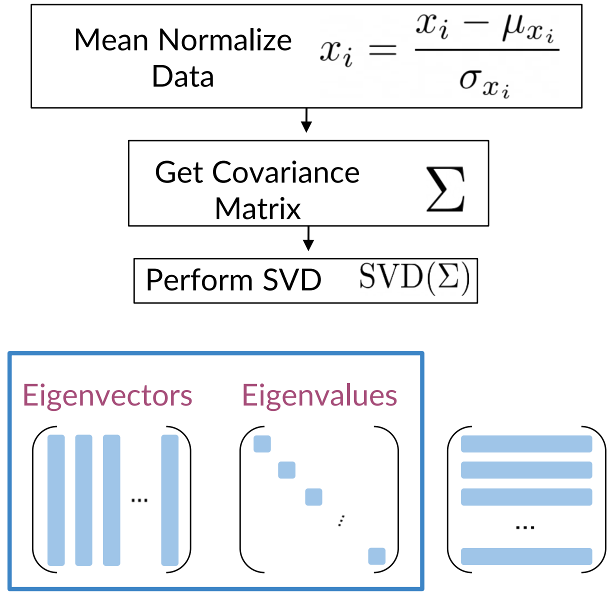
Principal Component Analysis



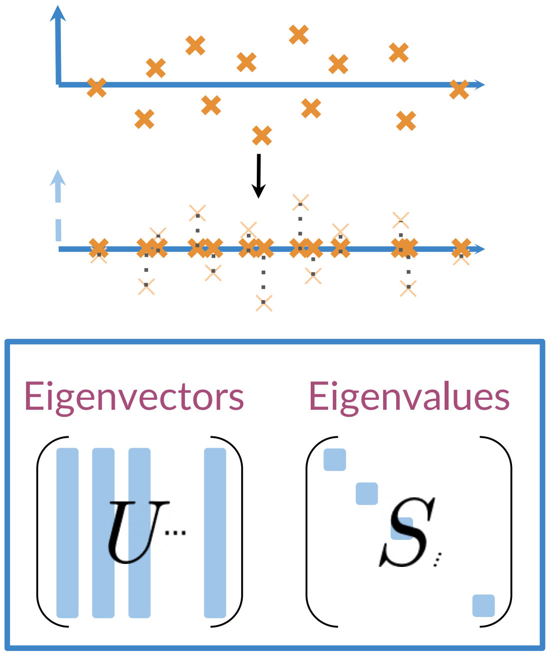
PCA Algorithm

- Eigenvector
  - Uncorrelated features of your data
- Eigenvalue
  - The amount of information retained by each feature





PCA Algorithm



Dot Product to Project Data

$$X' = XU[:, 0:2]$$

Percentage of Retained Variance

$$\frac{\sum_{i=0}^{1} S_{ii}}{\sum_{j=0}^{d} S_{jj}}$$

PCA Summary

- Eigenvectors give the direction of uncorrelated features
- Eigenvalues are the variance of the new features
- Dot product gives the projection on uncorrelated features

Assignment n. 2

- Prepare Jupiter notebooks for explaining Sentiment Analysis with Naïve Bayes and Logistic regression
  - Also, consider any preprocessing step
- It would be possible to use any python library for NLP e for Machine Learning, Data Analysis, and numerical computation (e.g., scikit-learn, Pandas, and NumPy)