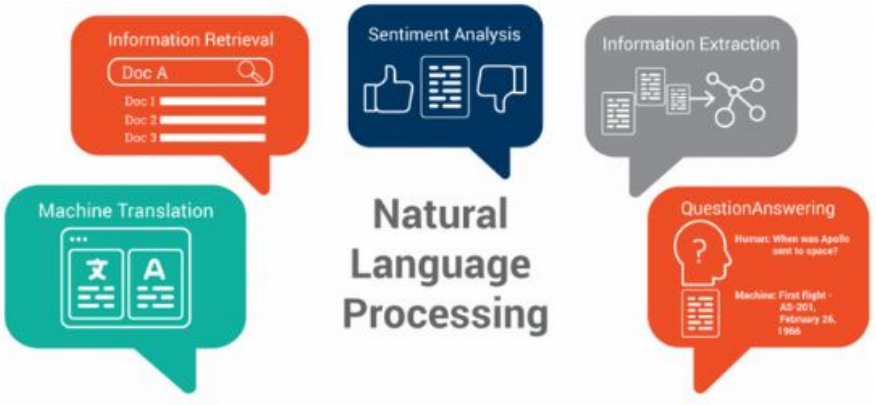


Solving Business Problems with NLP

Instructor: Juber Rahman

Omdena

Omdena School

Course Structure

- Natural Language Processing + Problem Solving
- Lectures (25%)
- Hands-on-Coding (50%)
- Reporting (25%)
- 20-30 minutes of lecture per week
- 40-60 minutes of coding session/ guided lab
- unguided project

Course Highlights

Project 1- Market Segmentation (supervised classification)

Project 2- Topic Classification (unsupervised classification)

Project 3- Churn Prediction (Deep learning)

Project 4- Named Entity Recognition (Semi-supervised)

Project 5- Competitive Intelligence (recommendation system)

Project 6- Capstone Project (Chatbot, CI model, etc.)

Week 1 Topics

- Common business problems
- Basics of NLP
- Scraping text data
- Pre-processing of text data
- Tokenization of Text
- Training ML models

Common Business Problems

- Buyer Propensity Estimation
- Customer Churn Prediction
- Market-Basket analysis/ Segmentation
- Dynamic goal allocation
- Customer Lifetime value prediction
- Customer acquisition
- Customer retention

Market Segmentation

What/ how to:

- Group customers in different cohort / basket
- Put similar customers in each basket
- Establish a hierarchy for the segments
- Put more important customer in higher segment

Why:

- Service level
- Pricing
- Support
- Business strategy

NLP Workflow

Synnex Corporation is an American multinational corporation that provides B2B IT services. It was founded in 1980 by Robert T. Huang and based in Fremont", ' California. As an information technology supply chain services company', ' it ' software publishers and reseller customers.[3] Originally founded as a technology hardware distributor4] …

Pre-processing
Stemming
Lemmatization

'synnex', 'corpor', 'american', 'multin', 'corpor', 'provid', 'servic', 'found', 'robert', 'huang', 'base', 'fremont', 'california', 'inform', 'technolog', 'suppli', 'chain', 'servic', 'compani', 'offer', 'servic', 'origin', 'equip', 'manufactur', 'softwar', 'publish', 'resel', 'custom', 'origin', 'found', 'technolog', 'hardwar', 'distributor'

Tokenization

Corpus vocabulary size?

Embeddings/
Vectorization

Shuffle,
Stratify
Train/Test
split (80:20
ratio)

Category 1

Category 2

labels

ML
Classifiers
(RF/GBM/
Stacking of
Ensembles)

Trained model

Word Embeddings

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

- Bag of words
- Term Frequency - Inverse Document Frequency
- Word to Vector

TF for the word this = (number of times ‘this’ appears in review 2)/(number of terms in review 2)

IDF(‘this’) = log(number of documents/number of documents containing the word ‘this’)

Popular NLP Libraries

- NLTK- text processing libraries for sentence detection, tokenization, lemmatization, stemming
- Spacy- spaCy can preprocess text for Deep Learning
- Gensim - topic modeling, document indexing, and similarity retrieval with large corpora
- TextBlob- sentiment analysis, classification, language translation, word inflection, parsing, n-grams, and WordNet integration
- CoRex- semi supervised classification

Supervised Classification

- There are three methods to establish a classifier

  a) Model a classification rule directly

  Examples: k-NN, decision trees, perceptron, SVM

  b) Model the probability of class memberships given input data

  Example: multi-layered perceptron with the cross-entropy cost

  c) Make a probabilistic model of data within each class

  Examples: naive Bayes, model based classifiers

- *a*) and *b*) are examples of discriminative classification
- *c*) is an example of generative classification
- *b*) and *c*) are both examples of probabilistic classification

Things We'd Like to Do

- Spam Classification
  - Given an email, predict whether it is spam or not
- Medical Diagnosis
  - Given a list of symptoms, predict whether a patient has disease X or not
- Weather
  - Based on temperature, humidity, etc… predict if it will rain tomorrow

Bayesian Classification

- Problem statement:
  - Given features $X_1, X_2, \ldots, X_n$
  - Predict a label Y

The Bayes Classifier

- Use Bayes Rule!

Likelihood

Prior

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Normalization Constant

- Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

The Bayes Classifier

- Let's expand this for our digit recognition task:

$$P(Y = 5|X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n|Y = 5)P(Y = 5)}{P(X_1, \ldots, X_n|Y = 5)P(Y = 5) + P(X_1, \ldots, X_n|Y = 6)P(Y = 6)}$$

$$P(Y = 6|X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n|Y = 6)P(Y = 6)}{P(X_1, \ldots, X_n|Y = 5)P(Y = 5) + P(X_1, \ldots, X_n|Y = 6)P(Y = 6)}$$

- To classify, we'll simply compute these two probabilities and predict based on which one is greater

Model Parameters

- For the Bayes classifier, we need to “learn” two functions, the likelihood and the prior

- How many parameters are required to specify the prior for our digit recognition example?

Model Parameters

- How many parameters are required to specify the likelihood?
  - (Supposing that each image is 30x30 pixels)

?

Model Parameters

- The problem with explicitly modeling $P(X_1,\ldots,X_n|Y)$ is that there are usually way too many parameters:
  - We'll run out of space
  - We'll run out of time
  - And we'll need tons of training data (which is usually not available)

The Naïve Bayes Model

- The *Naïve Bayes Assumption*: Assume that all features are independent **given the class label Y**
- Equationally speaking:

$$P(X_1, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

- (We will discuss the validity of this assumption later)

Why is this useful?

- # of parameters for modeling $P(X_1,\ldots,X_n|Y)$:
  - $2(2^n-1)$
- # of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$
  - $2n$

Naïve Bayes Training

- Training in Naïve Bayes is **easy**:
  - Estimate P(Y=v) as the fraction of records with Y=v

$$P(Y = v) = \frac{Count(Y = v)}{\#\ records}$$

$$P(X_i = u | Y = v) = \frac{Count(X_i = u \wedge Y = v)}{Count(Y = v)}$$

  - Estimate $P(X_i=u|Y=v)$ as the fraction of records with Y=v for which $X_i=u$

(This corresponds to Maximum Likelihood estimation of model parameters)

Example of the Naïve Bayes Classifier

The weather data, with counts and probabilities

outlook			temperature			humidity			windy		play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

A new day

outlook	temperature	humidity	windy	play
sunny	cool	high	true	?

Prediction for new data

- Likelihood of yes

$$= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

- Likelihood of no

$$= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

- Therefore, the prediction is No

Evaluating classification algorithms

- You have designed a new classifier.

- You give it to me, and I try it on my image dataset

Evaluating classification algorithms

- I tell you that it achieved 95% accuracy on my data.

- Is your technique a success?

Types of errors

- But suppose that
  - The 95% is the correctly classified pixels
  - Only 5% of the pixels are actually edges
  - It misses all the edge pixels

- How do we count the effect of different types of error?

Types of errors

Ground Truth	Prediction: Edge	Prediction: Not edge
Edge	True Positive	False Negative
Not Edge	False Positive	True Negative

Two parts to each: whether you got it correct or not, and what you guessed. For example for a particular pixel, our guess might be labelled…

True Positive

Did we get it correct?
True, we did get it correct.

What did we say?
We said ‘positive’, i.e. edge.

or maybe it was labelled as one of the others, maybe…

False Negative

Did we get it correct?
False, we did not get it correct.

What did we say?
We said ‘negative, i.e. not edge.

Sensitivity and Specificity

Count up the total number of each label (TP, FP, TN, FN) over a large dataset. In ROC analysis, we use two statistics:

Sensitivity = $\frac{TP}{TP+FN}$

Can be thought of as the likelihood of spotting a positive case when presented with one.

Or… the proportion of edges we find.

Specificity = $\frac{TN}{TN+FP}$

Can be thought of as the likelihood of spotting a negative case when presented with one.

Or… the proportion of non-edges that we find.

$$\text{Sensitivity} = \frac{TP}{TP+FN} = ? \qquad \text{Specificity} = \frac{TN}{TN+FP} = ?$$

Ground Truth	Prediction: 1	Prediction: 0
1	60	30
0	80	20

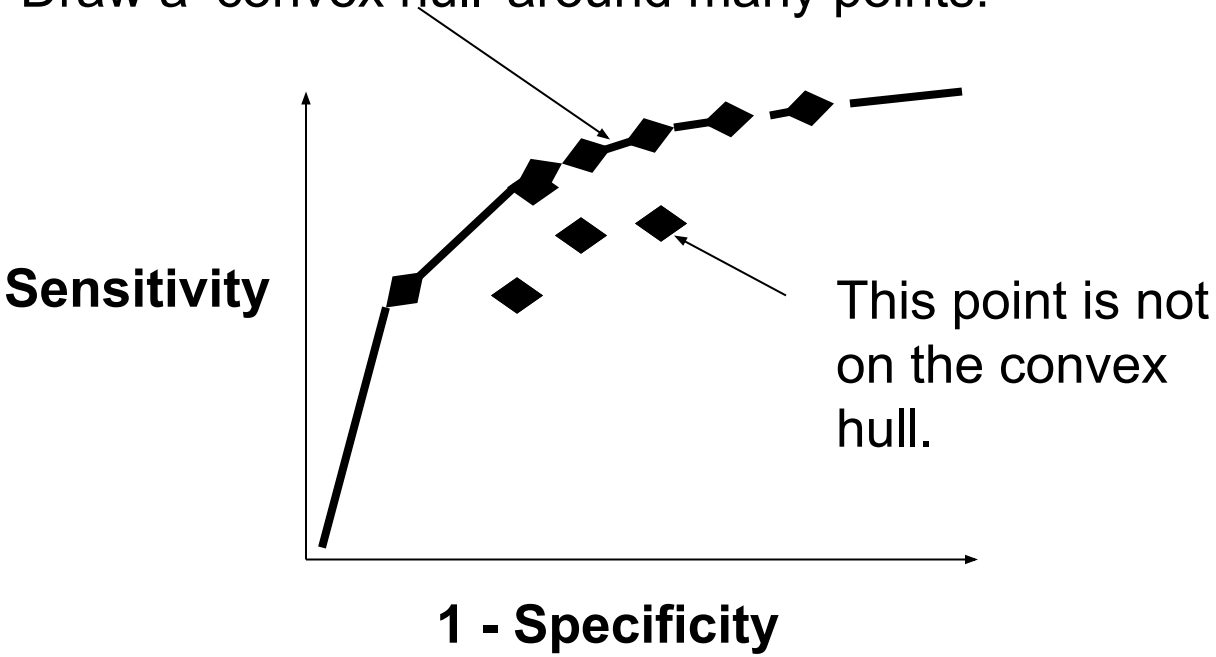
60+30 = 90 cases in the dataset were class 1 (edge)

80+20 = 100 cases in the dataset were class 0 (non-edge)

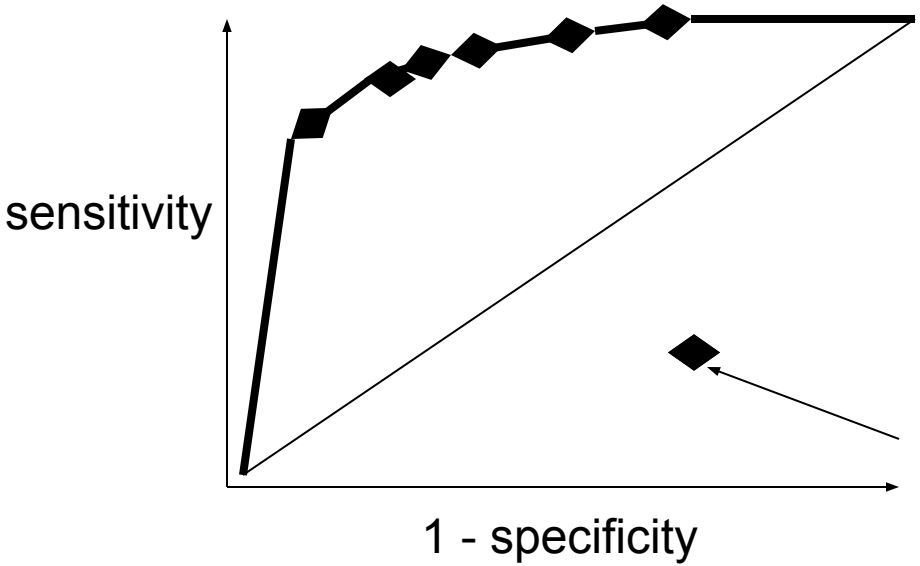
90+100 = 190 examples (pixels) in the data overall

The ROC Curve

Draw a ‘convex hull’ around many points:



ROC Analysis



All the optimal detectors lie on the convex hull.

Which of these is best depends on the ratio of edges to non-edges, and the different cost of misclassification

Take-home point : You should always quote sensitivity and specificity for your algorithm, if possible plotting an ROC graph.

Holdout estimation

- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training

Holdout estimation

- Problem: the samples might not be representative
  - Example: class might be missing in the test data

- Advanced version uses *stratification*
  - Ensures that each class is represented with approximately equal proportions in both subsets

Repeated holdout method

- Repeat process with different subsamples
- more reliable

  - In each iteration, a certain proportion is randomly selected for training (possibly with stratificiation)

  - The error rates on the different iterations are averaged to yield an overall error rate

Repeated holdout method

- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?
  - Of course!

Cross-validation

- *Cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training

- Called *k-fold cross-validation*

Cross-validation

- Often the subsets are stratified before the cross-validation is performed

- The error estimates are averaged to yield an overall error estimate

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten?
  - Empirical evidence supports this as a good choice to get an accurate estimate
  - There is also some theoretical evidence for this
- Stratification reduces the estimate variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Leave-One-Out cross-validation

- Leave-One-Out:
  a particular form of cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Makes best use of the data
- Involves no random subsampling
- Very computationally expensive
  - (exception: NN)

Leave-One-Out-CV and stratification

- Disadvantage of Leave-One-Out-CV: stratification is not possible
  - It *guarantees* a non-stratified sample because there is only one instance in the test set!
- A stratified random sampling involves dividing the entire population into homogeneous groups called strata (plural for stratum).

Lab-1

Classify Fulfillment customers from other customers.

- Preprocessing
- Stemming
- Lemmatization
- Embeddings
- Classification