Machine Learning and NLP: Advances and Applications
Day 2: NLP Basics

1/23/2020

Yoshi Suhara

Course Overview

- Goal: Learning ML/NLP basics and how to apply the techniques to your own problems

- The course will cover
  - both theory (lecture) and practice (hands-on)

Course Overview

- Day 1: Machine Learning Basics
- Day 2: NLP Basics
- Day 3: Advanced Techniques and Applications

Course Overview

- Day 1: Machine Learning Basics
  - **Hands-on material 1**
- Day 2: NLP Basics
  - **Hands-on material 2**
- Day 3: Advanced Techniques and Applications
  - **Hands-on material 3**

Day 1 **Day 2** **Day 3**

Recap: Course Overview

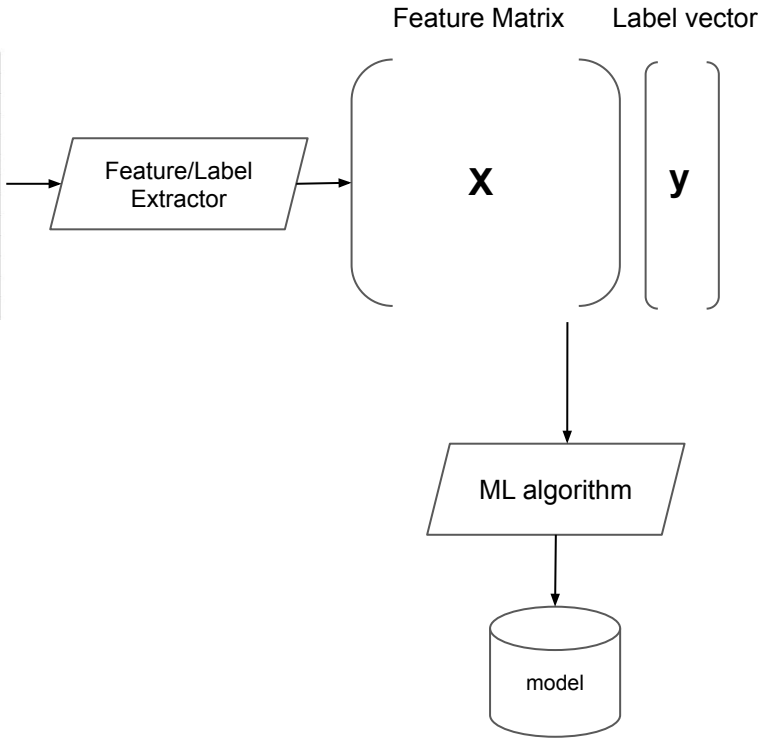
- Day 1: Machine Learning Basics
- Day 2: NLP Basics (+ ML Topics)
- Day 3: Advanced Techniques and Applications

Day 2

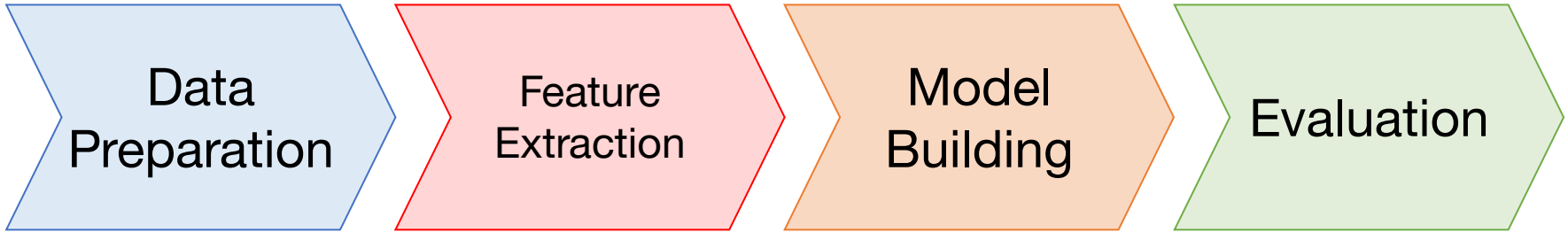
- Answers to Questions
- Unsupervised Learning
- Topic Models
- NLP Basics
- Further topics
- Hands-on Part

Recap: ML Workflow

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S



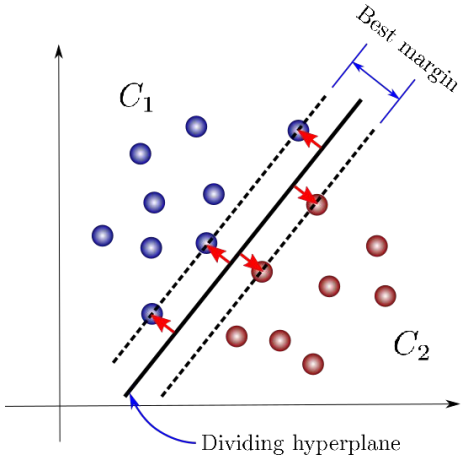
Recap: Machine Learning Pipeline

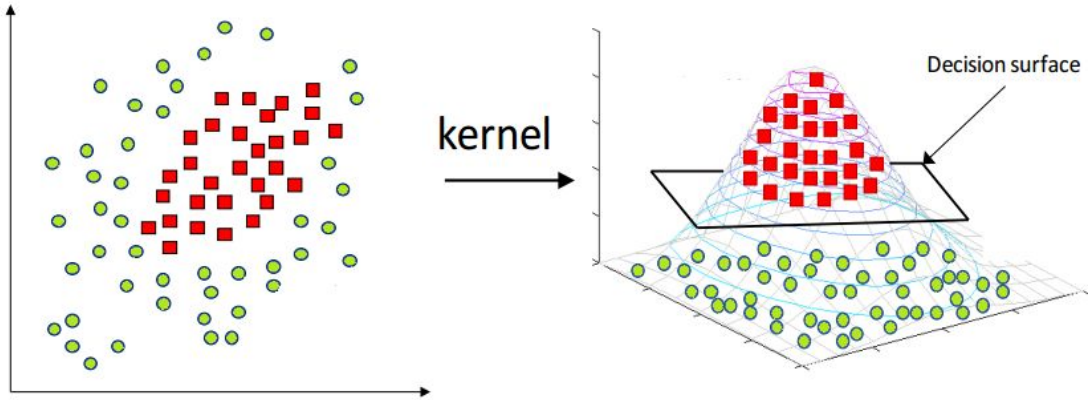


Answers to Question

Support Vector Machine (SVM)

- Max-margin linear model + kernel trick
  - =~ As powerful as non-linear models
- Convex loss function → global optimum (cf. Multi-layer NN)
  - =~ As simple as linear models



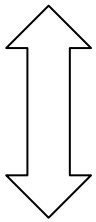


Understanding Kernel Trick with SVM in 2 minutes

Primal form

$$\arg\min_{w,b} \frac{1}{2}\| w \|^2 + C\sum_{n=1}^{N}\xi_n$$

$$s.t\ y_n\left(w^T x_n + b \right) \geqslant 1 - \xi_n\ (n = 1, \ldots, N)$$
$$\xi_n \geqslant 0$$



Dual form

$$L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \overrightarrow{x_{SV}i} \cdot \overrightarrow{x_{SV}j}$$

Understanding Kernel Trick with SVM in 2 minutes

Dual form

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \overrightarrow{x_{SV}i} \cdot \overrightarrow{x_{SV}j}$$

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^p; \text{ polynomial kernel.}$$

$$K(x_i, x_j) = e^{\frac{-1}{2\sigma^2}(x_i - x_j)^2}; \text{ Gaussian kernel; Special case of Radial Basis Function.}$$

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}; \text{ RBF Kernel}$$

$$K(x_i, x_j) = \tanh(\eta\, x_i \cdot x_j + \nu); \text{ Sigmoid Kernel; Activation function for NN.}$$

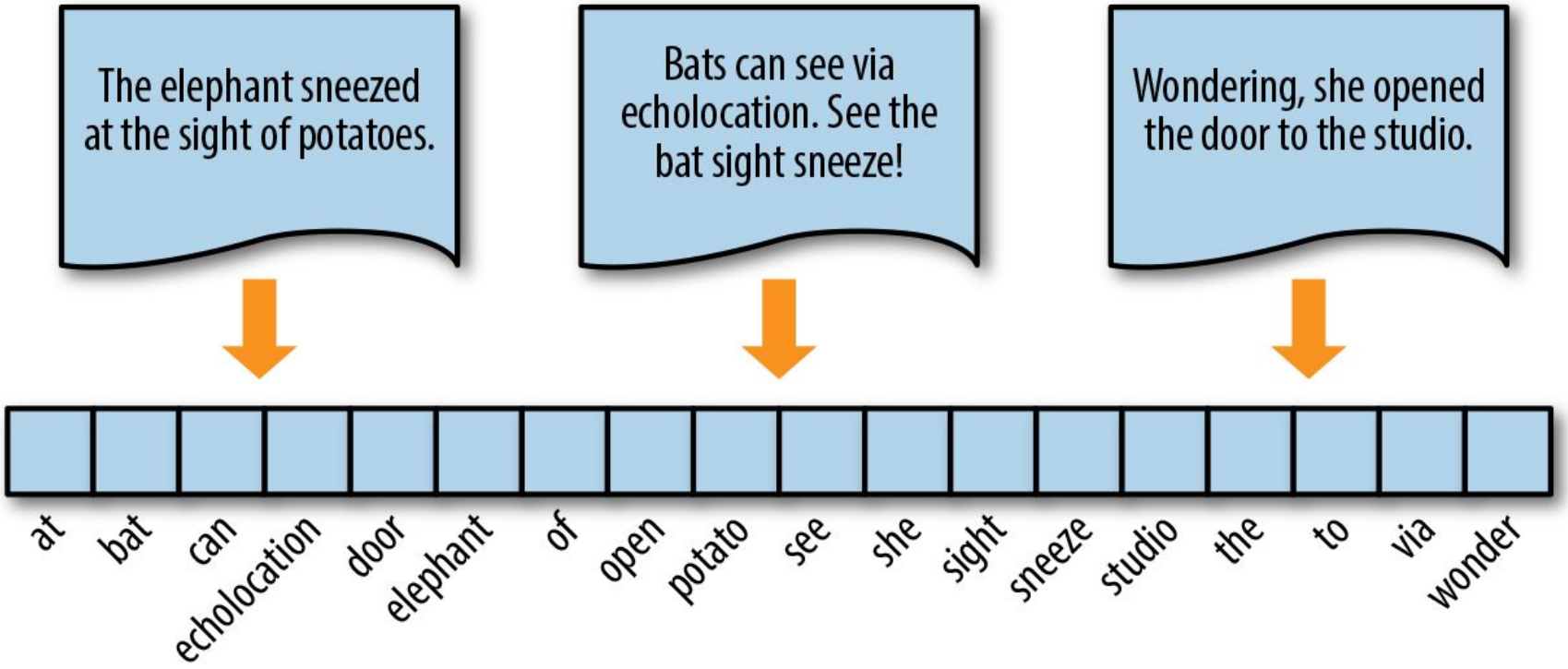
$$\arg\min_{w,b} \frac{1}{2} \| w \|^2 + C \sum_{n=1}^{N} \xi_n$$

$$s.t\, y_n(w^T x_n + b) \geqslant 1 - \xi_n \, (n = 1, \ldots, N)$$

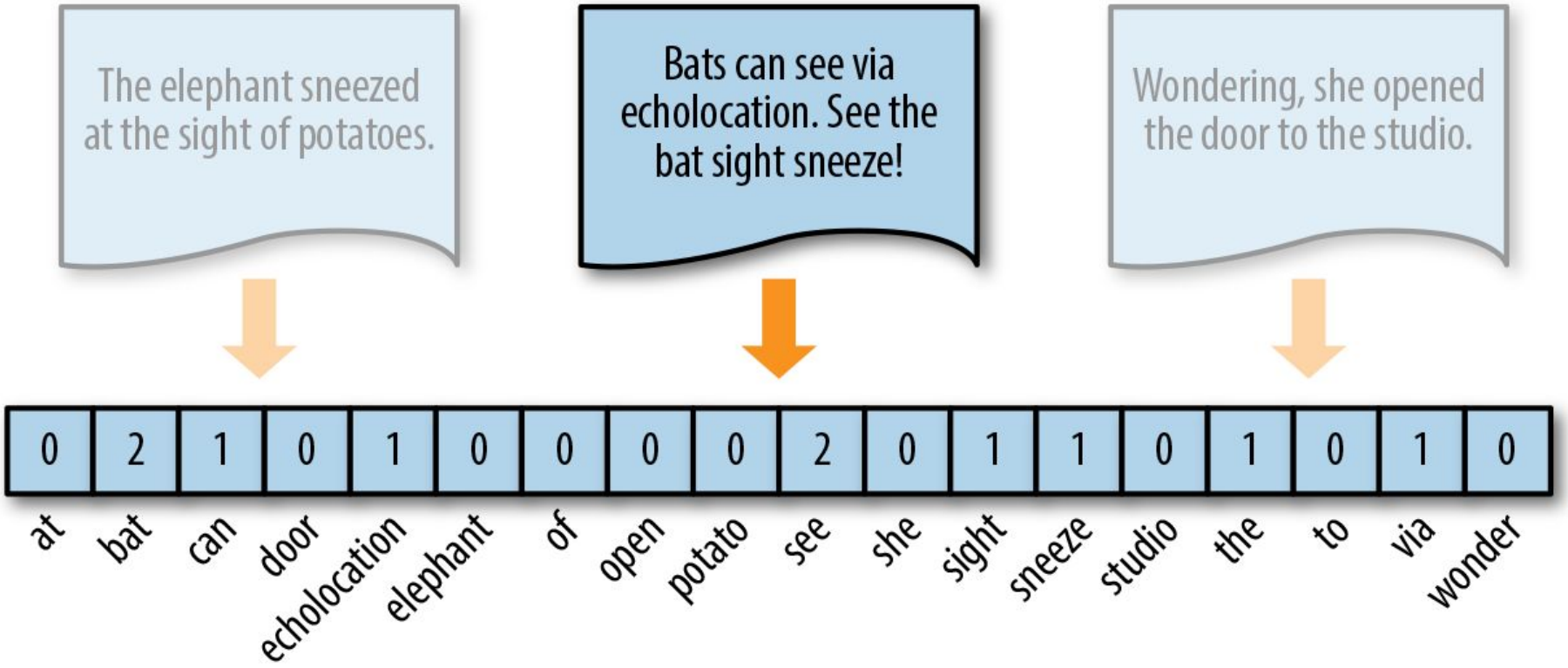
$$\xi_n \geqslant 0$$

How CountVectorizer Works

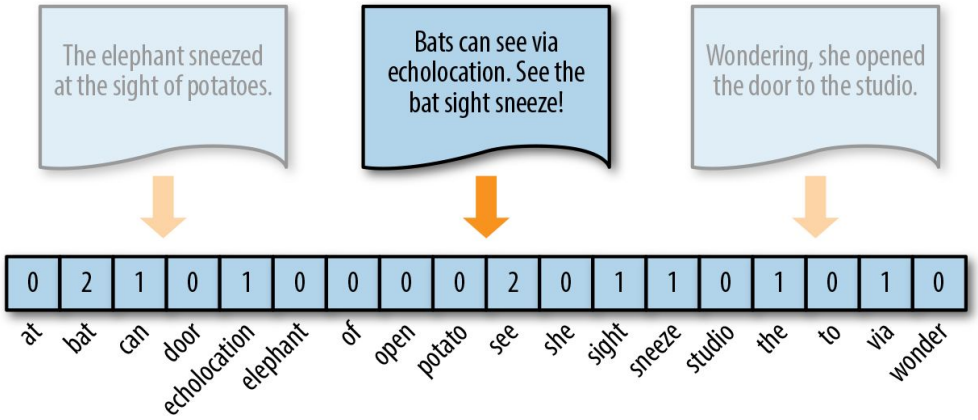
How CountVectorizer Works (1/2)



How CountVectorizer Works (2/2)



Issue: Not All Words are Important



- Solution 1: Remove Stop Words
- Solution 2: Use "importance" weights (e.g., TF-IDF)

TF-IDF

- For term *i* in document *j*

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

- $tf_{i,j}$: # of occurence of term i in document j
- df_i: # of documents that contain term i
- *N*: # of total documents

Quiz

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

- $w_{car,\ Doc1}$?
- $w_{auto,\ Doc2}$?

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

Quiz

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

- $w_{car,\ Doc1} = 0$ ($df_{car} = 3$)
- $w_{auto,\ Doc2} = 33 * \log(3/2) = 13.38$

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

IDF variants

Name	Value
Unary	1
Inverse Document Frequency	$\log \frac{\mathrm{N}}{n_t}$
Inverse Document Frequency Smooth	$\log(1 + \frac{\mathrm{N}}{n_t})$
Inverse Document Frequency Max	$\log(1 + \frac{\max\{t' \in d\} n_t}{n_t})$
Probabilistic Inverse Document Frequency	$\log \frac{\mathrm{N} - n_t}{n_t}$

Figure 2: Different versions of Inverse Document Frequency calculation

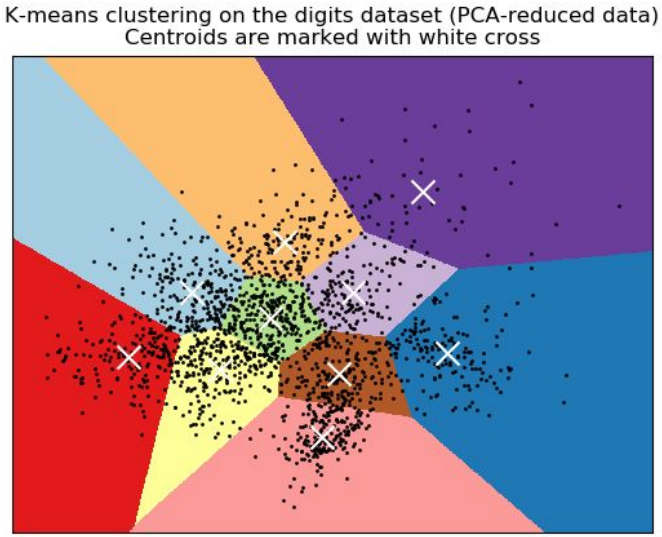
sklearn.feature_extraction.text.TfidfVectorizer

```
>>> # vectorizer = CountVectorizer()
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(filtered_hm_df["cleaned_hm"])
```

Unsupervised Learning

(2) Unsupervised Learning

- Clustering or Representation Learning for Visualization or better supervised learning models



Clustering + PCA

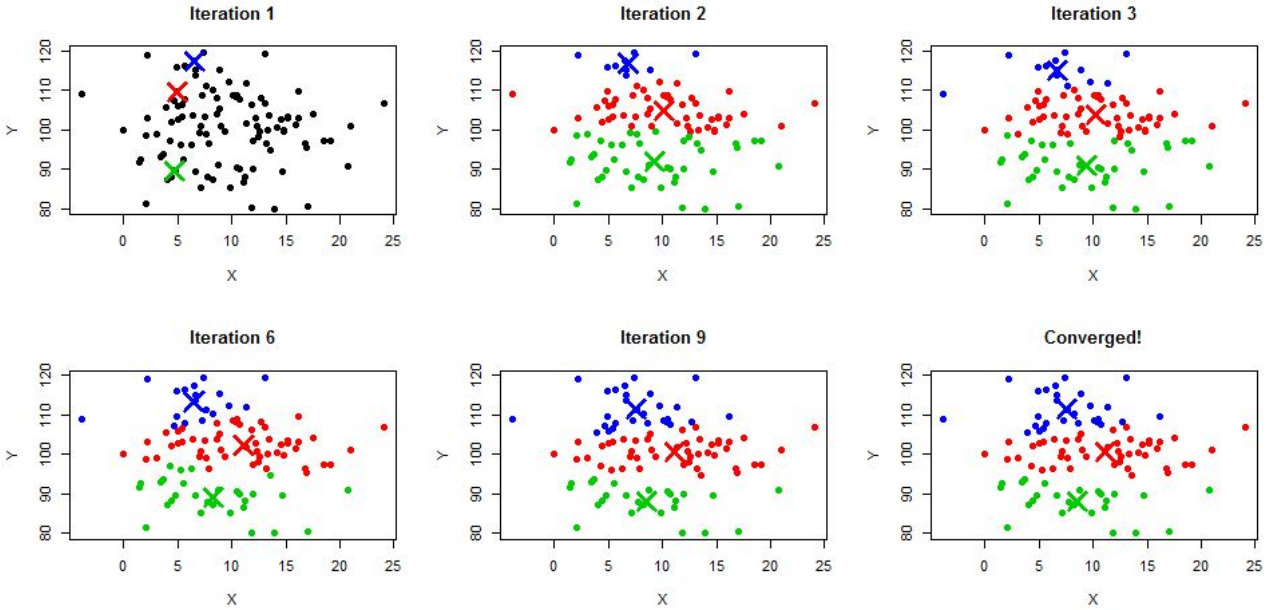
"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Topic Models

K-means

- Step 1. Update k centroids (e.g., k=3)
- Step 2. Assign data points to the nearest centroid



K-means with scikit-learn

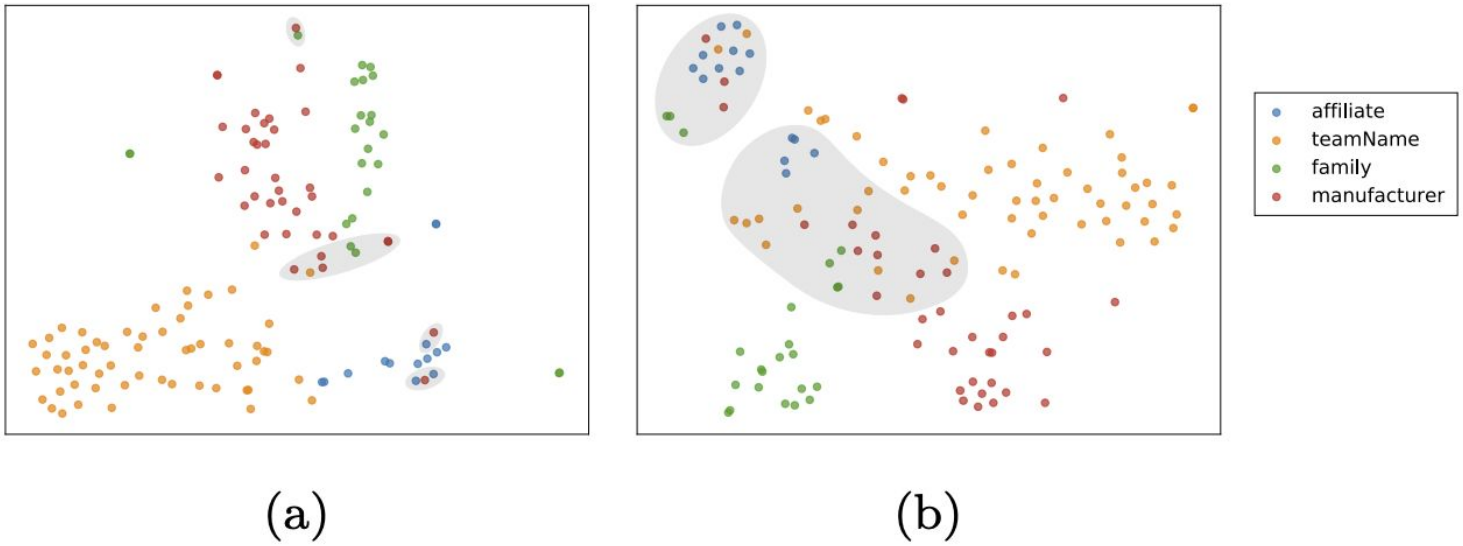
```
>>> from sklearn.cluster import KMeans

>>> clf = KMeans(n_clusters=3)
>>> clf.fit(X)
>>> clf.labels_
array([1, 1, 1, 0, 0, 0, ...], dtype=int32)
```

Dimensionality Reduction for Visualization

- We cannot directly visualize data points in dimensions higher than 3
- One solution is to convert high dimension into 2 or 3 dimensions for visualization

t-SNE for Qualitative Analysis

- Mapping high-dimensional features into 2d space for comparison



Dimensionality Reduction Algorithms

- SVD
- t-SNE

```
>>> from sklearn.manifold import TSNE
>>> from sklearn.decomposition import TruncatedSVD

>>> X2d_svd = TruncatedSVD(n_components=2).fit_transform(X)
>>> X2d_tsne = TSNE(n_components=2).fit_transform(X)
```

Topic Models

Topic Models (=~ Soft Clustering)

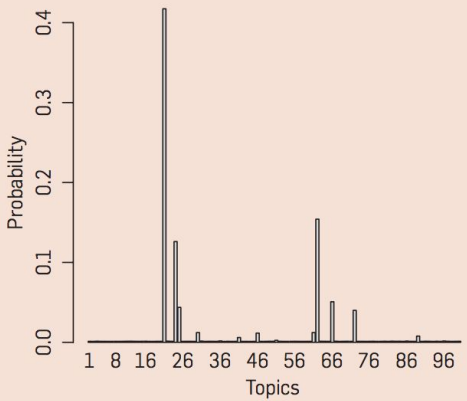
- Probabilistic graphical models that describes **generative process** of data
  - Latent variables = Topics

What are topics?

- Topics are latent groups that are not directly observable in data
  - = Clusters

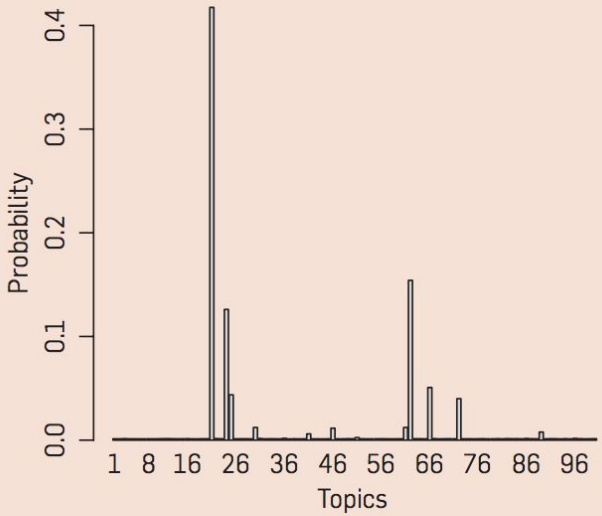
Figure 2. Real inference with LDA. We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left are the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.

Manually assigned



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Figure 2. Real inference with LDA. We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left are the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.

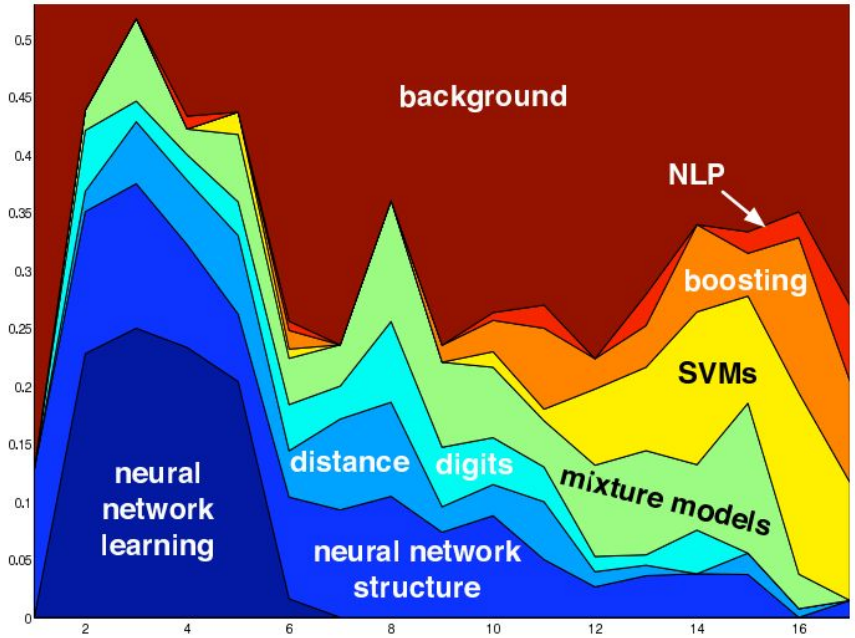


Topic distribution of a document

Topics

"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

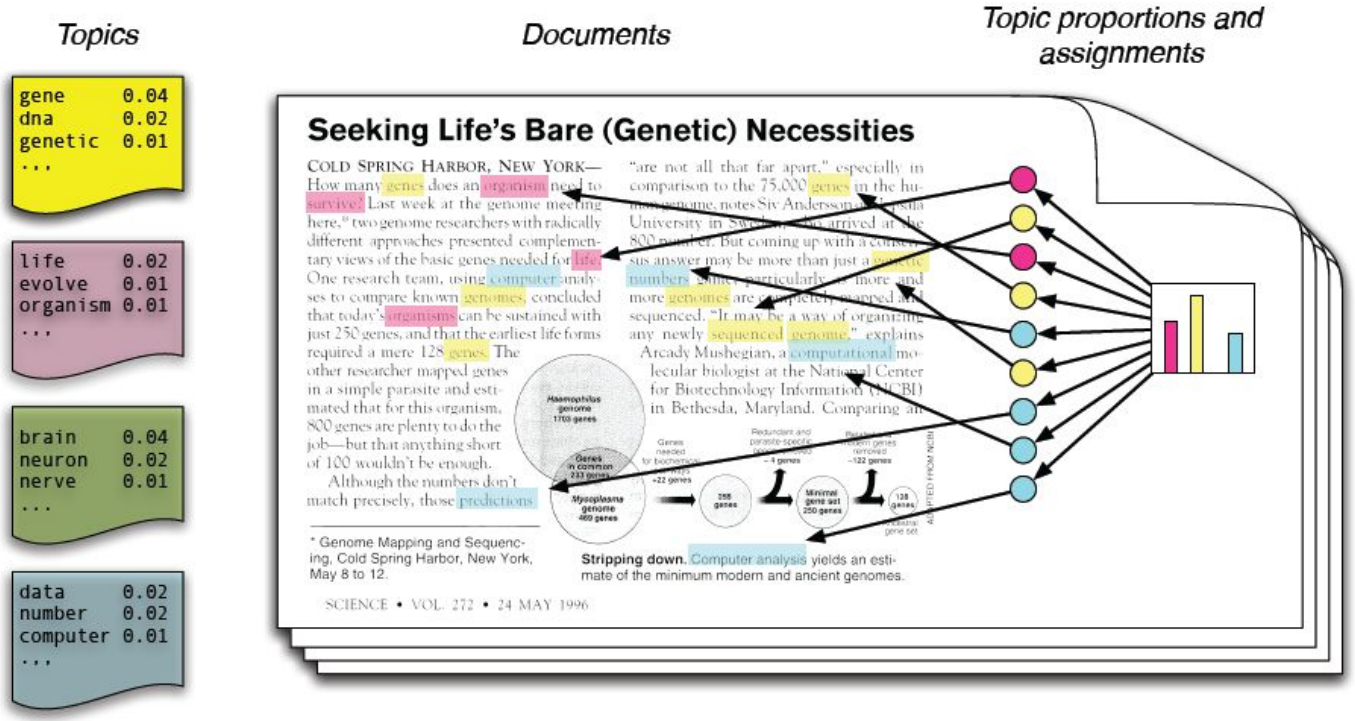
Topic transition over time: NurIPS paper trend



1987 2003

[Wang and McCallum KDD '06]

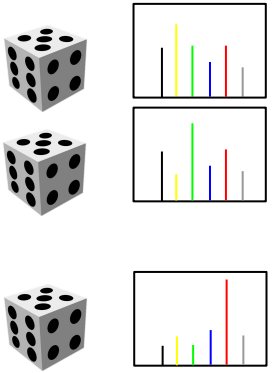
Latent Dirichlet Allocation (LDA)

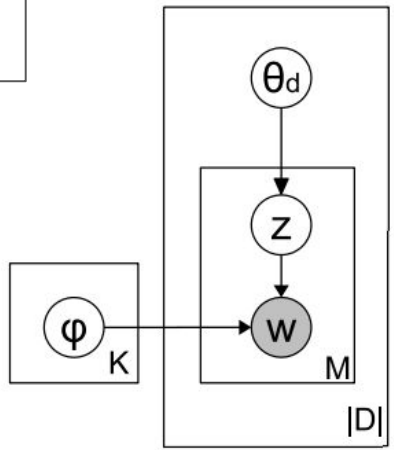


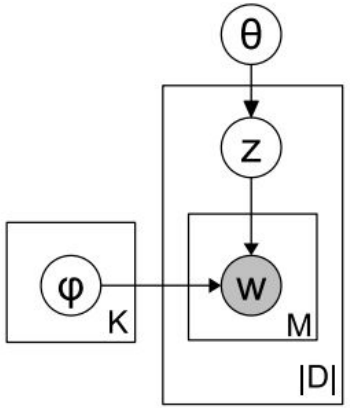
Latent Dirichlet Allocation (LDA): The most common topic model

V-sided dice
(V=vocabulary size)

k-sided dice (e.g., k=6)

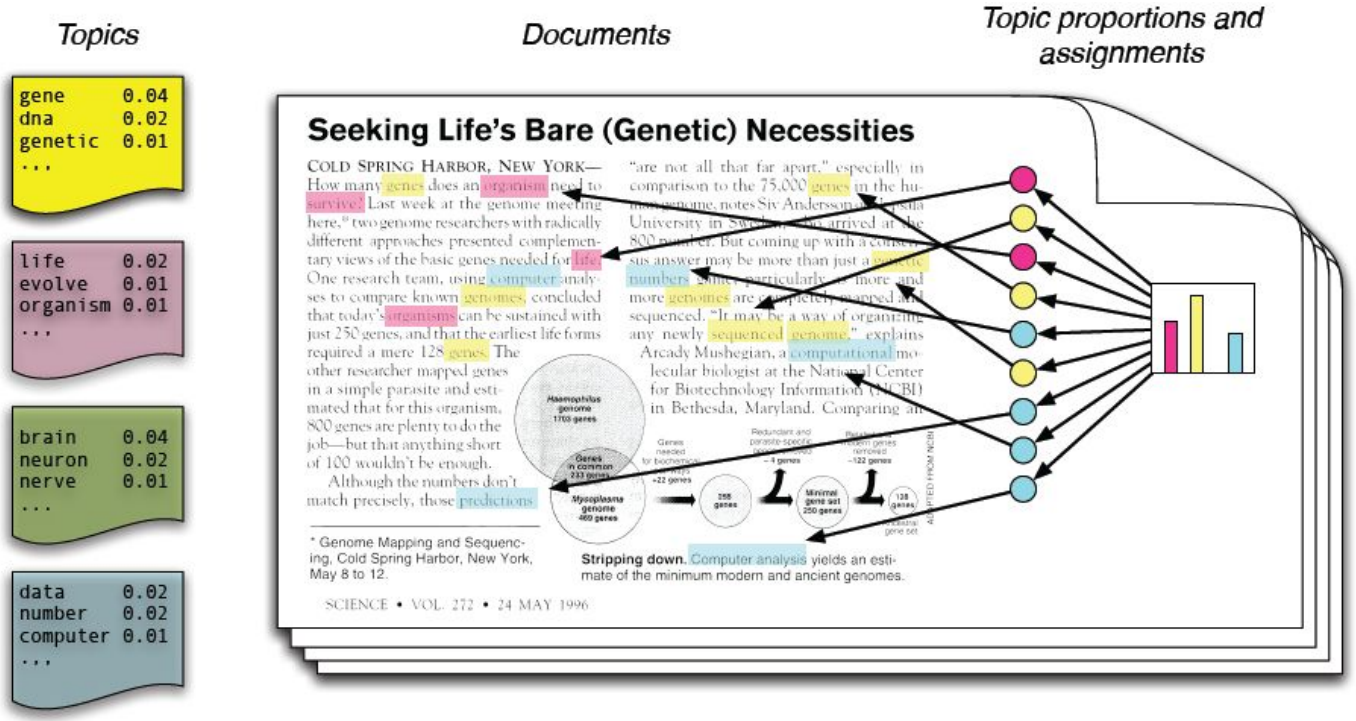




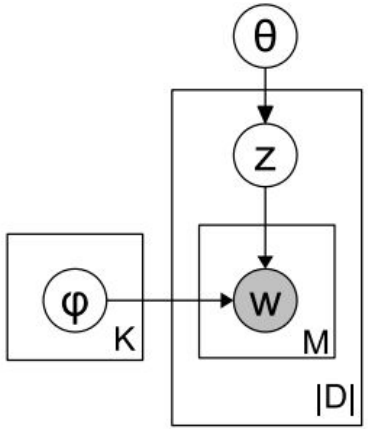


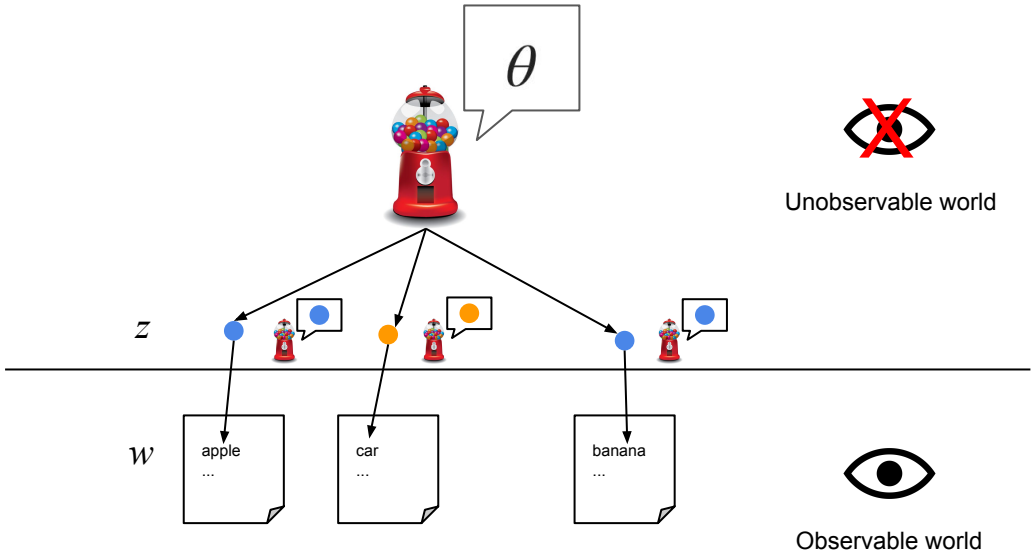
cf. unigram mixture

Latent Dirichlet Allocation (LDA)



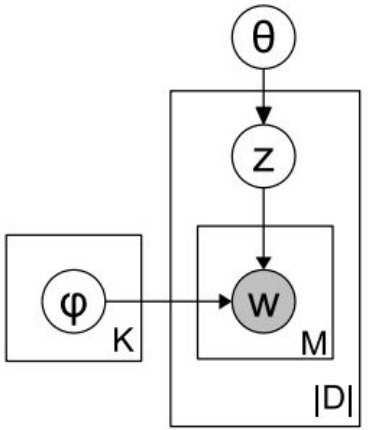
Unigram Mixture: The simplest topic model





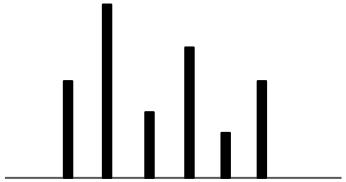
Unigram Mixture: The simplest topic model

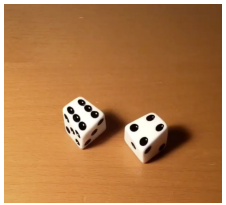
1. For each topic k
   a. Draw a topic-specific word distribution $\phi_k \sim \mathrm{Dir}(\beta)$
2. Draw a topic distribution $\theta \sim \mathrm{Dir}(\alpha)$ for the whole collection
3. For each term w
   a. Draw a topic assignment $z \sim \mathrm{Multi}(\theta)$
   b. Draw a word $w \sim \mathrm{Multi}(\phi_z)$



Note: Multinomial distribution

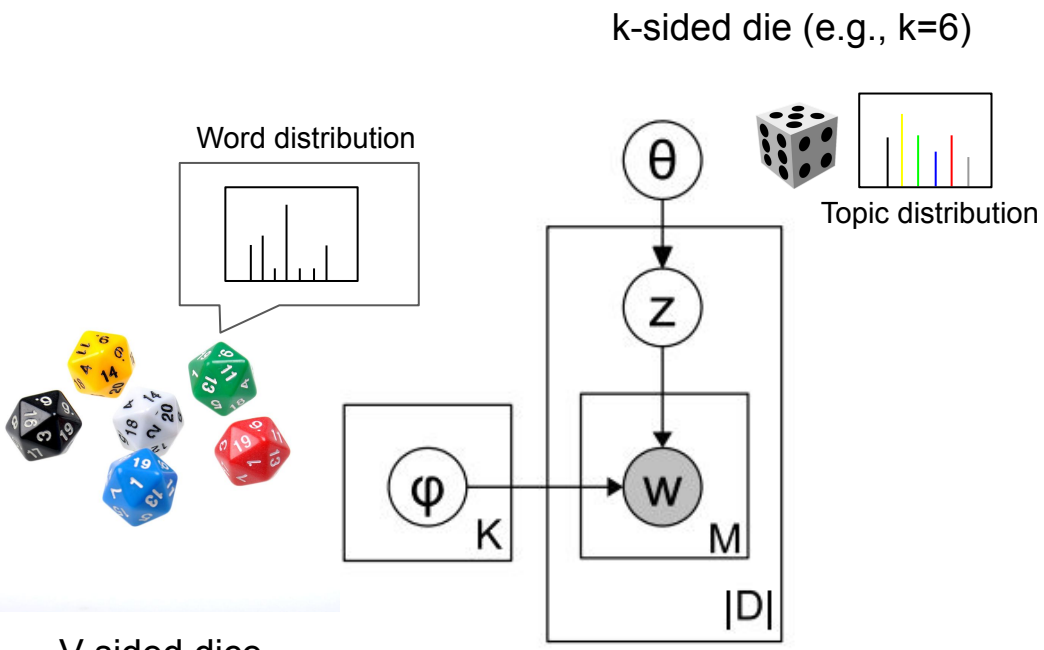
$$1 \geq \theta_i \geq 0, \sum_i \theta_i = 1$$



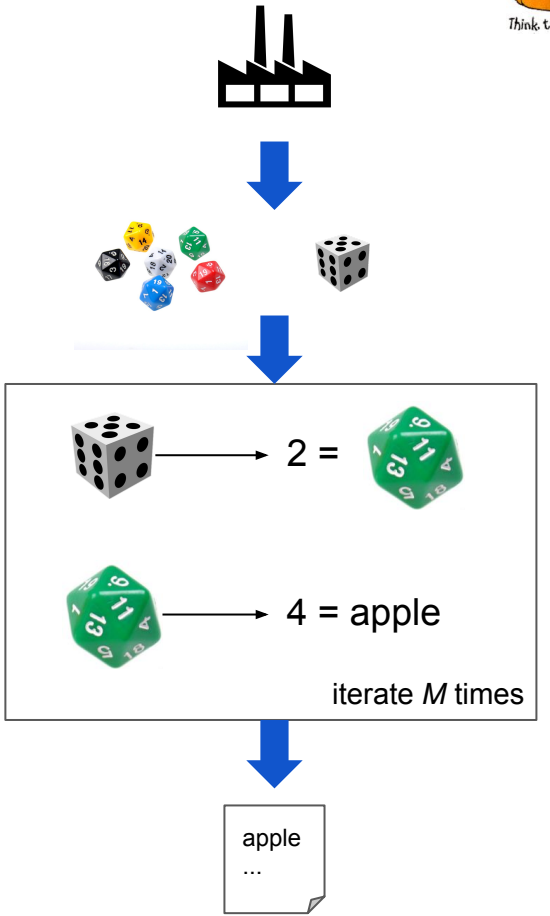


Technically, Categorical (*Multinoulli*) distribution since trial n=1 for each sampling

God does ~~not~~ play dice!



V-sided dice
(V=vocabulary size)



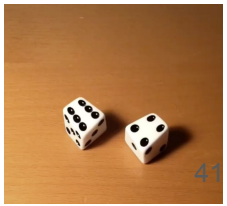
What do we need to estimate?

Model fitting for topic models is to **infer the probabilistic distributions of latent variables** that maximizes the likelihood of a model

In the unigram mixture case:

- Topic distribution: θ
- Word distributions: ϕ_k

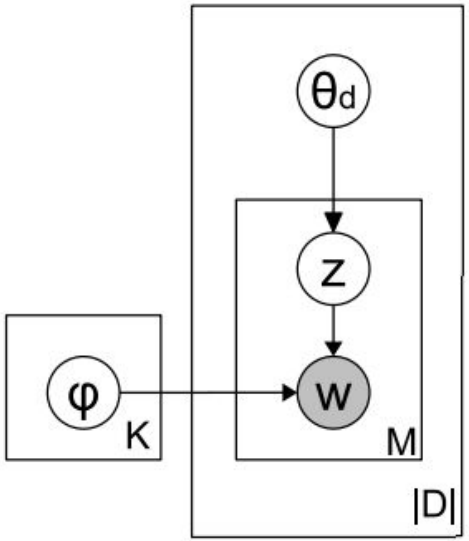
In other words, parameter estimation is to **design an asymmetric die that best explains observable data**



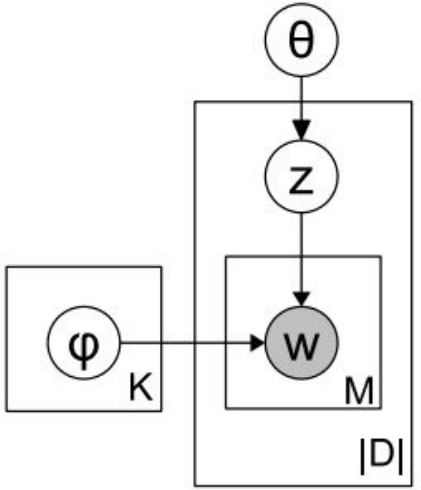
Note: Inference methods

- 1. EM algorithm
  - Pro: Faster inference
  - Cons: Only applicable to simple models (e.g., unigram mixture)
- 2. Markov Chain Monte Carlo (MCMC) algorithms
  - e.g., Gibbs sampling, Collapsed Gibbs sampling, Metropolis–Hastings sampling etc.
  - Pro:
    - Easy to use
    - Good approximation
  - Cons: Slow (does not scale)
- 3. Variational inference (VI) algorithms
  - e.g., Variational Bayes, Collapsed Variational Bayes etc.
  - Pro: Fast
  - Cons:
    - Need to play with math :(
    - Less good approximation (not really in some cases)

Q. What is the difference b/w LDA and Unigram Mixture?



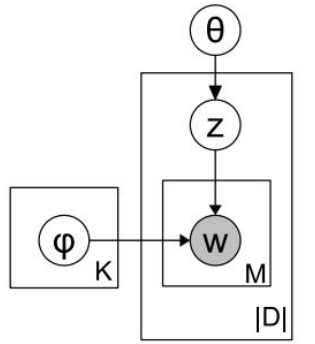
LDA

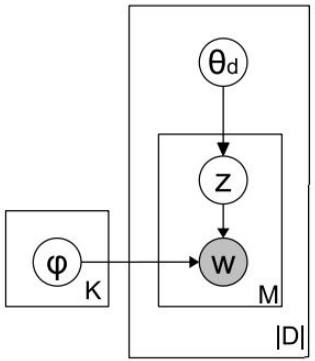


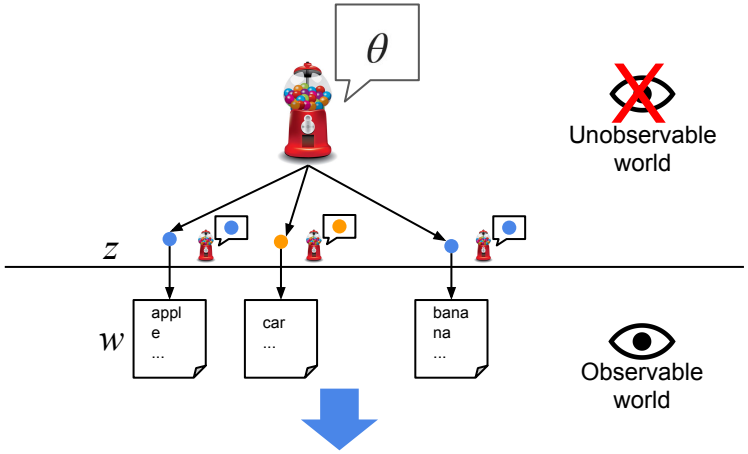
Unigram Mixture

(if (> 30min (curtime))

Q. How does the LDA version of the figure look like?



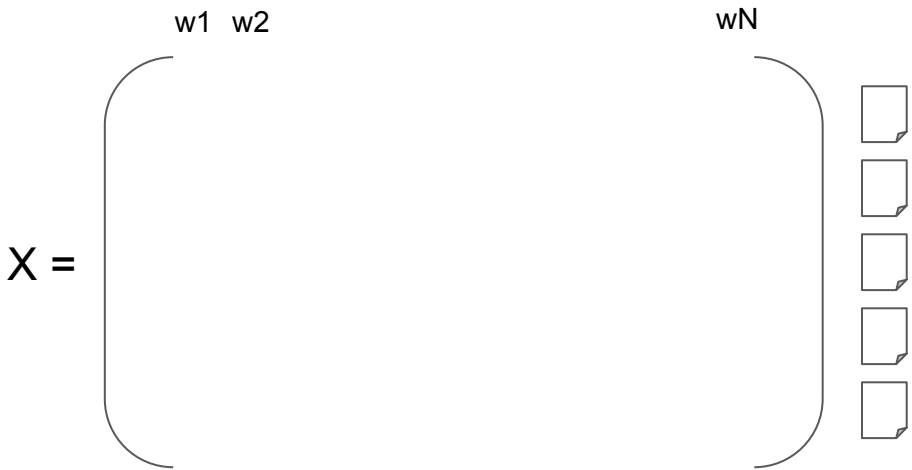




?

sklearn.decomposition.LatentDirichletAllocation

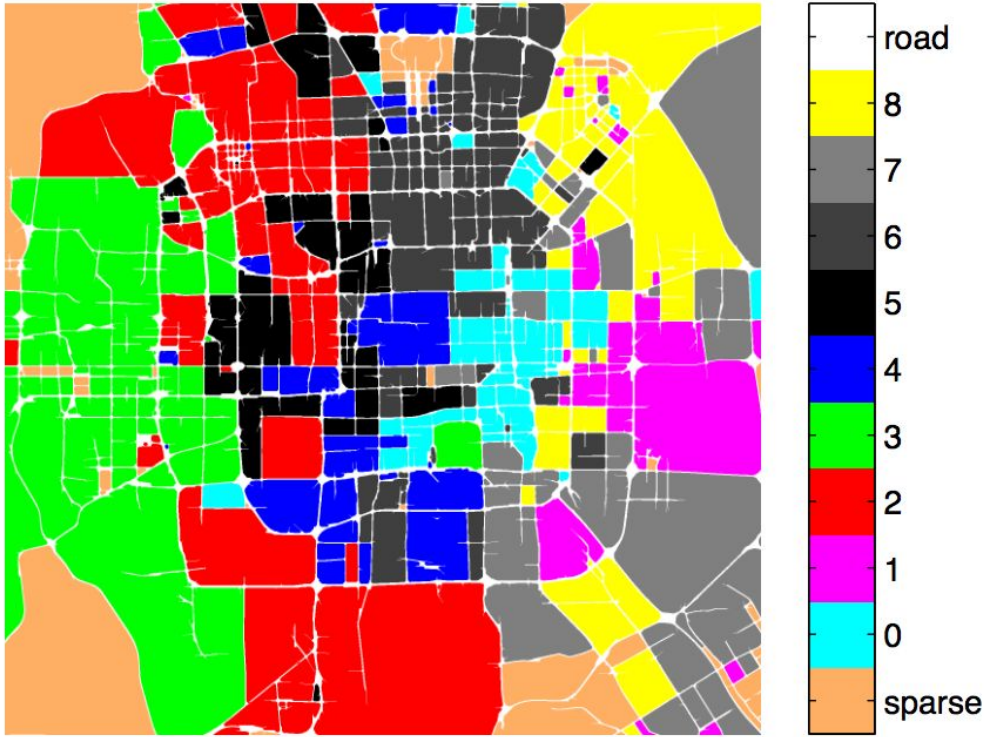
```
>>> lda = LatentDirichletAllocation(n_components=5)
>>> topic_dist = lda.fit_transform(X)
```



Document-term matrix

Topic models for non-NLP tasks

Topic models are not only for Text analysis: Functional regions colored by topics



Document: Region
Word: Point-of-Interest

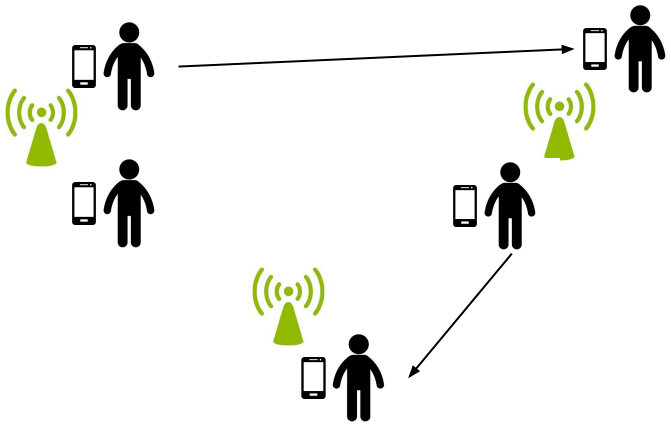
[Yuan+ KDD '12]

What do we need?

- Consider your problem; try to make a documents-and-words analogy
- Create a document-term matrix
- Ready for applying topic modeling libraries

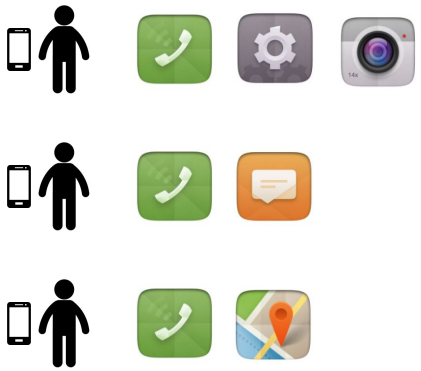
Example 1: Grouping Users Based on Mobility Patterns

- Document: User
- Word: WiFi spot



Example 2: Grouping Users Based on App Usage

- Document: User
- Word: Application



Tips: How do we choose the number of k?

- Trial and error (seriously!)
- Better start from excessive number rather than too few (e.g., k=100)
- Nonparametric Bayes (Hierarchical Dirichlet Process) LDA automatically chooses k, but it does not always work well

Tips: Can we fix α and β ?

- No! We need to tune α and β for better fit
- Especially, α should be optimized
  - e.g., Fixed point iteration [Minka 00]
  - Don't worry! Topic modeling libraries (e.g., Gensim) implement it (scikit-learn does not.)

		Symmetric β	Asymmetric β
Symmetric α		0.080 a **field emission** an **electron** the	0.042 a **field** the **emission** and **carbon** is
		0.080 a the **carbon** and **gas** to an	0.042 the **carbon catalyst** a **nanotubes**
		0.080 the of a to and about at	0.042 a the of **substrate** to **material** on
		0.080 of a **surface** the with in **contact**	0.042 **carbon single wall** the **nanotubes**
		0.080 the a and to is of **liquid**	0.042 the a **probe tip** and of to
Asymmetric α		0.895 the a of to and is in	1.300 the a of to and is in
		0.187 **carbon nanotubes nanotube catalyst**	0.257 and are of for in as such
		0.043 sub is c or and n sup	0.135 a **carbon material** as **structure nanotube**
		0.061 **fullerene compound fullerenes**	0.065 **diameter swnt** about **nm** than **fiber swnts**
		0.044 **material particles coating inorganic**	0.029 **compositions polymers polymer contain**

Asymmetric alpha groups "stop words" into some groups

(Maybe Skip)

Turbo Topic [Blei and Lafferty 09] (1/2)

Huffington Post

movie	barack obama	marriage	hillary clinton
the film	obamas	state	campaign
hollywood	campaign	in california	bill clinton
director	sen barack obama	gay	shes
first	democratic	decision	the clinton
character	the illinois senator	court	hillarys
documentary	michelle	law	president
theater	recent	supreme court	sen clinton
best	speech	couples	mark penn
sex and the city	choice	ruling	politics
hbo	sen clinton	rights	sexism
scene	david axelrod	equality	the first
to make	president	legal	her campaign
release	camp	to marry	supporters
screen	the huffington post	married	made
actor	endorsed	samesex couples	fight
made	seen	states	called
stars	attacks	gay marriage	mrs clinton
indiana jones	political	sexual orientation	political
seen	gave	the california supreme court	hillary rodham clinton

Physics arXiv

mass	model	lattice qcd	phase transitions
star formation	point	mass	model
stars	monte carlo simulations	dirac operator	symmetry
masses	fixed point	chiral perturbation theory	point
black hole	results	operators	quantum
stellar	lattice	quarks	systems
star	scaling	limit	phase transition
black holes	numerical	theta	phase diagram
massive	ising model	quark	system
msun	two	mev	order
solar masses	we study the	simulations	field
stellar mass	models	lattice spacing	order parameter
black hole mass	quantum monte carlo	chiral symmetry breaking	critical
the stellar	interactions	results	two
young	numerical simulations	effects	transitions in
the mass	simulation	small	models
times	dimensions	baryon	different
myr	analytical	in the continuum limit	symmetry breaking
imf	phase	physical	first order
supermassive black holes	spin glass	quenched	phenomena

n-gram topics

film	obama	california	clinton
movie	barack	marriage	hillary
films	obamas	gay	clintons
movies	sen	court	campaign
hollywood	campaign	state	bill
documentary	senator	couples	shes
director	democratic	supreme	president
jones	illinois	decision	hillarys
screen	president	married	supporters
character	presidential	samesex	penn
cannes	recent	rights	politics
festival	political	marry	sexism
city	speech	law	political
theater	huffington	ruling	rodham
star	politics	states	democratic
hbo	michelle	equality	first
scene	voters	legal	say
actor	supporters	lesbian	sen
played	candidacy	equal	mrs
indiana	choice	appeals	presidency

mass	carlo	lattice	phase
black	monte	qcd	transitions
star	simulations	chiral	phases
stellar	point	theory	transition
stars	model	mass	quantum
masses	results	quark	critical
hole	fixed	finite	symmetry
massive	critical	quenched	field
formation	study	perturbation	point
holes	two	limit	model
msun	lattice	quarks	order
function	dimensions	results	diagram
young	scaling	potential	systems
supermassive	numerical	staggered	two
accretion	simulation	chemical	theory
rate	transition	masses	system
solar	ising	simulations	study
initial	phase	theta	breaking
galactic	twodimensional	continuum	spin
central	temperature	volume	first

unigram topics

Turbo Topic [Blei and Lafferty 09] (2/2)

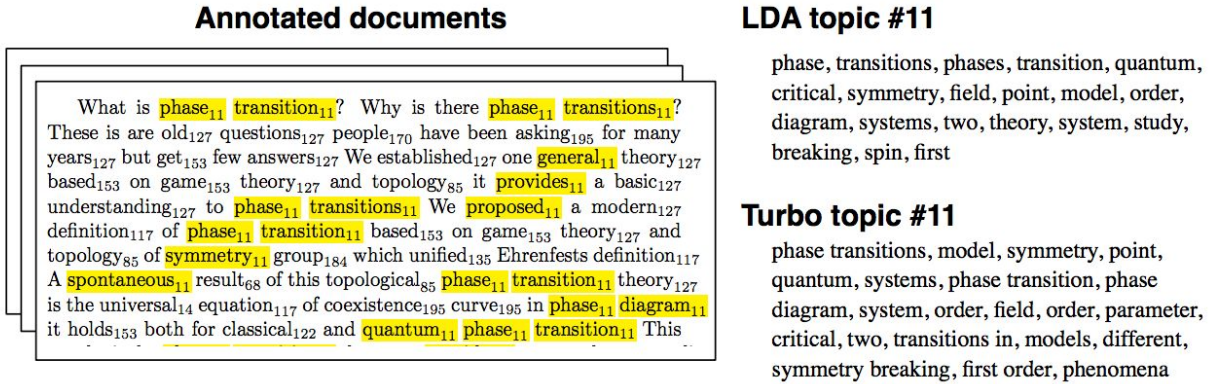


Figure 1: An illustration of the turbo topics strategy. We first estimate an LDA topic model (under the word exchangeability assumption). We next annotate each word in the original corpus with its most likely posterior topic. This is illustrated at left in the subscript on each word and with topic 11 highlighted in yellow. We run a hypothesis testing procedure over the annotated corpus to identify significant words that appear to the left or right of a word or phrase labeled with a given topic. This procedure is carried out recursively, until no more significant phrases are found. At right we illustrate the original top words from topic 11, and those find by the turbo topics strategy. Phrases like "phase diagram," "symmetry breaking," and "first order" are found by the procedure. More topics are illustrated in Figure 3.

A lot more ...

Tools

- Gensim
  - LDA, HP-LDA
- Scikit-learn
  - Only LDA (no alpha, beta optimization :()
- pyLDAvis
  - Visualizes Gensim results

Summary: Topic models as interpretable machine learning tools

- Consider your own problem to derive a "document-and-word" analogy
- Start from applying simple LDA
  - With the large number of k
  - Tuning \alpha asymmetrically
- Take a look at the result and interpret it!
  - e.g., topic distributions, word distributions

Break?

NLP: What and Why

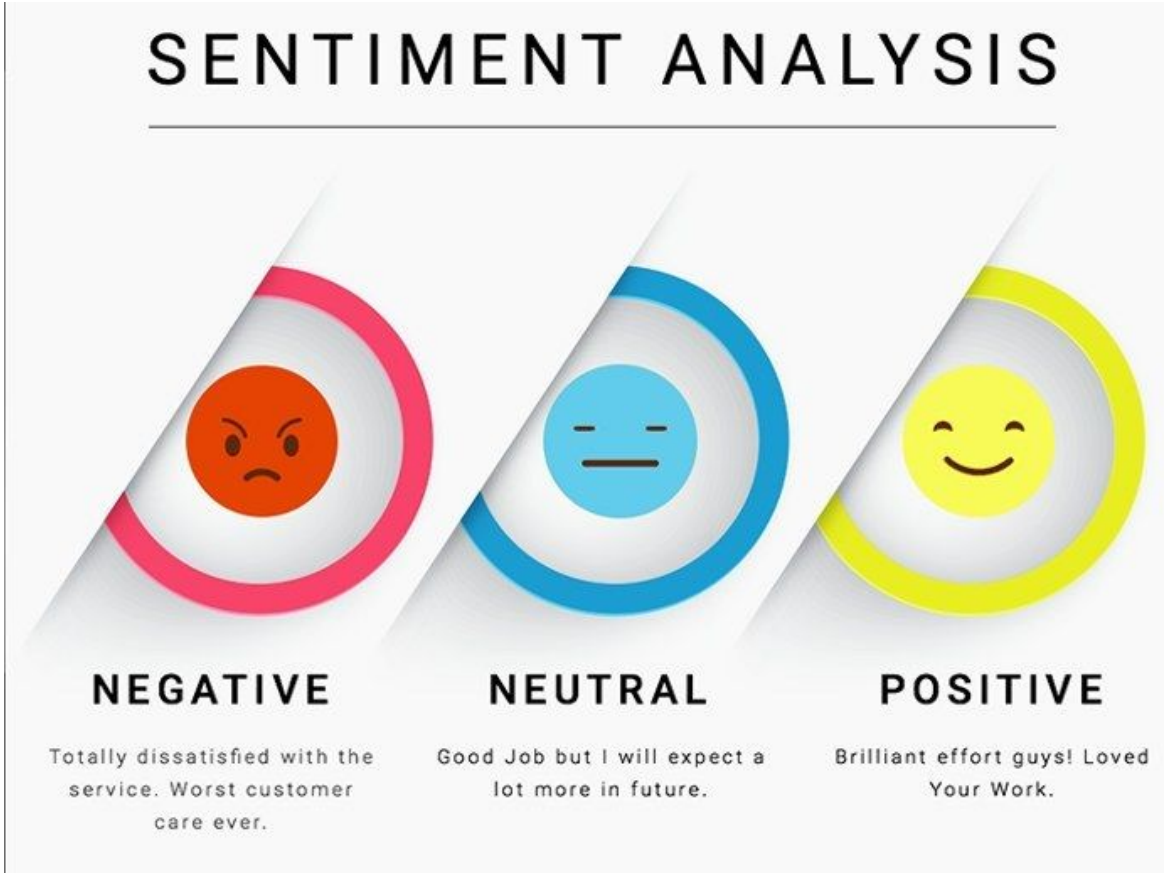
What is Natural Language Processing?

- Natural Language =~ Text (in this lecture)

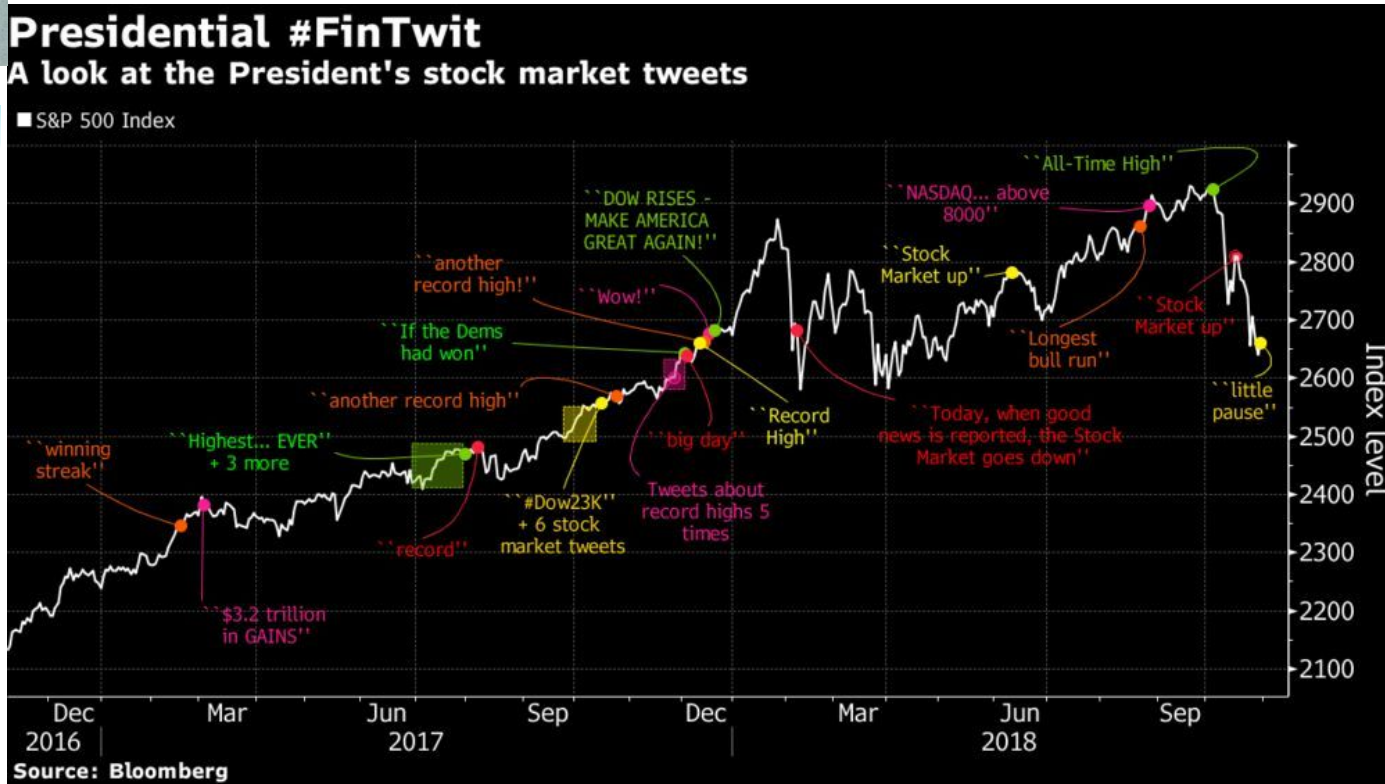
Why NLP?

- A lot of good ML application "templates"
- NLP != ML

Sentiment Analysis / Emotion Analysis



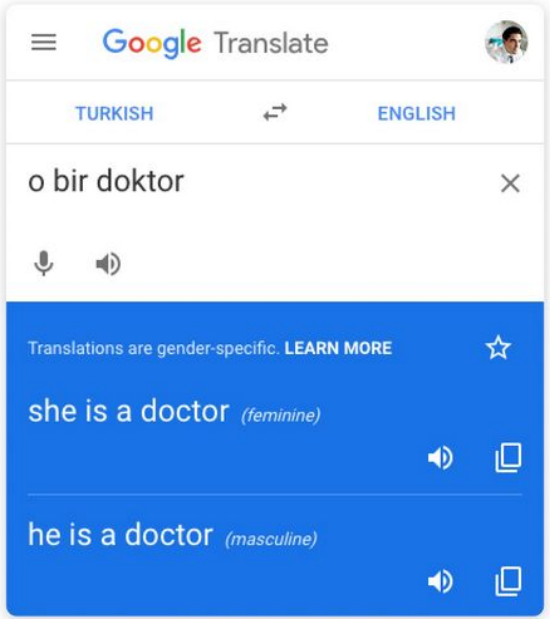
ALGORITHMIC TRADING



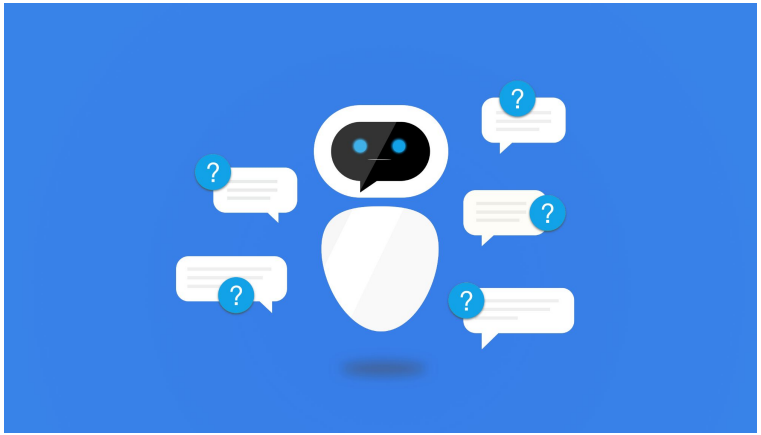
Machine Translation

Before

After



Conversational Interfaces



Question Answering / Reading Comprehension

Stanford Question Answering Dataset (SQuAD)

Passage

> Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

Question: Which NFL team won Super Bowl 50?

Answer: Denver Broncos

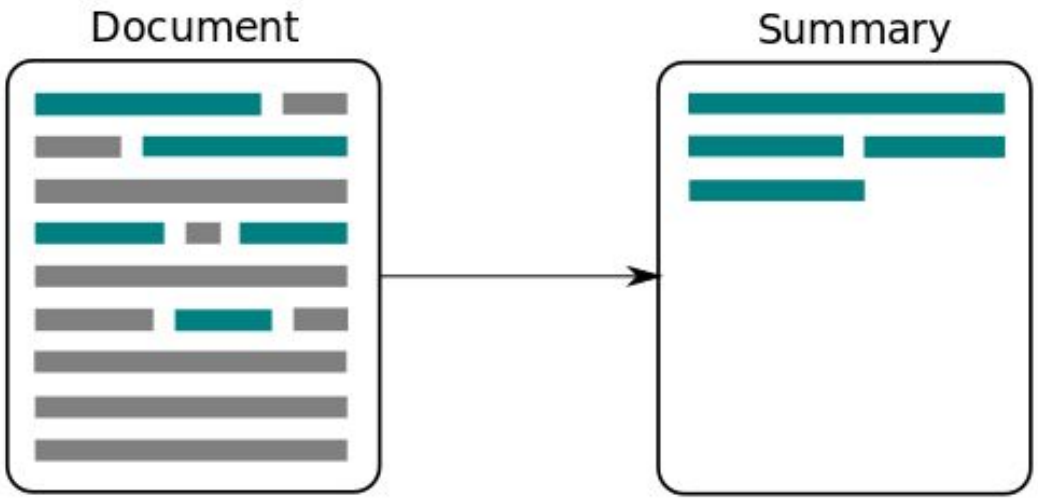
Question: What does AFC stand for?

Answer: American Football Conference

Question: What year was Super Bowl 50?

Answer: 2016

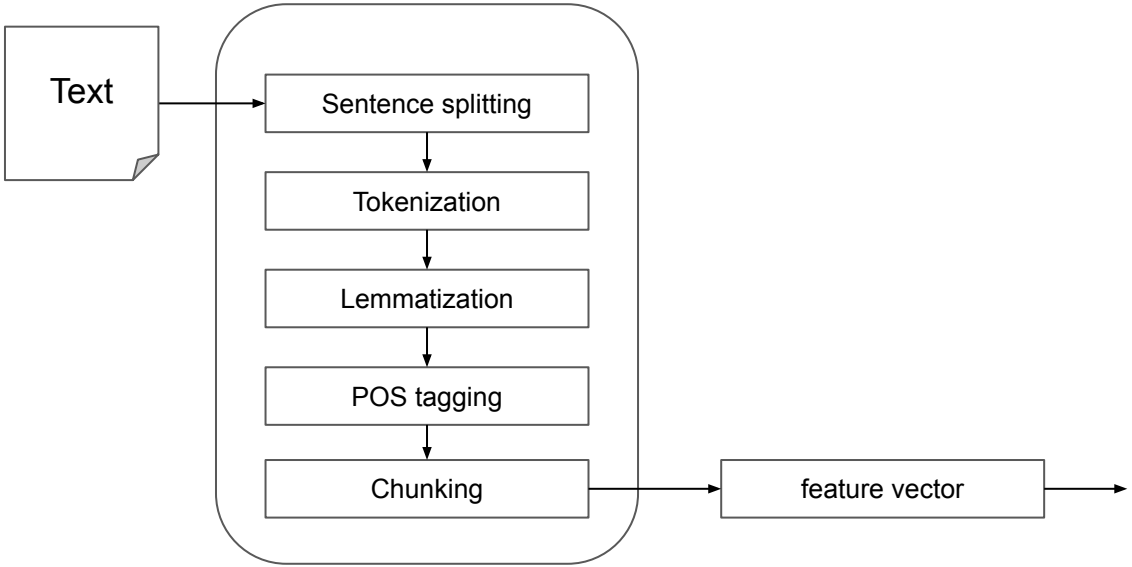
Text Summarization



NLP basics

Conventional NLP Pipeline

- Core NLP tasks (cf. Downstream NLP tasks)



Sentence splitting / Tokenization

Tokenization is NOT trivial for some languages

Japanese example

東京都の人口は約1400万人です。

The population of **Tokyo Metropolis** is about 14 million.

東京　都 (Tokyo Metropolis)
tokyo to

東　京都 (East Kyoto)
higashi kyoto

Links to Japanese Tokenizers

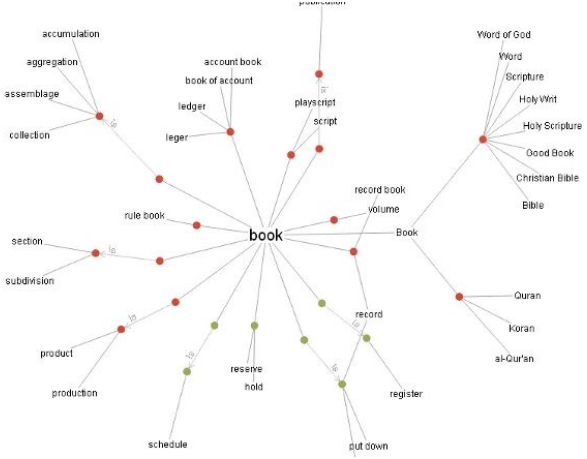
- Japanese Morphological Analyzer
  - MeCab https://pypi.org/project/mecab-python3/
  - SudachiPy https://github.com/WorksApplications/SudachiPy
    - spaCy + MeCab https://www.dampfkraft.com/nlp/japanese-spacy-and-mecab.html

- Universal Dependency Parser for Japanese
  - GiNZA https://github.com/megagonlabs/ginza

Lemmatization/Stemming

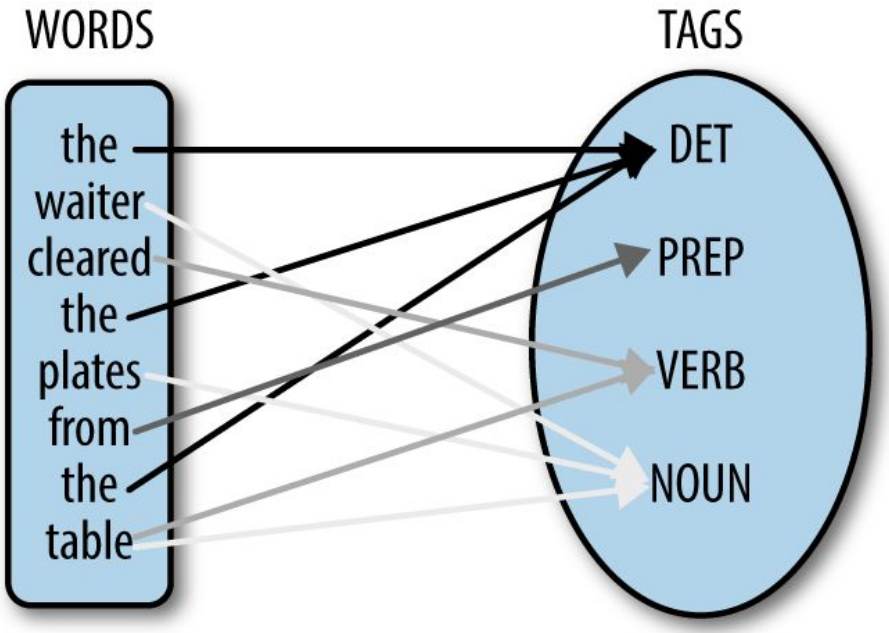
- To reduce inflectional forms and canonicalize into a common form
  - am, are, is → be
  - car, cars, car's, cars' → car

- Rule/dictionary-based
  - Porter stemmer
  - WordNet



Part-of-Speech (POS) tagging

- Assigning word-category tag for each token



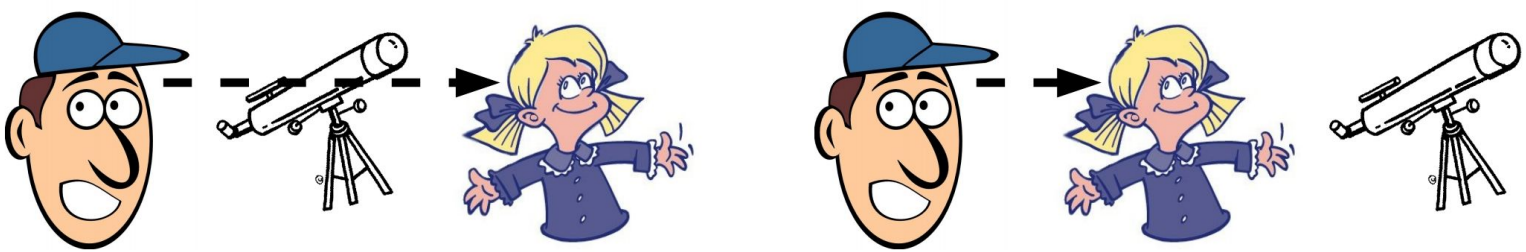
Named Entity Recognition

In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

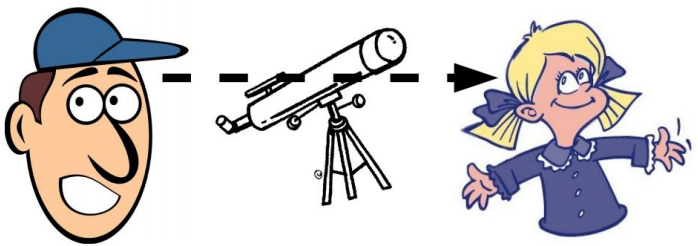
Syntactic Parsing

I saw a girl with a telescope



http://www.phontron.com/slides/nlp-programming-en-11-depend.pdf

Dependencies Resolve Ambiguity



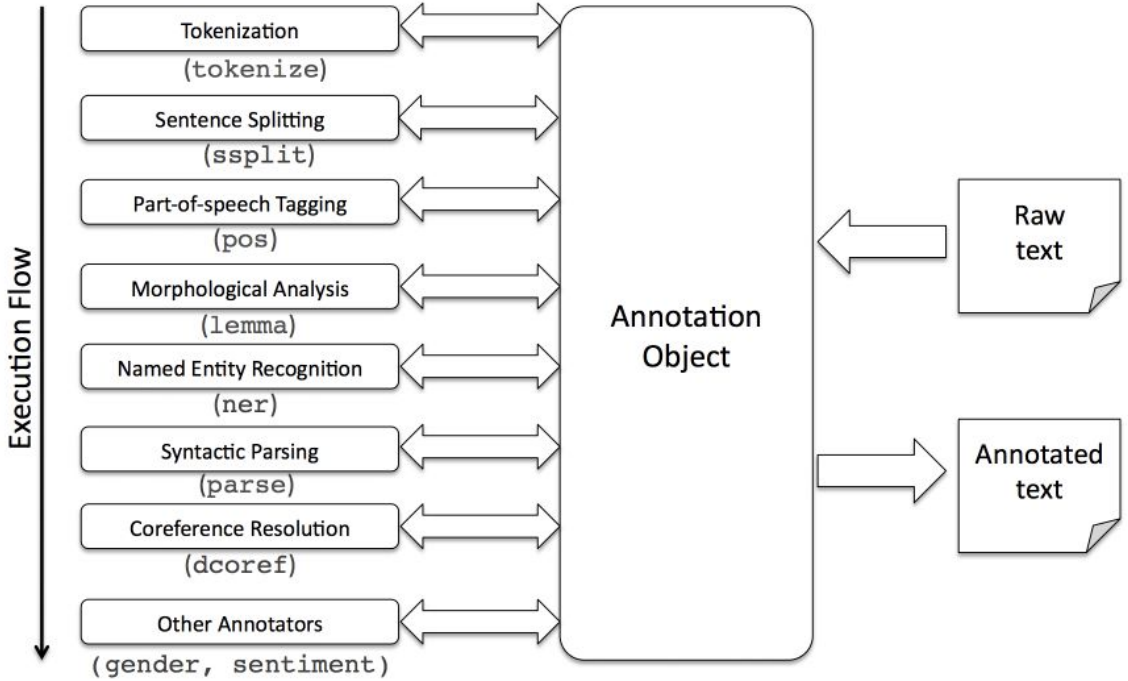
I saw a girl with a telescope

I saw a girl with a telescope

http://www.phontron.com/slides/nlp-programming-en-11-depend.pdf

Why NLP techniques are important?

- It provides multiple ways of annotating linguistic features to input text



A few lines of Python Code: spaCy Example

```
# pip install spacy
# python -m spacy download en_core_web_sm

import spacy

# Load English tokenizer, tagger, parser, NER and word vectors
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. “I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn’t "
        "worth talking to,” said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

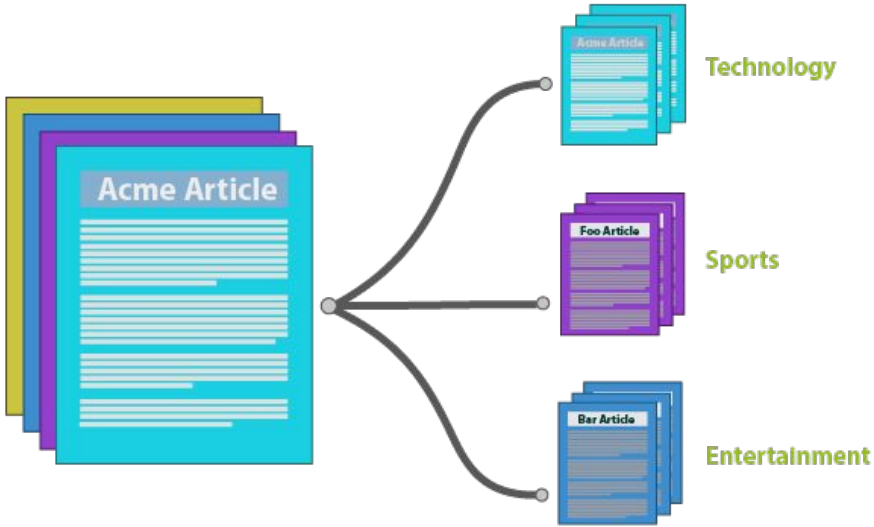
ML Problem Formulations for NLP tasks

ML Problem Formulation for NLP

- Binary/Multi-class classification
  - Text classification, Sentiment classification
- Sequential tagging
  - POS tagging, Named Entity Recognition,
- Sentence-pair classification
  - Textual entailment recognition

Sentiment classification / Text classification

- Input: Text
- Output: pos/neg, categories
- Task: Binary or multi-class classification



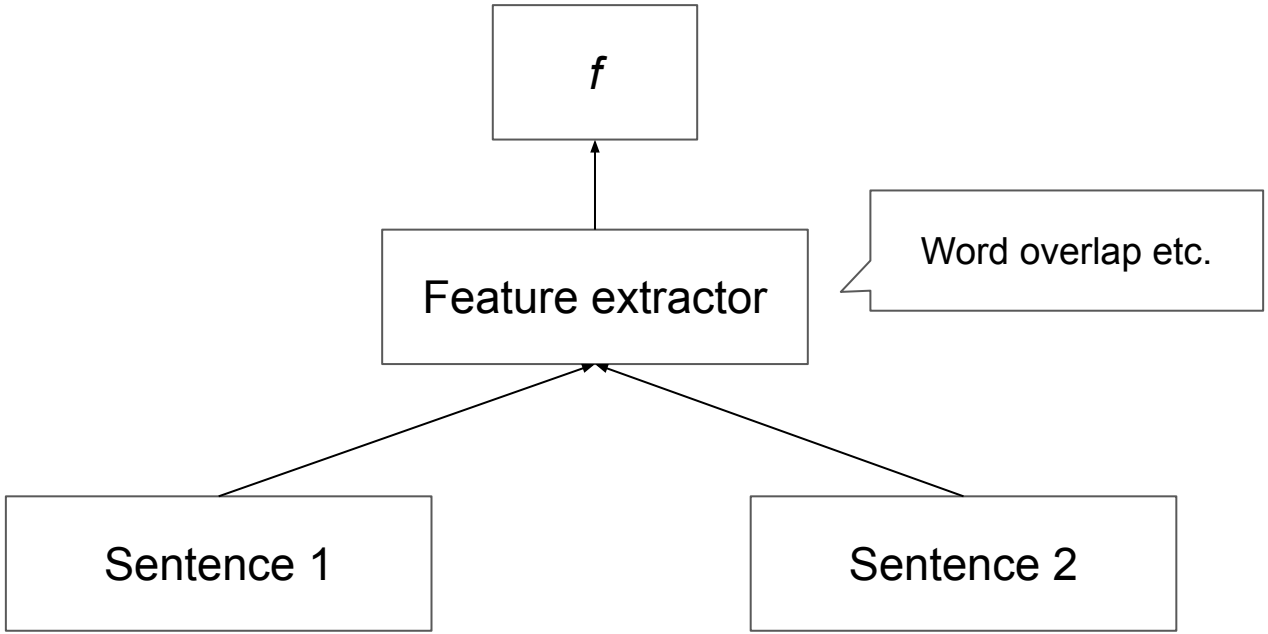
Sentence-pair Classification

- Textual Entailment Recognition
  - Given two texts (e.g., sentences), the task is to judge whether the meaning of one text is entailed (can be inferred) from another text

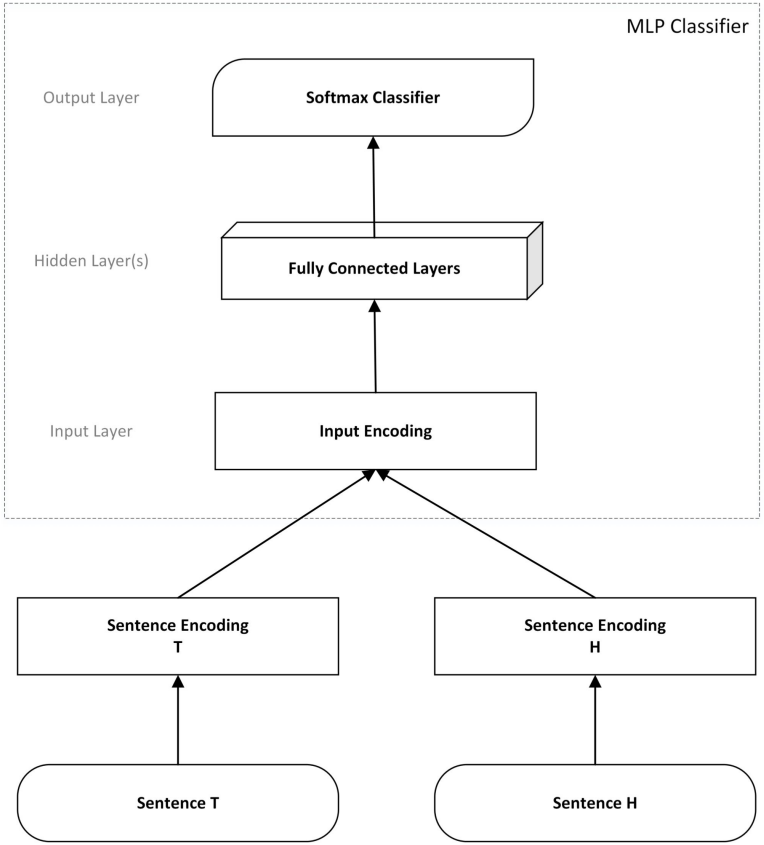
T: The carmine cat devours the mouse in the garden.
H: The red cat killed the mouse.

Label: Entailed (H is entailed by T)

Sentence-pair Classification: Naive Solution



Sentence-pair Classification: Recent Approach



Sequential Tagging

- Input: Sequence of tokens
- Output: Sequence of labels (new!)
- Task: **Sequential Tagging Problem** (new!)

B-ORG	O	O	O	O	B-GPE	O	O	B-MONEY	I-MONEY
Apple	is	looking	to	buy	U.K.	startup	for	$1	billion

BIO tagging schema

- B: Beginning
- I: Inside
- O: Outside

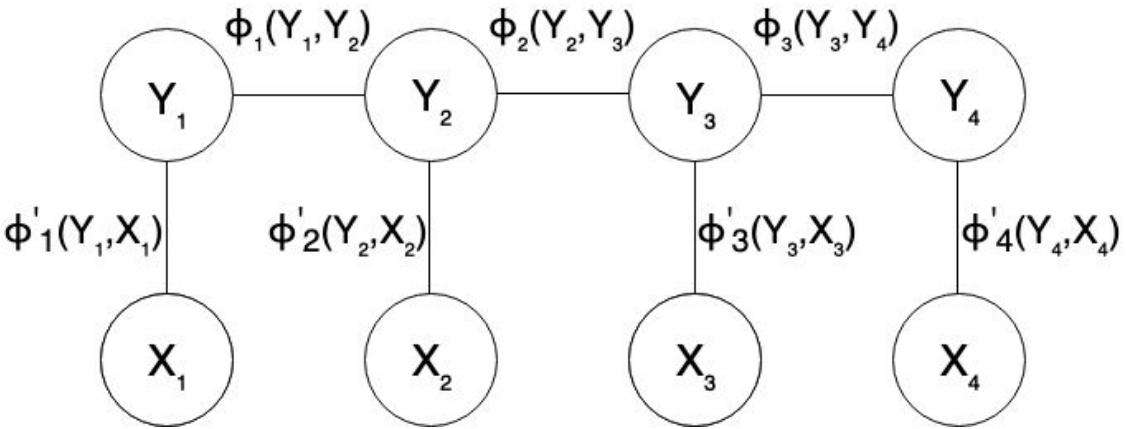
Sequential Tagging: A Naive Solution

- Sequential application of multi-class classifier
  - A) Independently
  - B) Sequentially
    - using prediction in the previous step as a feature

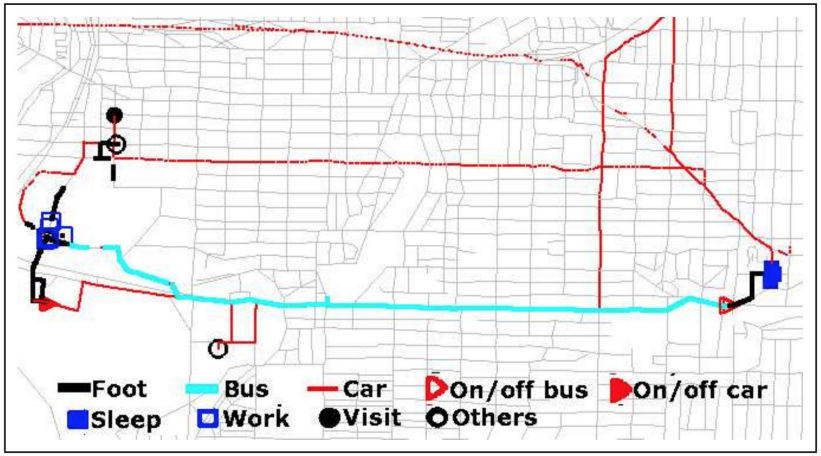
?	?	?	?	?	?	?	?	?	?
Apple	is	looking	to	buy	U.K.	startup	for	$1	billion

Sequential Tagging: A Standard Solution

- Conditional Random Field (CRF)
  - Example) Linear-chain CRF
- Those algorithms are called Structured Output Learning



Sequential Tagging for non-NLP tasks



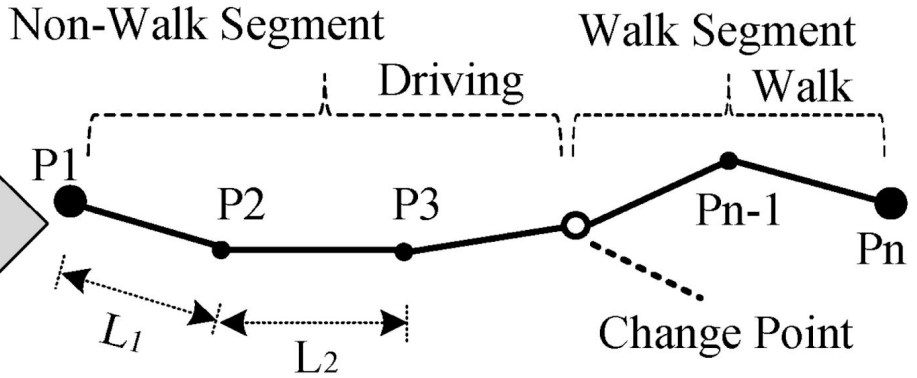
Latitude, longitude, Time

P1: Lat1, Long1, T1

P2: Lat2, Long2, T2

……..

Pn: Latn, Longn, Tn



Language Models

Language Models

- Probabilistic models of a language
  - Originally, for evaluating likelihood of text, text generation

Language Models from 2018

- Language Models came to have a special meaning in NLP from 2018
  - Have you heard about ELMo or BERT?

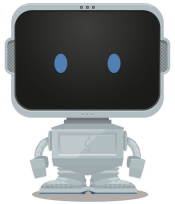
Figure

Will talk about this topic tomorrow

Further Topic: Automated Machine Learning (AutoML)

- Search everything!
- Automatically finds the best model

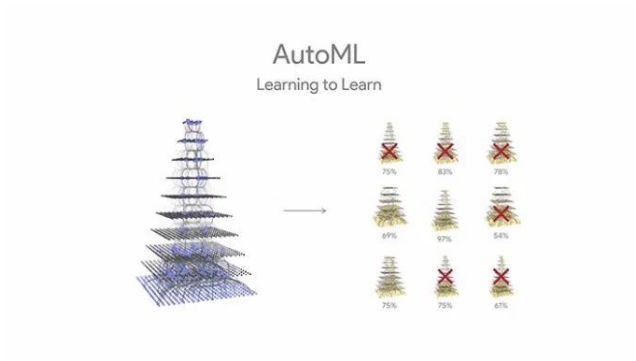
H2O.ai

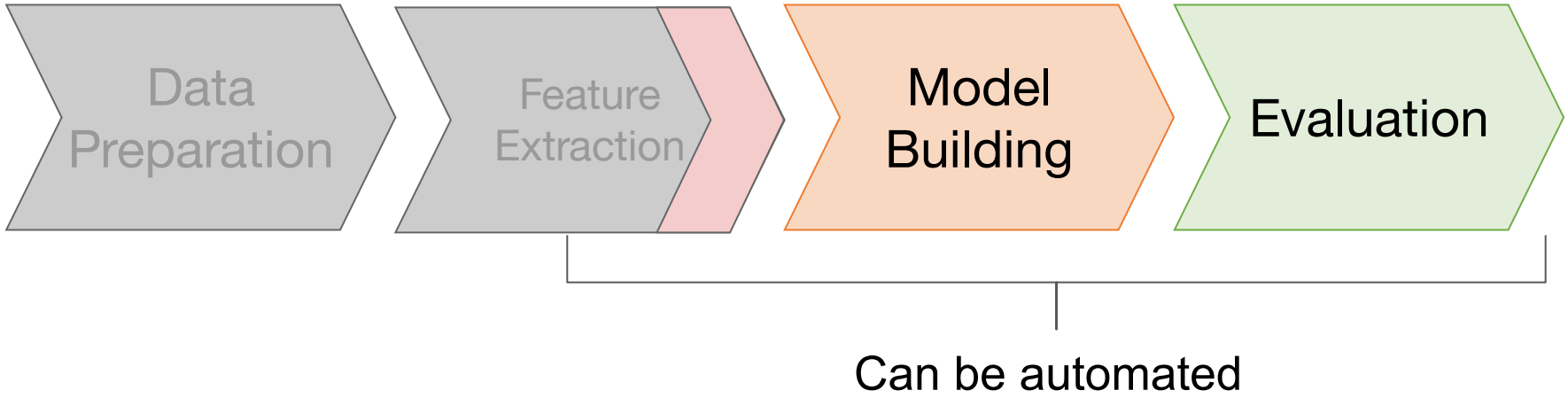


DataRobot

kaggle

Auto-Sklearn




Data Preparation
Feature Extraction
Model Building
Evaluation
Can be automated

Before

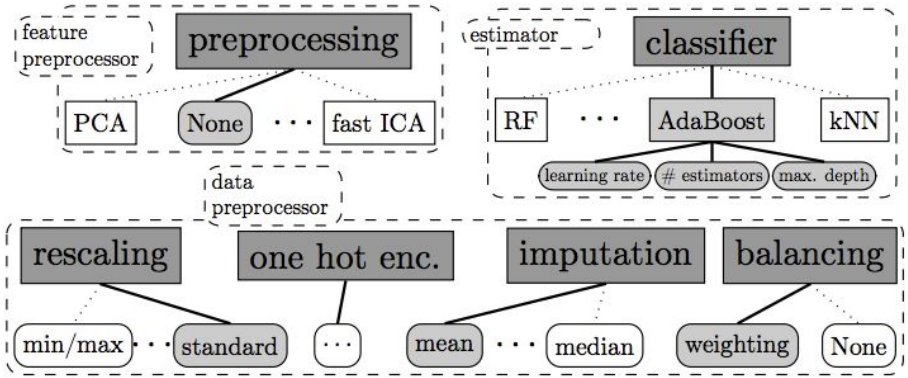
```
import numpy as np
import json

from sklearn.linear_model import Ridge
from sklearn.grid_search import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor, GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesRegressor, ExtraTreesClassifier
from sklearn.datasets import load_svmlight_file
from sklearn.cross_validation import KFold
import sklearn.metrics
import xgboost as xgb

import sys
sys.path.append('lib')
import features

if __name__ == '__main__':

    if len(sys.argv) < 3:
        print("Input XXX XXX [cvnum]")
        sys.exit()

    feature_baseid = sys.argv[1]
    classifier_baseid = sys.argv[2]

    cvnum = 10
    if len(sys.argv) >= 4:
        cvnum = int(sys.argv[3])

    train_filepath = 'model/%s_feature_train.txt' % (feature_baseid)
    # test_filepath = 'model/%s_feature_test.txt' % (feature_baseid)
    output_filepath = 'output/cv_k-%02d_%s-%s.json' % (cvnum, feature_baseid,
classifier_baseid)

    X, y = load_svmlight_file(train_filepath)
    X = X.todense()
    # X_test, y_test = load_svmlight_file(test_filepath)
    # X_test = X_test.todense()

    if classifier_baseid == '100':
        clf_type = 'r'
        parameters = {'learning_rate': [0.1,0.05,0.02,0.01],
                      'n_estimators': [10,100,1000],
                      'max_features': [1.0,0.3,0.1]}
        clf = GridSearchCV(GradientBoostingRegressor(random_state=1),
                           parameters,scoring='roc_auc',
                           n_jobs=-1, cv=3, verbose=2)

    elif classifier_baseid == '101':
        clf_type = 'r'
        blend_param = {'blend': 'average',
                       'clf': [{'clf': GradientBoostingRegressor(random_state=1),
                                'cvnum': 3,
                                'param': {'learning_rate': [0.1, 0.05, 0.02, 0.01],
                                          'n_estimators': [10, 100, 1000],
                                          'max_features': [1.0, 0.3, 0.1]}},
                               {'clf': GradientBoostingRegressor(random_state=2),
                                'cvnum': 3,
                                'param': {'learning_rate': [0.1, 0.05, 0.02, 0.01],
                                          'n_estimators': [10, 100, 1000],
                                          'max_features': [1.0, 0.3, 0.1]}},
                               {'clf': GradientBoostingRegressor(random_state=3),
                                'cvnum': 3,
                                'param': {'learning_rate': [0.1, 0.05, 0.02, 0.01],
                                          'n_estimators': [10, 100, 1000],
                                          'max_features': [1.0, 0.3, 0.1]}}]}
        clf = features.AverageBlender(blend_param, njobs=-1)

    elif classifier_baseid == '102':
        clf_type = 'r'
        blend_param = {'blend': 'average',
                       'clf': [{'clf': GradientBoostingRegressor(random_state=1),
                                'cvnum': 3,
                                'param': {'learning_rate': [0.1,0.05,0.02,0.01],
                                          'n_estimators': [10,100,1000],
                                          'max_features': [1.0,0.3,0.1]}},
                               {'clf': Ridge(),
                                'cvnum': 3,
                                'param': {'alpha': [0.001, 0.01, 0.05, 0.1, 0.2,
0.3]}}]}
        clf = features.AverageBlender(blend_param, njobs=-1)

    elif classifier_baseid == '103':
        clf_type = 'c'
        parameters = {'n_estimators': [30, 50, 100],
                      'max_features': [0.2, 0.4, 0.6, 0.8, 1.0],
                      'max_depth': [2, 3, None]}
        # clf = RandomForestClassifier(random_state=1, n_jobs=-1)
        clf = GridSearchCV(RandomForestClassifier(random_state=1),
                           parameters, scoring='roc_auc',
                           n_jobs=-1, cv=3, verbose=2)
```

```
elif classifier_baseid == '104':
    clf_type = 'c'
    parameters = {'n_estimators': [100, 200, 300],
                  'max_features': [0.2, 0.4, 0.6, 0.8, 1.0],
                  'max_depth': [2, 3, None]}
    clf = GridSearchCV(RandomForestClassifier(random_state=1),
                       parameters, scoring='roc_auc',
                       n_jobs=-1, cv=3, verbose=2)

elif classifier_baseid == '105':
    clf_type = 'c'
    parameters = {'n_estimators': [30, 50, 100],
                  'max_features': [0.2, 0.4, 0.6, 0.8, 1.0],
                  'max_depth': [2, 3, None]}
    clf = GridSearchCV(ExtraTreesClassifier(random_state=1),
                       parameters, scoring='roc_auc',
                       n_jobs=4, cv=3, verbose=2)

elif classifier_baseid == '106':
    clf_type = 'c'
    parameters = {'n_estimators': [100, 200, 300],
                  'max_features': [0.2, 0.4, 0.6, 0.8, 1.0],
                  'max_depth': [2, 3, None]}
    clf = GridSearchCV(ExtraTreesClassifier(random_state=1),
                       parameters, scoring='roc_auc',
                       n_jobs=-1, cv=3, verbose=2)

elif classifier_baseid == '107':
    clf_type = 'c'
    parameters = {'learning_rate': [0.05, 0.02, 0.01],
                  'n_estimators': [3000, 5000],
                  'max_features': [0.4, 0.3, 0.2]}
    clf = GridSearchCV(GradientBoostingClassifier(random_state=1),
                       parameters,scoring='roc_auc',
                       n_jobs=-1, cv=3, verbose=2)

elif classifier_baseid == '108':
    clf_type = 'r'
    parameters = {'learning_rate': [0.05, 0.02, 0.01],
                  'n_estimators': [3000, 5000],
                  'max_features': [0.4, 0.3, 0.2]}
    clf = GridSearchCV(GradientBoostingRegressor(random_state=1),
                       parameters,scoring='roc_auc',
                       n_jobs=-1, cv=3, verbose=2)

elif classifier_baseid == '109':
    clf_type = 'r'
    parameters = {'learning_rate': [0.05, 0.02, 0.01],
                  'n_estimators': [3000, 5000],
                  'max_features': [0.4, 0.3, 0.2]}
    clf = GridSearchCV(GradientBoostingRegressor(random_state=1),
                       parameters,scoring='roc_auc',
                       n_jobs=-1, cv=5, verbose=2)

elif classifier_baseid == '110':
    clf_type = 'r'
    parameters = {'learning_rate': [0.05, 0.02, 0.01],
                  'n_estimators': [3000, 5000],
                  'max_features': [0.4, 0.3, 0.2]}
    clf = GridSearchCV(GradientBoostingRegressor(random_state=2),
                       parameters,scoring='roc_auc',
                       n_jobs=-1, cv=5, verbose=2)

elif classifier_baseid == '111':
    clf_type = 'r'
    parameters = {'learning_rate': [0.05, 0.02, 0.01],
                  'n_estimators': [3000, 5000],
                  'max_features': [0.4, 0.3, 0.2]}
    clf = GridSearchCV(GradientBoostingRegressor(random_state=3),
                       parameters,scoring='roc_auc',
                       n_jobs=-1, cv=5, verbose=2)

elif classifier_baseid == '112':
    clf_type = 'c'
    parameters = {'n_estimators': [100, 500, 1000],
                  'learning_rate': [0.05, 0.02, 0.01],
                  'max_depth': [2, 3, 5],
                  'subsample': [0.4, 0.6, 0.8],
                  'colsample_bytree': [0.4, 0.6, 0.8]}
    clf = GridSearchCV(xgb.XGBClassifier(),
                       parameters, scoring='roc_auc',
                       n_jobs=1, cv=3, verbose=2)


else:
    print("Invalid classifier_baseid: %s" % (classifier_baseid))
    sys.exit()

pos_label = 1
test_fold_list = []
train_fold_list = []
best_params_list = []
for i, (train_idx, test_idx) in enumerate(KFold(len(X), n_folds=cvnum, shuffle=True,
random_state=1)):
    X_train, X_test = X[train_idx], X[test_idx]
    y_train, y_test = y[train_idx], y[test_idx]
    clf.fit(X_train, y_train)
```

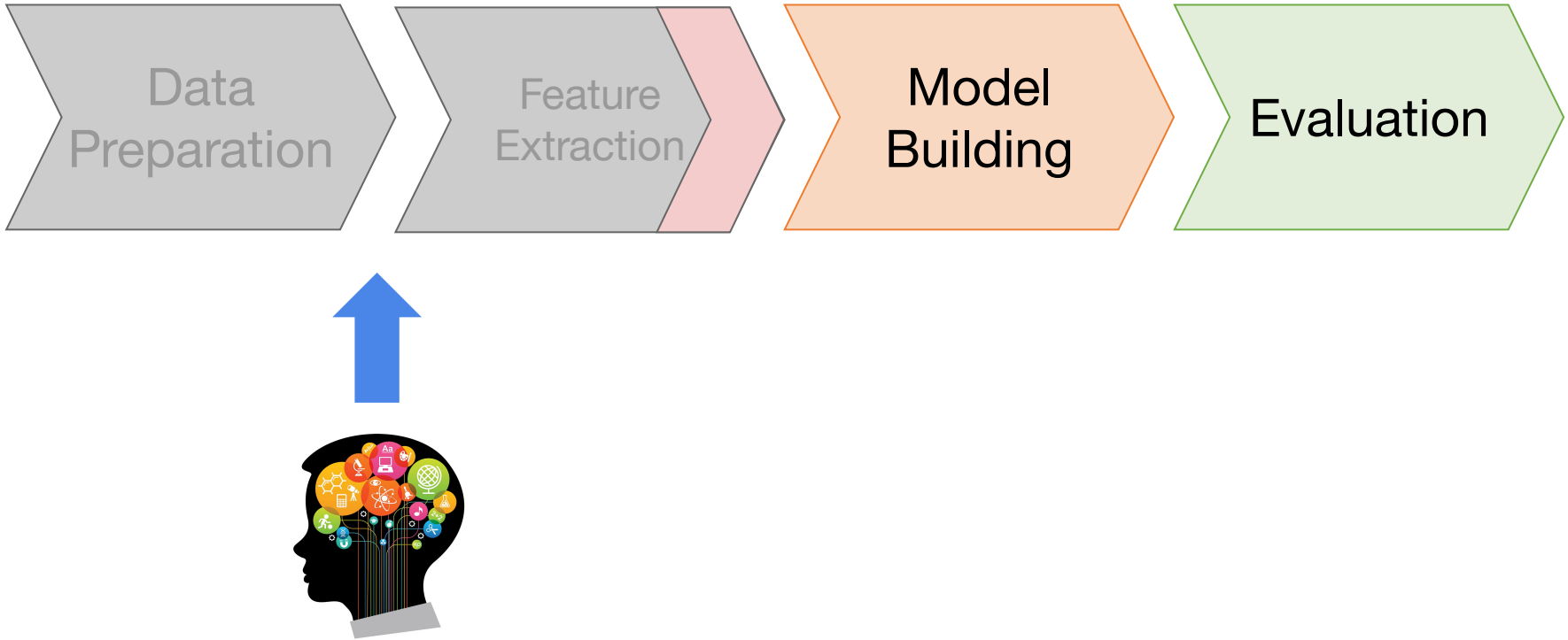
```
        if clf_type == 'r':
            y_pred = clf.predict(X_test)
            y_pred_train = clf.predict(X_train)
        elif clf_type == 'c':
            if classifier_baseid == '112':
                pos_idx = 1
            else:
                pos_idx = np.where(clf.best_estimator_.classes_ == pos_label)[0][0]
            y_pred = clf.predict_proba(X_test)[:, pos_idx]
            y_pred_train = clf.predict_proba(X_train)[:,pos_idx]

        fpr, tpr, _ = sklearn.metrics.roc_curve(y_test, y_pred, pos_label=1)
        test_fold_list.append(sklearn.metrics.auc(fpr, tpr))

        fpr_train, tpr_train, _ = sklearn.metrics.roc_curve(y_train, y_pred_train,
pos_label=1)
        train_fold_list.append(sklearn.metrics.auc(fpr_train, tpr_train))

        best_params_list.append(clf.best_params_)

    test_fold_array = np.array(test_fold_list)
    train_fold_array = np.array(train_fold_list)

    result = {'test':
              {'auc':
               {'fold': test_fold_list,
                'mean': test_fold_array.mean(),
                'std': test_fold_array.std()}},
              'train':
              {'auc':
               {'fold': train_fold_list,
                'mean': train_fold_array.mean(),
                'std': train_fold_array.std()}},
              'best_params_': best_params_list}

    with open(output_filepath, 'w') as fout:
        json.dump(result, fout)
```

After: auto-sklearn

```
from autosklearn.classification import AutoSklearnClassifier()
clf = AutoSklearnClassifier()
clf.fit(X_train, y_train)
predictions = clf.predict(X_test, y_test)
```



But, feature engineering is still important!



Hands-on

Google Colab

```
def run_cv(X, y, clf, num_classes):
  kf = KFold(n_splits=5, random_state=1)
  cm = np.zeros([num_classes,
                 num_classes],
                 dtype="int") # Initialize confusion matrix with 0
  f1_list = []
  for i, (train_index, test_index) in enumerate(kf.split(X)):
    print("Fold {}".format(i + 1))
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    cur_clf = clone(clf)
    cur_clf.fit(X_train, y_train)
    y_pred = cur_clf.predict(X_test)
    cm += confusion_matrix(y_test, y_pred)
    f1_list.append(f1_score(y_test, y_pred, average="macro"))
  f1_scores = np.array(f1_list)
  return (f1_scores, cm)
```

Try!

- Let's try to improve the happiness classification model using different features
  - CountVectorizer()
  - TfidfVectorizer()
  - LatentDirichletAllocation()
  - …
  - Feature engineering!