Text Classification

Instructor: Jackie CK Cheung

COMP-550

Reading: J&M Chapter 4 (3rd ed.)

Ed Discussion

Make sure you are enrolled on ed and can access the course materials.

Recap of Last Lecture

Introduction to **computational linguistics** and **natural language processing**

- NLU vs. NLG

Goals of CL / NLP

- Technology and applications
- Understanding natural language through computational modelling
- Mathematical analysis of algorithms and formalism

Subfields of Linguistics

Match the following terms to their object of study

Semantics	Speech sounds
Pragmatics	Literal meaning
Discourse	Sound patterns
Syntax	Implied meaning
Phonology	Word structure
Phonetics	Sentence structure
Morphology	Passage structure

Risks and uncertainties abound, as restructuring is slowed by legal difficulties.

Will taking NLP make you rich?

This company has solid fundamentals and good growth prospects.

Text Classification

Assigning a label or category to a piece of text

Above is an example of **sentiment analysis**

Other common text classification tasks:

- **Spam detection**
- **Language identification**
- **Authorship attribution**

Outline

Machine learning basics

- Basic definitions
- Experimental procedure in NLP

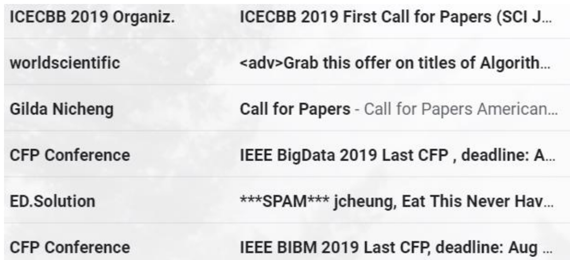
Text classification

- Experimental methodology
- Feature extraction

Machine Learning for NLP

Common research paradigm:

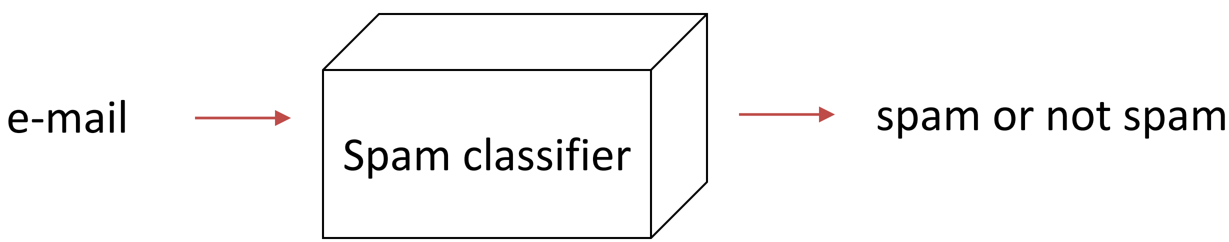
1. Find interesting NLP problem from language data or need



ICECBB 2019 Organiz.	ICECBB 2019 First Call for Papers (SCI J...
worldscientific	<adv>Grab this offer on titles of Algorith...
Gilda Nicheng	Call for Papers - Call for Papers American...
CFP Conference	IEEE BigData 2019 Last CFP , deadline: A...
ED.Solution	***SPAM*** jcheung, Eat This Never Hav...
CFP Conference	IEEE BIBM 2019 Last CFP, deadline: Aug ...

Which e-mails are spam?

2. Formulate NLP problem as machine learning problem



3. Solve problem by using machine learning techniques

Supervised vs. Unsupervised Learning

How much information do we give to the machine learning model?

Supervised – model has access to some input data, and their corresponding output data (e.g., a label)

- Learn a function y = f(x), given examples of (x, y) pairs

input

output

model

Unsupervised – model only has the input data

- Given only examples of x, find some interesting patterns in the data

Supervised Learning Examples

1. Predict whether an e-mail is spam or non-spam (given examples of spam and non-spam e-mails)

2. Given examples, predict the **part of speech** (POS) of a word
   - *run* is a verb (or a noun)
   - *ran* is a verb
   - *cat* is a noun
   - *the* is a determiner

What Does Learning Mean?

Supervised setting: determining what the function f(x) should be, given the data.

- i.e., find parameters to the model θ that minimize some kind of **loss** or **error** function
- For example, the model should minimize the number of incorrectly classified pairs in the training set.

Regression vs. Classification

Supervised learning maps input x to output y:

$$y = f(x)$$

Can distinguish based on property of y:

- **Regression**: y is a continuous outcome

  e.g., similarity score of 3.5

- **Classification**: y is a discrete outcome

  e.g., spam vs. non-spam, verb vs. noun vs. adjective, etc.

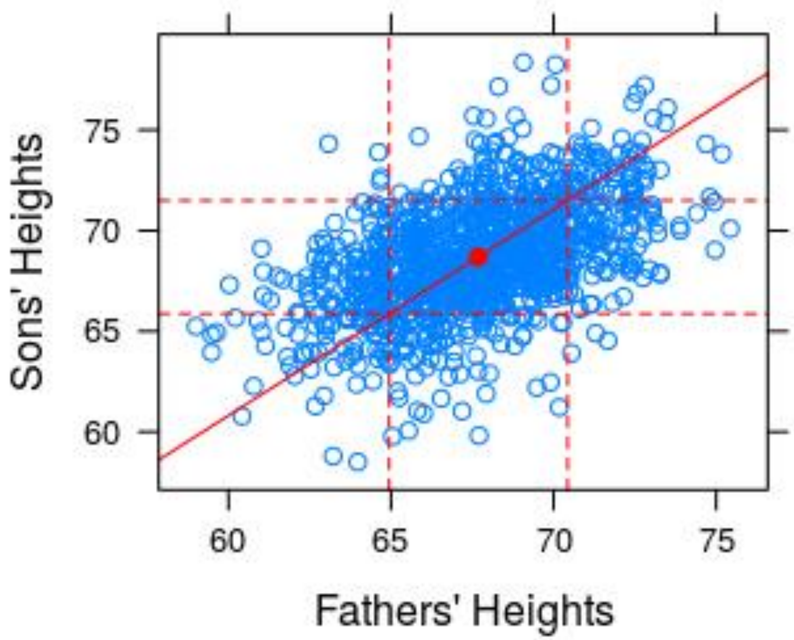
Linear Regression

The function is linear:

$$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b$$

Line of best fit:

- Galton plotted son's height to father's height



Classification

Most NLP work involving text end up being classification problems.

Linguistic units of interest are often discrete:

• words:	*apple, banana, orange*
• POS tags:	NOUN, VERB, ADJECTIVE
• semantic categories	AGENT, PATIENT, EXPERIENCER
• discourse relations	EXPLANATION, CAUSE, ELABORATION

Unsupervised Learning

Find hidden structure in the data without any labels.

1. Grammar induction
   - *the* and *a* seem to appear in similar contexts
   - *very* and *hope* don't appear in similar contexts
   - Cluster *the* and *a* into the same POS, *very* and *hope* into different ones
2. Learning word relatedness
   - *cat* and *dog* are related words with similarity score 0.81
   - *good* and *bad* are related words with similarity score 0.56

Will return to this later in the course

Steps in Building a Text Classifier

1. **Define problem and collect data set**
2. Extract features from documents
3. Train a classifier on a training set
4. Apply classifier on test data

Problem Definition

$$y = f(\vec{x})$$

input document

document label

classifier

Some basic decisions:

- What is the problem being solved?
- What is the input to the model?
- What are the output categories?
- How do we get annotated data of this format?

This is actually a big part of the NLP problem!

Steps in Building a Text Classifier

1. Define problem and collect data set
2. **Extract features from documents**
3. Train a classifier on a training set
4. Apply classifier on test data

Feature Extraction

Need to extract properties from the document that might give a clue to its category label

"proposal that would be beneficial to you"

"please can I talk to you ??"

"legitimate business of $21,300.000."

Feature Extraction

$$y = f(\vec{x})$$

document label

classifier

document

Represent document $\vec{x}$ as a list of features

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \ldots$

1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0 …

Feature Extraction and Classification

We can use these feature vectors to train a classifier

Training set:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8 …	y
1.0,	0.0,	1.0,	1.0,	0.0,	0.0,	1.0,	1.0 …	1
1.0,	0.0,	0.0,	1.0,	1.0,	0.0,	0.0,	1.0 …	1
0.0,	1.0,	1.0,	1.0,	0.0,	1.0,	0.0,	1.0 …	0
0.0,	1.0,	1.0,	1.0,	0.0,	1.0,	0.0,	0.0 …	0
0.0,	0.0,	0.0,	0.0,	1.0,	0.0,	0.0,	1.0 …	1
0.0,	0.0,	1.0,	1.0,	0.0,	1.0,	0.0,	0.0 …	1
				…				
1.0,	1.0,	1.0,	1.0,	0.0,	1.0,	1.0,	1.0 …	0

Testing:

1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0 … ?

Words as Features

Words in a document are a good clue of its contents:

money -> spam?

teach -> non-spam?

Each dimension of feature vector records presence of some word in input

money x_1	teach x_2	earn x_3	course x_4	win x_5	summer x_6	think x_7	aardvark x_8 …	y
1.0,	0.0,	1.0,	1.0,	0.0,	0.0,	1.0,	1.0 …	1
1.0,	0.0,	0.0,	1.0,	1.0,	0.0,	0.0,	1.0 …	1
0.0,	1.0,	1.0,	1.0,	0.0,	1.0,	0.0,	1.0 …	0
0.0,	1.0,	1.0,	1.0,	0.0,	1.0,	0.0,	0.0 …	0
0.0,	0.0,	0.0,	0.0,	1.0,	0.0,	0.0,	1.0 …	1
0.0,	0.0,	1.0,	1.0,	0.0,	1.0,	0.0,	0.0 …	1
				…				
1.0,	1.0,	1.0,	1.0,	0.0,	1.0,	1.0,	1.0 …	0

Other Commonly Used Features

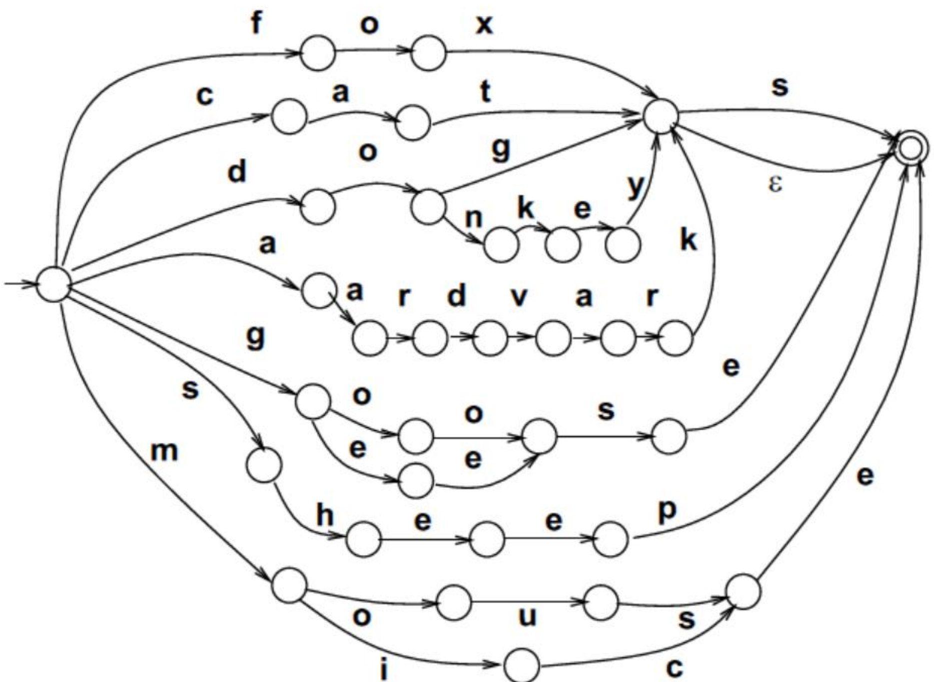
1. **Lemma –** remove affixes and recover lemma (the form you'd look up in a dictionary)
   - *foxes -> fox*
   - *flies -> fly*
   - *geese -> goose*
2. **Stemming**

   Cut affixes off to find the stem
   - *airliner -> airlin*

How is Lemmatization Done?

Finite state automata

- See COMP/LING-445



J&M

Stemming – Porter Stemmer

An ordered list of rewrite rules to approximately recover the stem of a word (Porter, 1980)

- Basic idea: chop stuff off and glue some endings back on
- Not perfect, but sometimes results in a slight improvement in downstream tasks

Examples of Porter Stemmer Rules

ies -> i

- *ponies -> poni*

ational -> ate

- *relational -> relate*

If word is long enough (# of syllables, roughly speaking),

al -> ε

- *revival -> reviv*

N-grams

- Sequences of adjacent words
- E.g. unigrams (N=1) are just words in isolation
- Called “bag-of-words” (if unigrams), or “bag-of-n-grams”

Versions:

- Presence or absence of an N-gram (1 or 0)
- Count of N-gram
- Proportion of the total document
- Scaled versions of the counts (e.g., discount common words like *the*, and give higher weight to uncommon words like *penguin*)

POS Tags

Sequences of POS tags are also popular as features – crudely captures syntactic patterns in text

- Very useful for authorship attribution, for example

Need to **preprocess** the documents for their POS tags

Most common tag set in English:

https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Exercise

Extract unigram and bigram (N-gram for N=2) features for the following, and turn them into a feature vector form by recording frequency **with lemmatization**.

Good day! Do you need a personal loan without any upfront charges/fees? Kindly apply by visiting my application.

- What other issues do you have to deal with/decisions do you have to make?
- Which words or bigrams do you think might be most useful in deciding whether this is spam? Least useful?

Removing Stop Words

Common words may not be so useful for some document classification tasks

However, this is highly task-dependent

Standardized lists of such **stop words** are commonly removed

e.g., partial list from NLTK:

‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’, ‘until’, ‘below’, ‘are’, ‘we’, ‘these’, ‘your’, ‘his’, ‘through’, ‘don’, ‘nor’, ‘me’, ‘were’, …

Steps in Building a Text Classifier

1. Define problem and collect data set
2. Extract features from documents
3. **Train a classifier on a training set**
4. Apply classifier on test data

Classification Models

$$y = f(\vec{x})$$

document label

classifier

input document

What form should f take? Popular choices:

- Naïve Bayes
- Support vector machines
- Logistic regression
- Artificial neural networks (multilayer perceptrons)

We'll start discussing these next class!

The Model Selection Problem

Many possible model variations for classification

Preprocessing decisions:

- Feature extraction and selection scheme
- Feature representation scheme

Classifier selection:

- Naïve Bayes, SVM, logistic regression, neural networks…

How do we know which one is the best?

Training and Testing Data

We need to evaluate it on *unseen* data that the model has not been exposed to.

- We are testing the model's ability to *generalize*.

Given a corpus, how is the data usually split?

Training data: often 60-90% of the available data

Testing data: often 10-20% of the available data

There is often also a **development** or **validation** data set, for deciding between different versions of a model.

Model Selection

Procedure:

1. Train one or more models on the training set
2. Test (repeatedly, if necessary) on the dev/val set; choose a final setting
3. Test the final model on the final testing set (once only)

Another Strategy: Cross Validation

k-fold cross validation: splitting training data into k partitions or folds; iteratively test on each after training on the rest

e.g., 3-fold CV: split dataset into 3 folds

	Fold 1	Fold 2	Fold 3
Exp. 1	test	train	train
Exp. 2	train	test	train
Exp. 3	train	train	test

Average results from above experiments

- CV is often used if the corpus is small

Supervised Classifiers in Python

scikit-learn has many simple classifiers implemented, with a common interface.

e.g., SVMs

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
>>> clf.predict([[2., 2.]])
```

Steps in Building a Text Classifier

1. Define problem and collect data set
2. Extract features from documents
3. Train a classifier on a training set
4. **Apply classifier on test data**

Key Issues in Evaluation

What evaluation measure to use?

- Accuracy, precision, recall, F1

How do we tell if a model is really better?

- Statistical significance tests

Does this actually matter?

More on this in future lectures

Getting Rich…?

So should you trade stocks by building a sentiment analysis system?

Let's discuss how we might build a text classification system for this problem:

1. Define problem and collect data set
2. Extract features from documents
3. Train a classifier on a training set
4. Apply classifier on test data

Potential obstacles to profiting from this system?