

NATURAL LANGUAGE PROCESSING AND TEXT CLASSIFICATION

Abbas Chokor, Ph.D.

Staff Data Scientist, Seagate Technology

OUR PROGRESS SO FAR

UNIT 1: RESEARCH DESIGN AND EXPLORATORY DATA ANALYSIS

What is Data Science	Lesson 1
Research Design and Pandas	Lesson 2
Statistics Fundamentals I	Lesson 3
Statistics Fundamentals II	Lesson 4
Flexible Class Session	Lesson 5

UNIT 2: FOUNDATIONS OF DATA MODELING

Introduction to Regression	Lesson 6
Evaluating Model Fit	Lesson 7
Introduction to Classification	Lesson 8
Introduction to Logistic Regression	Lesson 9
Communicating Logistic Regression Results	Lesson 10
Flexible Class Session	Lesson 11

UNIT 3: DATA SCIENCE IN THE REAL WORLD

Decision Trees and Random Forests	Lesson 12
› Natural Language Processing	Lesson 13
› Dimensionality Reduction	Lesson 14
› Time Series Data I	Lesson 15
› Time Series Data II	Lesson 16
› Database Technologies	Lesson 17
› Where to Go Next	Lesson 18
› Flexible Class Session	Lesson 19
› Final Project Presentations	Lesson 20



Today's Class

LAST CLASS

WHAT DID WE LEARN?

- Understand and build **decision tree** models for classification and regression
- Understand and build **random forest** models for classification and regression
- Know how to extract the most **important predictors** in a random forest model



You got all objectives?



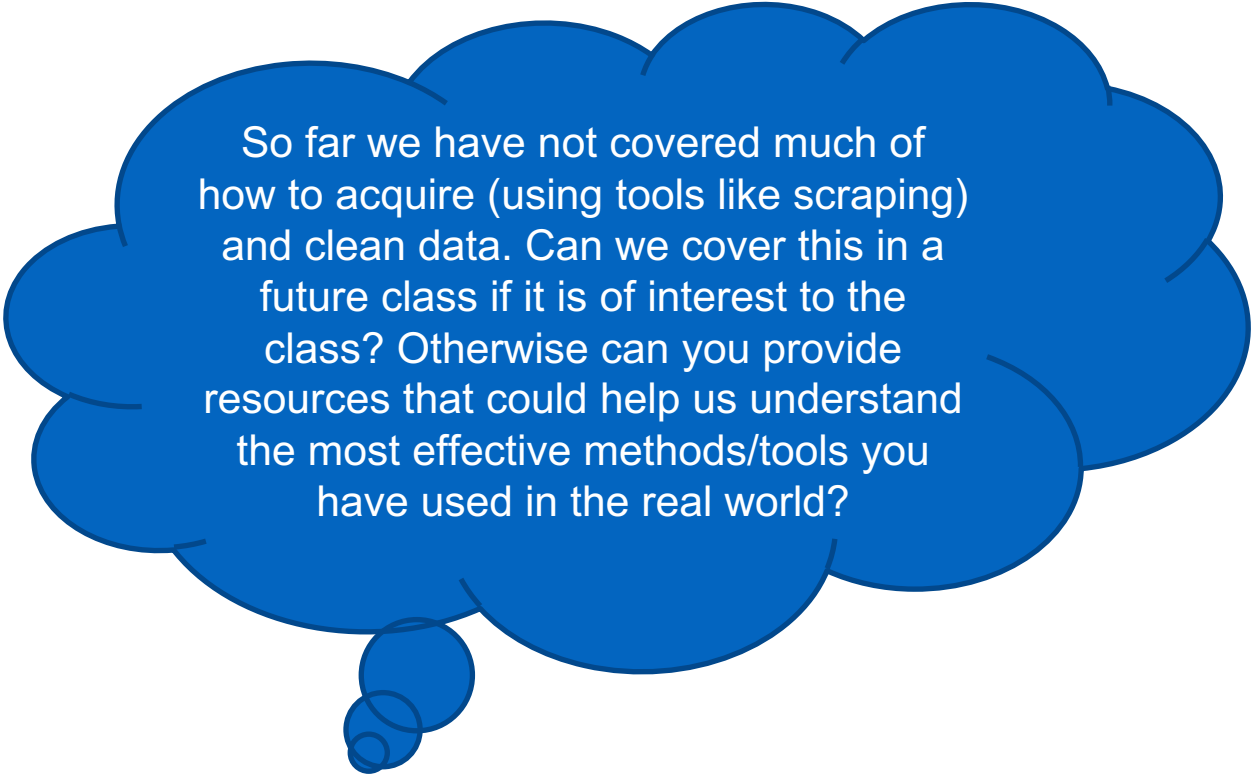
Not all of them...

Let's form groups of 1's and 2's ...

LAST CLASS

ANNOUNCEMENTS

- ❖ Send me a message if you will be joining through WebEx
- ❖ Next Class is January 3rd



So far we have not covered much of how to acquire (using tools like scraping) and clean data. Can we cover this in a future class if it is of interest to the class? Otherwise can you provide resources that could help us understand the most effective methods/tools you have used in the real world?

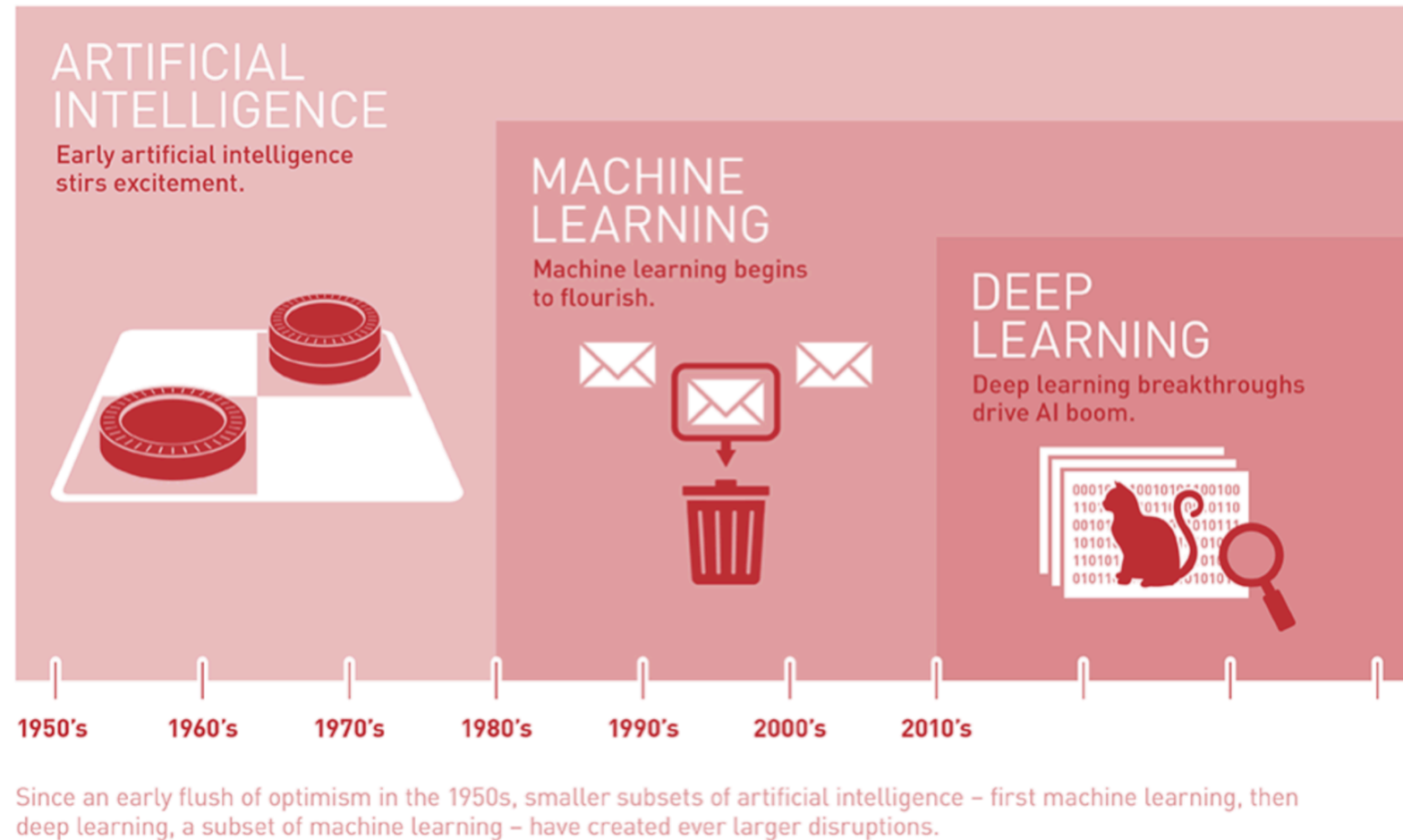


Others?

WHAT IS MACHINE LEARNING

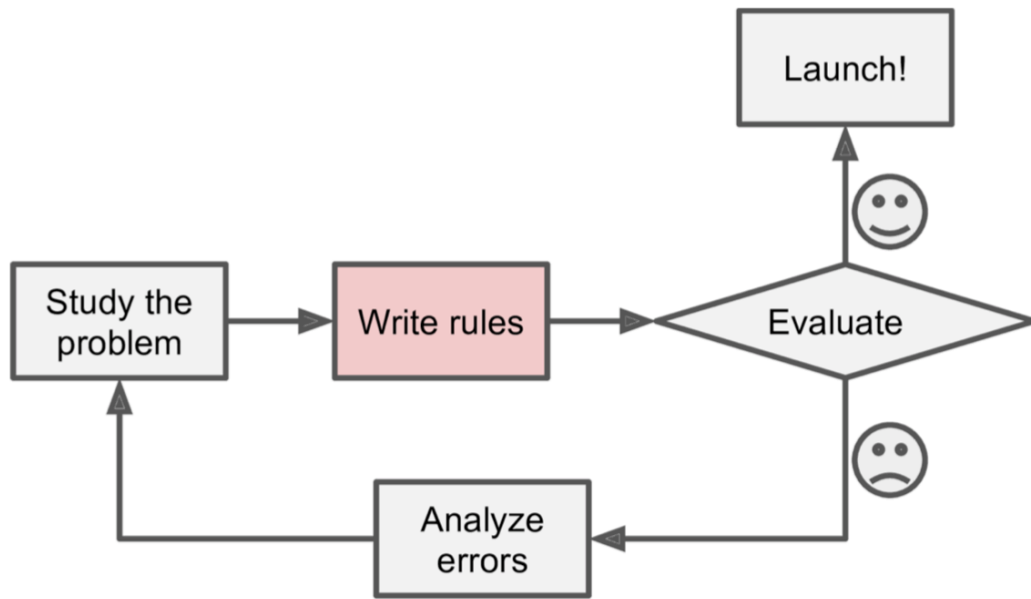
[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959



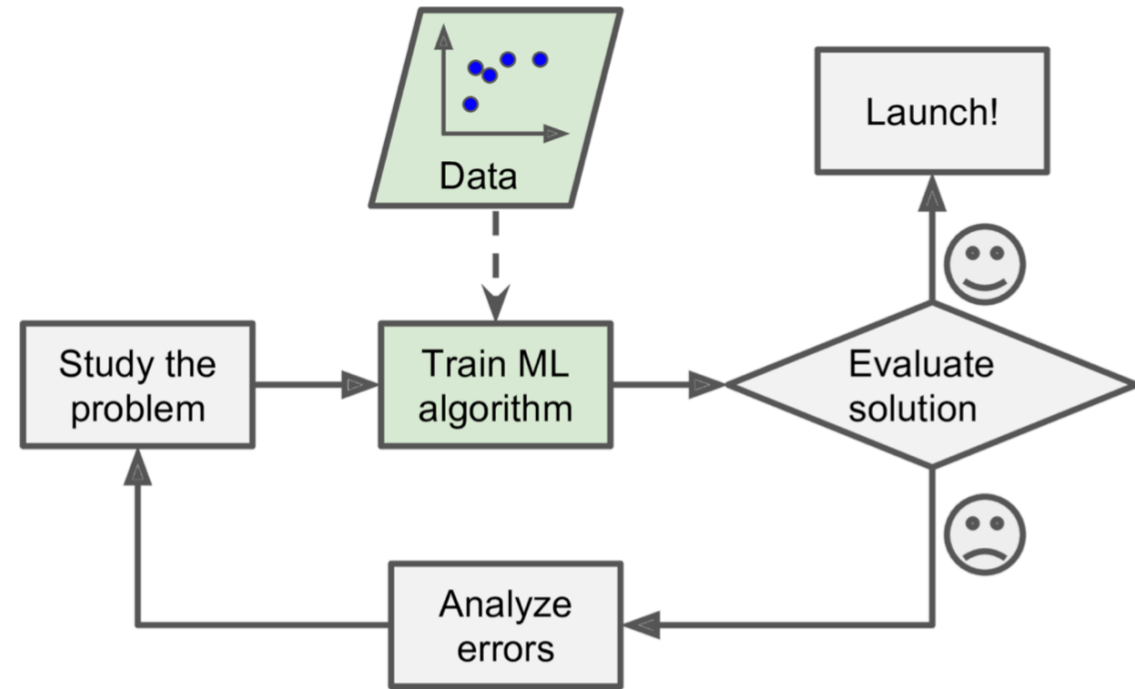
Machine Learning: is the science (and art) of programming computers so they can *learn from data*.

WHY TO USE MACHINE LEARNING?



Traditional approach

Vs.



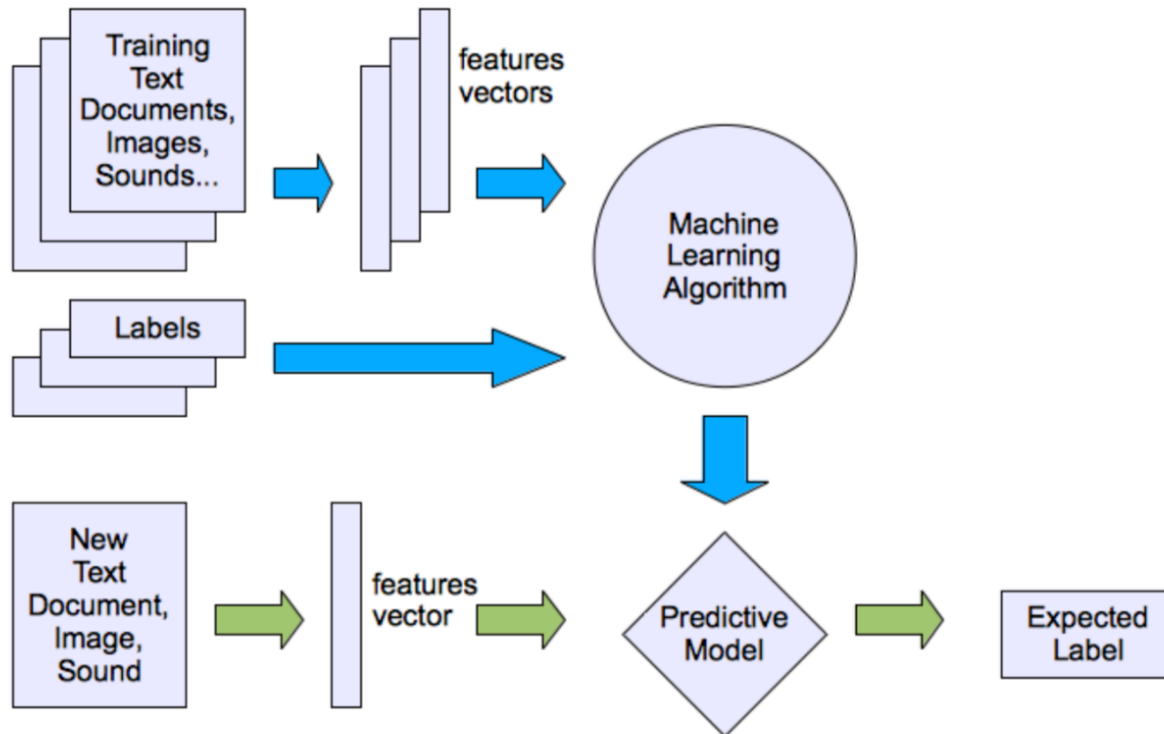
Machine Learning approach

Did you know?

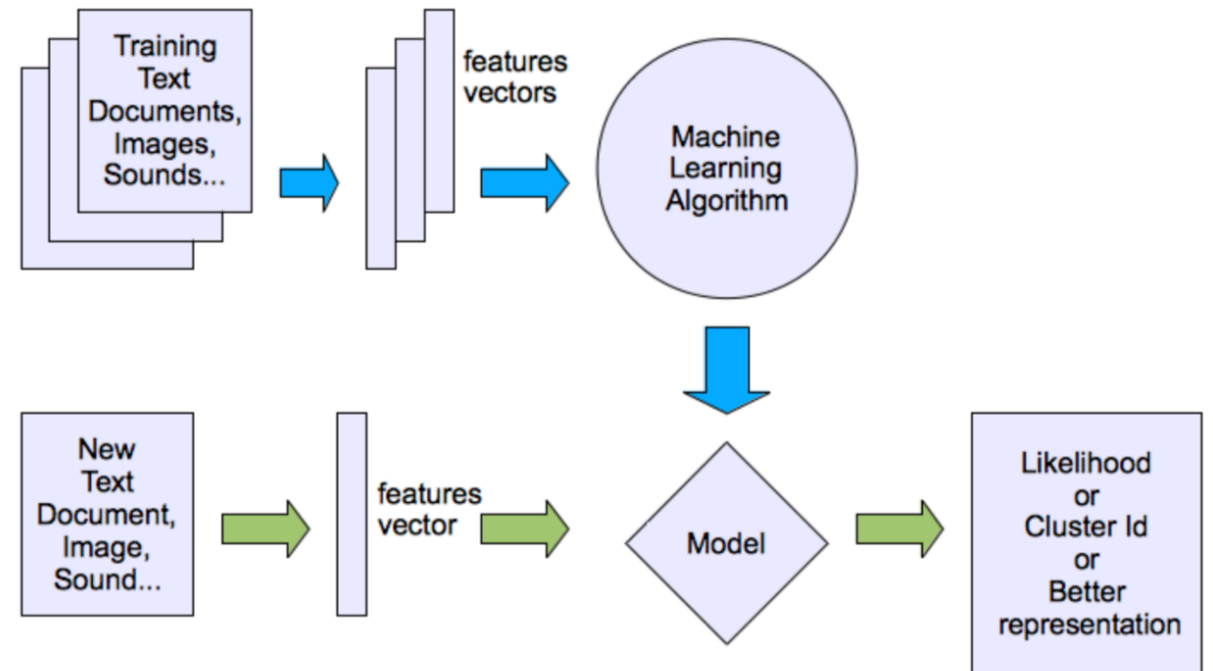
Machine learning algorithms are expected to replace 25% of the jobs across the world, in the next 10 years.

TYPES OF MACHINE LEARNING

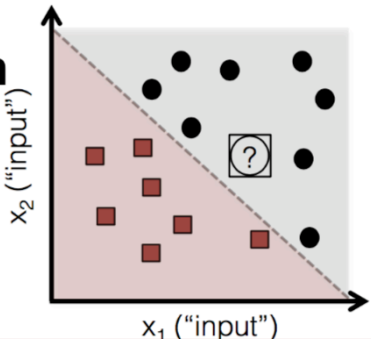
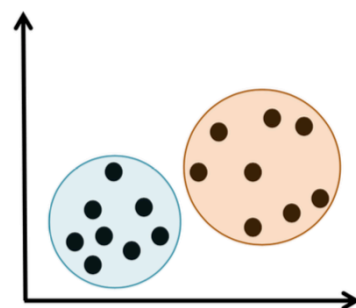
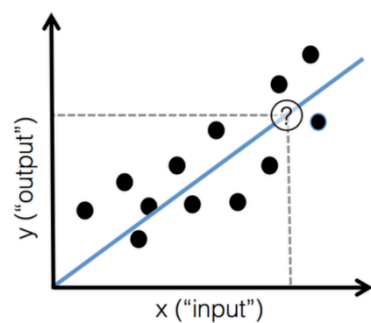
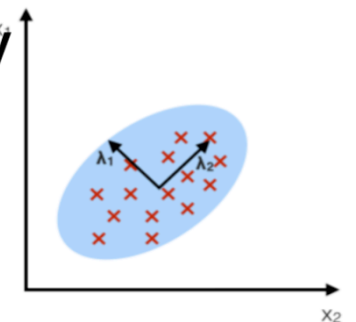
Supervised



Unsupervised



TYPES OF MACHINE LEARNING

	Supervised Working with Labeled Data	Unsupervised Working with Unlabeled Data
Discrete Countable Data	Classification 	Clustering 
Continuous Infinite Data	Regression 	Dimensionality Reduction 

LEARNING OBJECTIVES

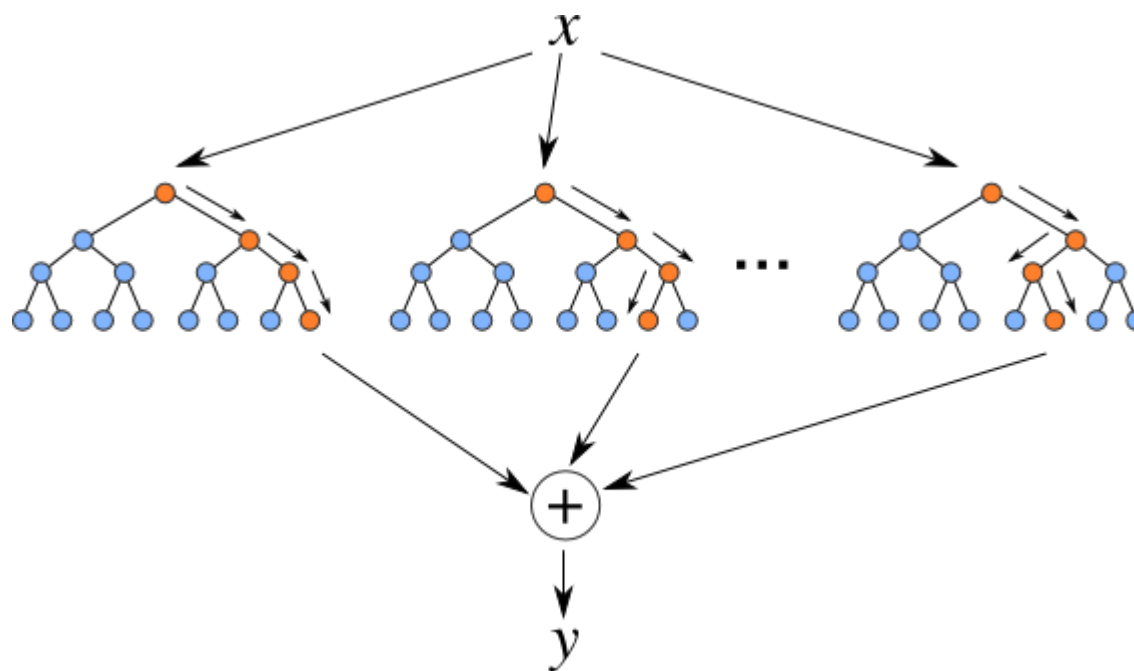
- Define natural language processing
- List common tasks associated with
 - NLP Use-cases
 - Tokenization
 - Lemmatization and Stemming
 - Tagging and parsing
- Demonstrate how to classify text or documents using scikit-learn

COURSE

HOW TO PUT EVERYTHING TOGETHER?

REVIEW: DECISION TREES AND RANDOM FORESTS

- Decision trees are models that ask a series of questions. The next question depends upon the answer to the previous question.
- Random forest models are ensembles of decision trees that are randomized in the way they are created.



PRE-WORK REVIEW

- Experience with scikit-learn classifiers, specifically:
 - ❖ Logistic Regression
 - ❖ KNN Classifier
 - ❖ Decision trees
 - ❖ Random forests
- How to put everything together?
- Let's go to Lab13 in-class group exercise review

OPENING

NATURAL LANGUAGE PROCESSING AND TEXT CLASSIFICATION

PRE-WORK REVIEW

- Install the Python package `spacy` with `pip install spacy`
- Run the `spacy` download data command

```
python -m spacy.en.download --force all
```

Note: you can simply download it from conda using

```
conda install -c spacy spacy
```

WHAT IS NATURAL LANGUAGE PROCESSING (NLP)?

- Natural language processing is the task of extracting meaning and information from text documents.
- There are many types of information we might want to extract.
- These tasks may range from simple classification tasks, such as deciding what category a piece of text falls into, to more complex tasks like translating or summarizing text.
- For most tasks, a fair amount of pre-processing is required to make the text digestible for our algorithms. We typically need to *add structure* to our *unstructured data*.

WHAT IS NATURAL LANGUAGE PROCESSING (NLP)?

- Many AI assistant systems are typically powered by fairly advanced NLP engines.
- A system like Siri uses voice-to-transcription to record a command and then various NLP algorithms to identify the question asked and possible answers.



What are some real-world examples of NLP?

- Search engines (E.g., Google and Bing)
- Natural language assistants (E.g., Apple's Siri uses voice recognition to record a command and then various fairly advanced NLP engines to identify the question asked and possible answers)
- Machine translation (E.g., Google Translate)
- Question answering (E.g., IBM's Watson)
- News digest (E.g., Yahoo!)

TOKENIZATION

- Tokenization is the task of separating a sentence into its constituent parts, or *tokens*.
- Determining the “words” of a sentence seems easy but can quickly become complicated with unusual punctuation (common in social media) or different language conventions.

TOKENIZATION

- What sort of difficulties can you find in the following sentence?
- The L.A. Lakers won the NBA championship in 2010, defeating the Boston Celtics.

TOKENIZATION EXAMPLES

My house is located in Uptown. → [My, house, is, located, in, Uptown]

The Lakers are my favorite team. → [The, Lakers, are, my, favorite, team]

Data Science is the future! → [Data, Science, is, the, future]

GA has many locations. → [GA, has, many, locations.]

LEMMATIZATION AND STEMMING

- How would you describe the relationship between the terms ‘bad’ and ‘badly’ or ‘different’ and ‘differences’?
- *Stemming* and *lemmatization* help identify common roots of words.

LEMMATIZATION AND STEMMING

- ***Stemming*** is a crude process of removing common endings from sentences, such as ‘s’, ‘es’, ‘ly’, ‘ing’, and ‘ed’.
- To stem a word is to reduce it to a base form, called the *stem*, after removing various suffixes and endings and, sometimes, performing some additional transformations.
- In practice, prefixes are sometimes preserved, so ‘rescan’ will not be stemmed to ‘scan’.

LEMMATIZATION AND STEMMING

- ***Lemmatization*** is a more refined process that uses specific language and grammar rules to derive the root of a word.
- This is useful for words that do not share an obvious root such as ‘better’ and ‘best’.
- What are some other examples of words that do not share an obvious root?

LEMMATIZATION AND STEMMING EXAMPLES

Lemmatization

shouted → shout

best → good

better → good

good → good

wiping → wipe

hidden → hide

Stemming

badly → bad

computing → comput

computed → comput

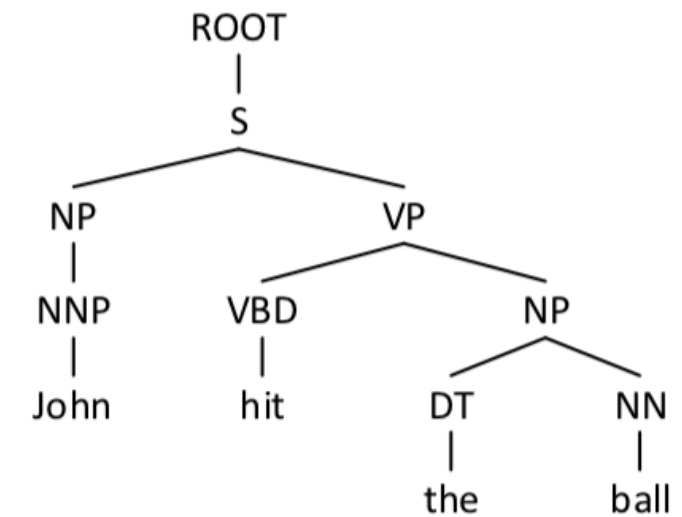
wipes → wip

wiped → wip

wiping → wip

PARSING AND TAGGING

- In order to understand the various elements of a sentence, we need to *tag* important topics and *parse* their dependencies.
- Our goal is to identify the *actors* and *actions* in the text in order to make informed decisions.



DT - Determiner
NN - Noun, singular or mass
NNP - Proper noun, singular
NP - Noun phrase
S - Simple declarative clause
VBD - Verb, past tense
VP - Verb phrase

PARSING AND TAGGING

- If we are processing financial news, we might need to identify which companies are involved and which actions they are taking.
- If we are writing an assistant application, we might need to identify specific command phrases in order to determine what is being asked:
 - e.g. “Siri, when is my next appointment?”

PARSING AND TAGGING

- Tagging and parsing is made up of a few overlapping subproblems:
 - “Parts of speech” tagging: What are the parts of speech in a sentence (e.g. noun, verb, adjective, etc)?
 - Chunking: Can we identify the pieces of the sentence that go together in meaningful chunks (e.g. noun or verb phrases)?
 - Named entity recognition: Can we identify *specific* proper nouns? Can we pick out people and locations?

PARSING AND TAGGING

Tagging

John/NNP hit/VBD the/DT ball/NN

Parsing

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBD hit)
      (NP (DT the) (NN ball))))))
```

DEMO

NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

- Most NLP techniques require pre-processing large collections of annotated text in order to learn specific language rules.
- There are many tools available for English and other popular languages.
- Each tool typically requires a large amount of data and large databases of special use-cases, including language inconsistencies and slang.
- In Python, two popular NLP packages are `nltk` and `spacy`.
- `nltk` is more popular but not as advanced and well maintained. `spacy` is more modern but not available for commercial use.

NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

- We'll be using spacy to process some news article titles. First load the NLP toolkit by specifying the language. Go to LAB 13 Starter:

```
from spacy.en import English  
nlp_toolkit = English()
```

- This toolkit has 3 pre-processing engines:
 - A tokenizer to identify the word tokens
 - A lemmatization / stemming to get root words
 - A tagger to identify the concepts described by the words
 - A parser to identify the phrases and links between different words

NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

- The first title is “[IBM Sees Holographic Calls, Air Breathing Batteries](#)”.
- From this, we may want to extract several pieces of information: this title references a company and that company is referencing a new possible product: air-breathing batteries.



NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

- We can use spacy to get information about this title (GO TO STARTER)

```
title = "IBM sees holographic calls, air breathing batteries"
parsed = nlp_toolkit(title)
```

```
for (i, word) in enumerate(parsed):
    print("Word: {}".format(word))
    print("\t Phrase type: {}".format(word.dep_))
    print("\t Is the word a known entity type? {}".format(word.ent_type_) if
word.ent_type_ else "No"))
    print("\t Lemma: {}".format(word.lemma_))
    print("\t Parent of this word: {}".format(word.head.lemma_))
```

- `nlp_toolkit` runs each of the individual pre-processing tools.

NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

- The output will look similar to this:

Word: IBM

Phrase type: nsubj

Is the word a known entity type? ORG

Lemma: ibm

Parent of this word: see

Word: sees

Phrase type: ROOT

Is the word a known entity type? No

Lemma: see

Parent of this word: see

Word: holographic

Phrase type: amod

Is the word a known entity type? No

Lemma: holographic

Parent of this word: call

NATURAL LANGUAGE PROCESSING WITH ‘SPACY’

- In this output:
 - “IBM” is identified as an organization (ORG).
 - We identify a phrase: “holographic calls”.
 - We identify a compound noun phrase: “air breathing batteries”.
 - We identify that “see” is at the root as an action “IBM” is taking.
 - We can see that “batteries” was lemmatized to “battery”.

COMMON PROBLEMS IN NLP

- These subtasks are very difficult, because language is complex and changes frequently.
- Most often, we are looking for heuristics to search through large amounts of text data. The results may not be perfect... and that's okay!
- Older techniques rely on rule-based systems. More recent techniques use flexible systems, focusing on the words used rather than the structure of the sentence.
- We'll see an example of these modern approaches in the next class.

INTRODUCTION

TEXT CLASSIFICATION

TEXT CLASSIFICATION

- Text classification is the task of predicting which category or topic a text sample is from.
- For example, we may want to identify whether an article is a sports or business story. Or whether an article has positive or negative sentiment.
- Typically, this is done by using the text as features and the label as the target output. This is referred to as *bag-of-words* classification.
- To include text as features, we usually create a *binary* feature for each word, i.e. does this piece of text contain that word?

TEXT CLASSIFICATION

- To create binary text features, we first create a vocabulary to account for all possible words in our universe.
- As we do this, we need to consider several things.
 - Does order of words matter?
 - Does punctuation matter?
 - Does upper or lower case matter?

TEXT CLASSIFICATION

▸ This table illustrates features created from the following passage.

“It’s a great advantage not to drink among hard drinking people.”

Feature	Value
it’s	1
great	1
good	0
advantage	1
not	1
think	0
drink	1
from	0
hard	1
drinking	1

Feature	Value
people	1
withhold	0
random	0
smoke	0
among	1
whenever	0
thoughtful	0
inexhaustible	0
men	0
Nick	0

ACTIVITY: KNOWLEDGE CHECK

ANSWER THE FOLLOWING QUESTIONS

Discuss your answers to the following questions and explain your reasoning.

1. Does word order matter?
2. Does word case (e.g. upper or lower) matter?
3. Does punctuation matter?

DELIVERABLE

Answers to the above questions



EXERCISE

DEMO

TEXT PROCESSING IN SCIKIT-LEARN

TEXT PROCESSING IN SCIKIT-LEARN

- Scikit-learn has many pre-processing utilities that simplify tasks required to convert text into features for a model.
- These can be found in the `sklearn.preprocessing.text` package.
- We will use the StumbleUpon dataset again to perform text classification. This time, we will use the text content itself to predict whether a page is 'evergreen' or not.
- Open the starter code notebook to follow along.

COUNTVECTORIZER

- CountVectorizer converts a collection of text into a matrix of features. Each row will be a sample (an article or piece of text) and each column will be a text feature (usually a count or binary feature per word).
- CountVectorizer takes a column of text and creates a new dataset. It generates a feature for every word in all of the pieces of text.
- **REMEMBER:** Using all of the words can be useful, but we may need to use *regularization* to avoid overfitting. Otherwise, rare words may cause the model to overfit and not generalize.

COUNTVECTORIZER

- Instantiate a new CountVectorizer.

```
from sklearn.feature_extraction.text import CountVectorizer  
  
vectorizer = CountVectorizer(max_features = 1000,  
                             ngram_range=(1, 2),  
                             stop_words='english',  
                             binary=True)
```

COUNTVECTORIZER PARAMETERS

- There are several parameters to utilize.
- `ngram_range` - a range of word phrases to use
 - `(1, 1)` means use all single words
 - `(1, 2)` means use all contiguous pairs of word
 - `(1, 3)` means use all triples
- `stop_words='english'`
 - Stop words are non-content words (e.g. 'to', 'the', 'it', etc). They aren't helpful for prediction, so they get removed.

COUNTVECTORIZER PARAMETERS

- `max_features=1000`
 - Maximum number of words to consider (uses the first N most frequent)
- `binary=True`
 - To use a dummy column as the entry (1 or 0, as opposed to the count). This is useful if you think a word appearing 10 times is no more important than whether the word appears at all.

COUNTVECTORIZER

- Vectorizers are like other models in scikit-learn.
 - We create a vectorizer object with the parameters of our feature space.
 - We fit a vectorizer to learn the vocabulary.
 - We transform a set of text into that feature space.

COUNTVECTORIZER

- Note: there is a distinction between *fit* and *transform*.
 - We **fit** from our training set. This is part of the model building process, so we don't look at our test set.
 - We **transform** our test set using our model fit on the training set.

COUNTVECTORIZER EXAMPLE

```
titles = data['title'].fillna('')
```

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer = CountVectorizer(max_features = 1000,  
                             ngram_range=(1, 2),  
                             stop_words='english',  
                             binary=True)
```

```
# Use `fit` to learn the vocabulary of the titles  
vectorizer.fit(titles)
```

```
# Use `transform` to generate the sample X word matrix - one column per  
feature (word or n-grams)  
X = vectorizer.transform(titles)
```

RANDOM FOREST PREDICTION MODEL

- We can now build a random forest model to predict “evergreenness”.

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(n_estimators = 20)
```

```
# Use `fit` to learn the vocabulary of the titles vectorizer.fit(titles)
```

```
# Use `transform` to generate the sample X word matrix - one column per feature (word or n-grams)
```

```
X = vectorizer.transform(titles)
```

```
y = data['label']
```

```
from sklearn.cross_validation import cross_val_score
```

```
scores = cross_val_score(model, X, y, scoring='roc_auc')
```

```
print('CV AUC {}, Average AUC {}'.format(scores, scores.mean()))
```

TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY

- An alternative *bag-of-words* approach to CountVectorizer is a Term Frequency - Inverse Document Frequency (TF-IDF) representation.
- TF-IDF uses the product of two intermediate values, the *Term Frequency* and *Inverse Document Frequency*.

TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY

- *Term Frequency* is equivalent to CountVectorizer features, just the number of times a word appears in the document (i.e. count).
- *Document Frequency* is the percentage of documents that a particular word appears in.
- For example, “the” would be 100% while “Peru” is much lower.
- *Inverse Document Frequency* is just $1/\text{Document Frequency}$.

TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY

- Combining, $\text{TF-IDF} = \text{Term Frequency} * \text{Inverse Document Frequency}$ or $\text{TF-IDF} = \text{Term Frequency} / \text{Document Frequency}$
- The intuition is that the words that have high weight are those that either appear ***frequently*** in this document or appear ***rarely*** in other documents (and are therefore unique to this document).
- This is a good alternative to using a static set of “stop” words.

```
from sklearn.feature_extraction.text import TfidfVectorizer  
vectorizer = TfidfVectorizer()
```

TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY

Term Frequency (TF)

- › *Term Frequency* is equivalent to `CountVectorizer` features but using frequencies, not counts

$$tf(t, d) = \frac{\text{number of occurrences of term } t \text{ in document } d}{\text{number of terms in document } d}$$

- › *Term Frequency* assigns high weight to frequent words (words that appear frequently) in a documents

Inverse Document Frequency (IDF)

- › *Document Frequency* is the percentage of documents that a particular word appears in
- › *Inverse Document Frequency* is *Document Frequency*'s inverse

$$idf(t, D) = \frac{\text{total number of documents } D}{\text{number of documents term } t \text{ appears in them}}$$

- › *Inverse Document Frequency* assigns high weight to rare words in all the documents

ACTIVITY: KNOWLEDGE CHECK



ANSWER THE FOLLOWING QUESTIONS

1. What does TF-IDF stand for?
2. What does this function do and why is it useful?
3. Use `TfidfVectorizer` to create a feature representation of the StumbleUpon titles.

DELIVERABLE

Answers to the above questions and feature representation

INDEPENDENT PRACTICE

TEXT CLASSIFICATION IN SCIKIT-LEARN

ACTIVITY: TEXT CLASSIFICATION IN SCIKIT-LEARN



EXERCISE

DIRECTIONS (30 minutes)

1. Use the text features of `title` with one or more feature sets from the previous random forest model. Train this model to see if it improves AUC.
2. Use the body text instead of the `title`. Does this give an improvement?
3. Use `TfidfVectorizer` instead of `CountVectorizer`. Does this give an improvement?

Check: Were you able to prepare a model that uses both quantitative features and text features? Does this model improve the AUC?

DELIVERABLE

Three new models

CONCLUSION

TOPIC REVIEW

LET'S REVIEW

- Natural language processing (NLP) is the task of pulling meaning and information from text.
- This typically involves many subproblems including tokenization, cleaning (stemming and lemmatization), and parsing.
- After we have structured our text, we can identify features for other tasks, including classification, summarization, and translation.
- In scikit-learn, we use vectorizers to create text features for classification, such as `CountVectorizer` and `TfidfVectorizer`.

OUR PROGRESS SO FAR

UNIT 1: RESEARCH DESIGN AND EXPLORATORY DATA ANALYSIS

What is Data Science	Lesson 1
Research Design and Pandas	Lesson 2
Statistics Fundamentals I	Lesson 3
Statistics Fundamentals II	Lesson 4
Flexible Class Session	Lesson 5

UNIT 2: FOUNDATIONS OF DATA MODELING

Introduction to Regression	Lesson 6
Evaluating Model Fit	Lesson 7
Introduction to Classification	Lesson 8
Introduction to Logistic Regression	Lesson 9
Communicating Logistic Regression Results	Lesson 10
Flexible Class Session	Lesson 11

UNIT 3: DATA SCIENCE IN THE REAL WORLD

Decision Trees and Random Forests	Lesson 12
Natural Language Processing	Lesson 13
Dimensionality Reduction	Lesson 14
Time Series Data I	Lesson 15
Time Series Data II	Lesson 16
Database Technologies	Lesson 17
Where to Go Next	Lesson 18
Flexible Class Session	Lesson 19
Final Project Presentations	Lesson 20



LESSON

Q & A

LESSON

EXIT TICKET

DON'T FORGET TO FILL OUT YOUR EXIT TICKET