U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

M.EIC

Natural Language Processing

Henrique Lopes Cardoso

FEUP / LIACC

hlc@fe.up.pt

Text Classification

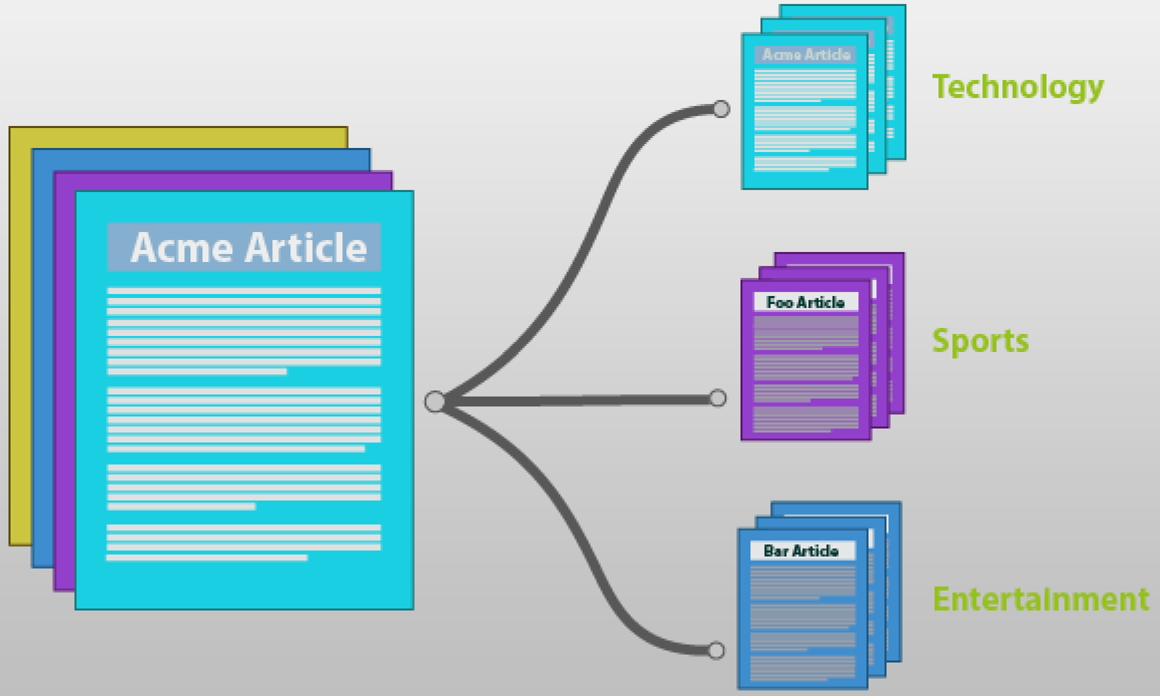
bag-of-words, Naïve Bayes, features, evaluation

Text Classification Tasks

- Given a text, classify it according to a number of classes
  - Spam detection in emails: spam/not spam
  - Sentiment analysis in product reviews: positive/negative, –/0/+, – –/–/0/+/++
  - Assign subject categories, topics, or genres
  - Authorship identification from a closed list
  - Age/gender identification
  - Language detection

Text Classification

- Input:
  - A document d
  - A fixed set of classes $C = \{c_1, c_2, \ldots, c_m\}$
- Output:
  - The predicted class $c \in C$ for document d



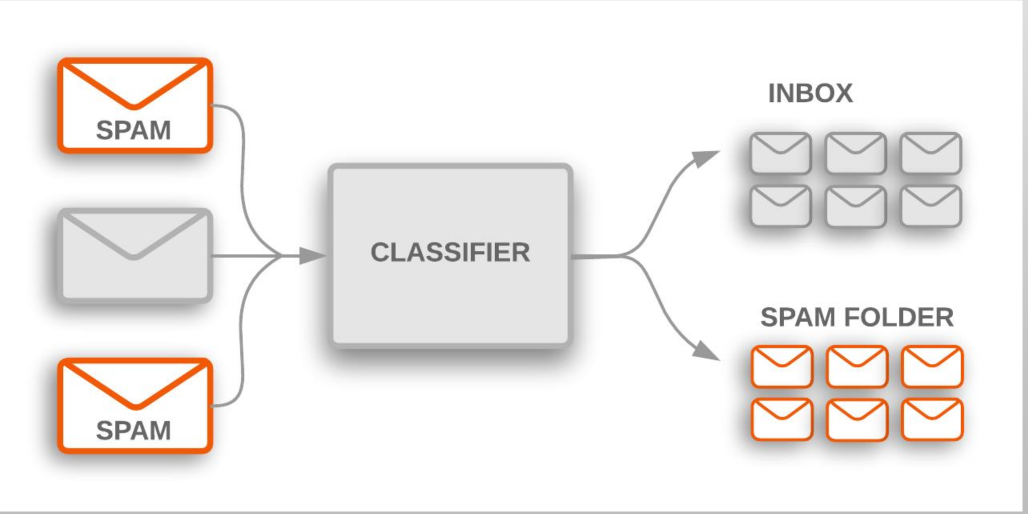
Hand-coded Rules

- Rules based on combinations of words or other features
  - Spam detection: black-list of addresses and keyword detection
  - Sentiment analysis: ratio of word polarities appearing in a sentiment lexicon

- Accuracy can be high…
  - If rules are carefully refined by an expert
- …but building and maintaining these rules is expensive

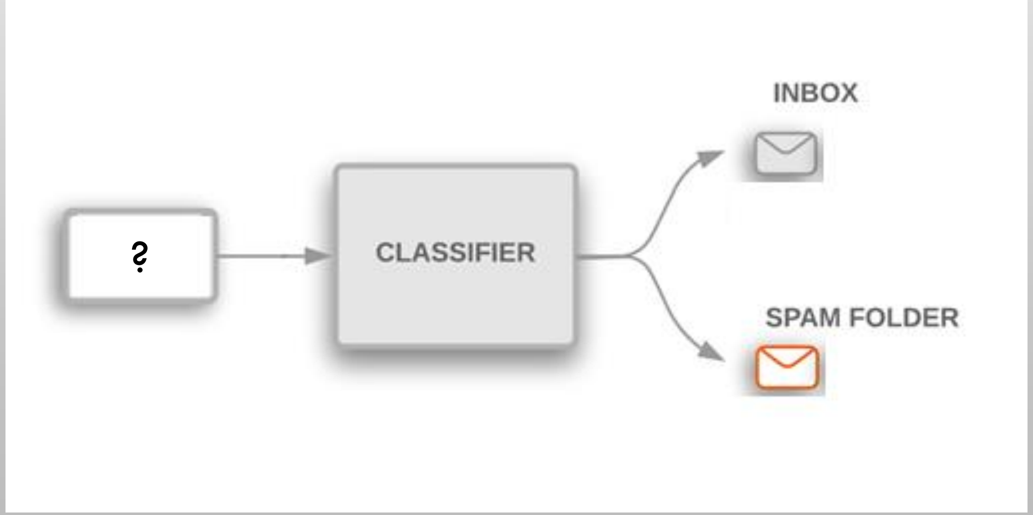
Supervised Machine Learning

- Making use of annotated datasets through Machine Learning algorithms
- Building a model
  - Input:
    - A fixed set of classes $C = \{c_1, c_2, \ldots, c_m\}$
    - A training set of m hand-labeled documents $\{(d_1,c_1), (d_2,c_2), \ldots, (d_n,c_n)\}$, where $d_i \in D$ and $c_i \in C$
  - Output: a classifier $\gamma : D \rightarrow C$
    - a mapping from documents to classes (or class probabilities)



Supervised Machine Learning

- Making use of annotated datasets through Machine Learning algorithms



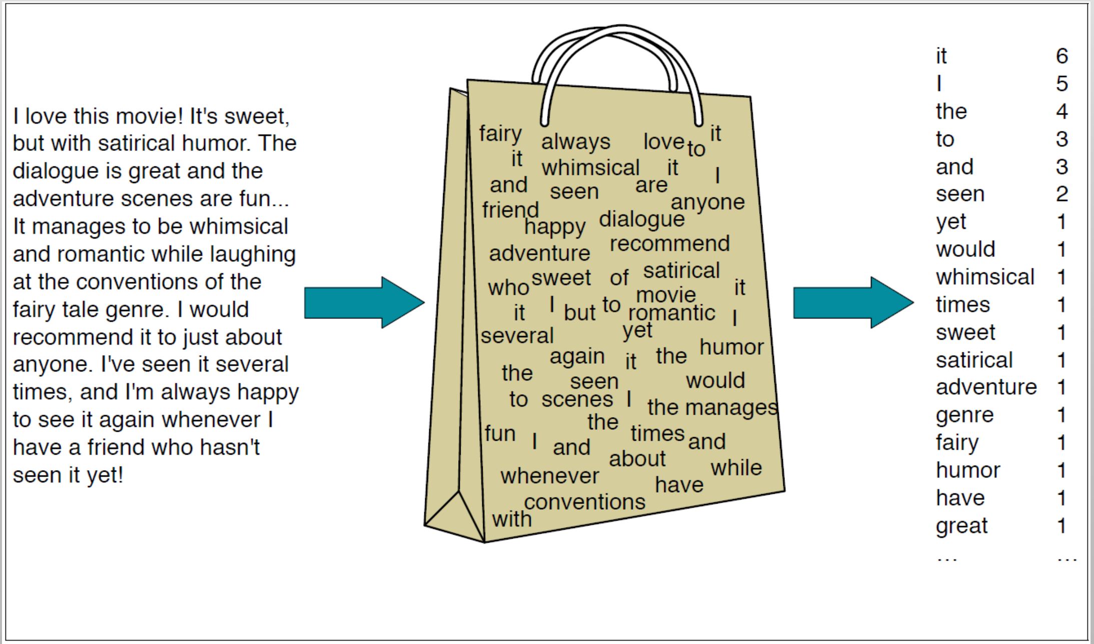
- Classifying a document
  - Input
    - a document d
    - a classifier $\gamma : D \rightarrow C$
  - Output: predicted class $c \in C$ for document d

Classifiers

- Classifier: given a document, assign a class to it
- Probabilistic classifier: more than predicting a class, outputs the probability of the observed document belonging to each of the classes
- Generative vs Discriminative classifiers
  - Generative classifiers build a model of how a class could generate some input data
    - Given an observation, return the class that has most likely produced the observation
    - Example: Naïve Bayes
  - Discriminative classifiers learn what features from the input are most useful to discriminate between the different possible classes
    - Examples: Decision Trees, Logistic Regression, Support Vector Machines

Representing a Document with a Bag of Words

- Machine Learning methods require that the data is represented as a set of features
- We thus need a way of going from a document *d* to a vector of features *X*
- The bag-of-words model
  - an unordered set of words, keeping only their frequency in the document
  - assume position does not matter



Naïve Bayes

- Naïve Bayes (NB) makes a simplifying (naïve) assumption about how the features interact
- Bayes rule:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

- Most likely class:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} = \underset{c \in C}{\operatorname{argmax}}\, P(d|c)P(c)$$

Naïve Bayes Classifier

- Representing a document with features:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; \overbrace{P(d|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; \overbrace{P(f_1, f_2, ..., f_n|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

- Assuming conditional independence:

$$P(f_1, f_2, ..., f_n|c) \; = \; P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

- Naïve Bayes classifier:

$$c_{NB} \; = \; \underset{c \in C}{\operatorname{argmax}} \, P(c) \prod_{f \in F} P(f|c)$$

Applying Naïve Bayes

- Applying NB to the text:

$$\begin{aligned} \text{positions} &\leftarrow \text{all word positions in test document} \\ c_{NB} &= \underset{c \in C}{\operatorname{argmax}}\, P(c) \prod_{i \in positions} P(w_i|c) \end{aligned}$$

- Going to log space:
  - avoid underflow and increase speed

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

  - highest log probability class is still most probable

Naïve Bayes: Computing Probabilities

- Class priors: $$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

- Word probabilities per class:

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)}$$

- Handling non-occurring words in a class
  - Add-one (Laplace) smoothing:

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Naïve Bayes Example

- A sentiment analysis (or polarity) task:

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- Prior distributions:

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

- Word probabilities per class:

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$
$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$
$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

  - “with” doesn’t occur in training set: ignore it

- Class probabilities:

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$
$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

  - Chosen class: **negative (-)**

Naïve Bayes is Not So Naïve

- Very fast, low storage requirements
- Robust to irrelevant features: they tend to cancel each other without affecting results
- Very good in domains with many equally important features
  - Decision Trees suffer from fragmentation in such cases – especially if little data
- Optimal if the assumed independence assumptions hold

- A good dependable baseline for text classification

Word Occurrence vs Word Frequency

- Binary NB: in how many documents of the class does the word occur?
  - Word occurrence can be more important than word frequency

remove duplicates

Four original documents:

– it was pathetic the worst part was the boxing scenes
– no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film

After per-document binarization:

– it was pathetic the worst part boxing scenes
– no plot twists or great scenes
+ and satire great plot twists
+ great scenes film

	NB Counts		Binary Counts	
	+	–	+	–
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Dealing with Negation

- *I really like this movie* (positive)
- *I didn't like this movie* (negative)

- Prepending `NOT_` to words affected by negation tokens (n't, not, no, never, …)
  - `I did n't like this movie , but I`
  - `I did n't NOT_like NOT_this NOT_movie , but I`

- Using bigrams instead of single words
  - Sequences of two words: instead of "not" and "recommend", "not recommend"

Making use of Lexicons

- Lexicons provide external knowledge that can be very useful for the task!
- Sentiment lexicons
  - Lists of words that are pre-annotated with positive or negative polarity
  - Example: VADER sentiment lexicon
    - 7520 "words", positive or negative biased, including intensity
    - + : magnificently (3.4), beautiful (2.9), admirable (2.6), confident (2.2), :-) (1.3), defensive (0.1)
    - – : amortize (-0.1), bias (-0.4), :-((-1.5), harsh (-1.9), bad (-2.5), catastrophe (-3.4), rapist (-3.9)
- Features based on the occurrence of (positive or negative) sentiment-biased words
  - Useful when training data is sparse or vocabulary usage in test and training sets do not match
  - Dense lexicon features may generalize better than sparse individual-word features

Building other Features

- Predefine likely sets of words or phrases
  - Spam detection: “viagra”, “password will expire”, “Your mailbox has exceeded the storage limit”, “millions of dollars”, “click here”, “urgent reply”, …
- Paralinguistic and extra-linguistic features
  - Words in capital letters
  - HTML with low ratio of text-to-image, sender email address, …
- N-grams (character or word level)
  - Sequences of two (bigrams), three (trigrams) or even more words or characters
  - Can help alleviate the conditional independence assumption of NB
  - But typically generates a very sparse feature space (many bigrams will rarely occur)

Naïve Bayes as a Language Model

- Use all occurring words as features: a set of class-specific unigram language models
- The likelihood features from NB assign a probability to each word: $P(word|c)$
- Probability of a sentence:

$$P(s|c) = \prod_{i \in positions} P(w_i|c)$$

- Example:

w	P(w|+)	P(w|-)
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...	...	...

$$P(\text{"I love this fun film"}|+) = 0.1 \times 0.1 \times 0.01 \times 0.05 \times 0.1 = 0.0000005$$
$$P(\text{"I love this fun film"}|-) = 0.2 \times 0.001 \times 0.01 \times 0.005 \times 0.1 = .0000000010$$

Evaluation: Confusion Matrix

		Gold standard	
		positive	Negative
System output	positive	true positive	false positive
	negative	false negative	true negative

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

- Accuracy doesn't work well when the classes are unbalanced
- Precision: percentage of items predicted as positive that are if fact positive
- Recall: percentage of positive items that the system has actually predicted as positive
- Precision and Recall emphasize true positives – the things we are supposed to be looking for!

Evaluation: F-measure

- Combining precision and recall: F-measure
  - $F_\beta = \frac{(\beta^2+1) \times precision \times recall}{\beta^2 \times precision + recall}$
    - harmonic mean of precision and recall
    - β: how much more important recall is than precision?
      - E.g., for $\beta = 2$, recall is twice as important as precision
    - $\beta > 1$ favors recall, $\beta < 1$ favors precision
- F1-score
  - $F_1 = \frac{2 \times precision \times recall}{precision + recall}$

More than Two Classes

- Sentiment analysis: **–/0/+** or **– –/–/0/+/++**
- Topic classification: {culture, economy, local, politics, sci-tech, society, sports, world}$^{+}$ (1 or more)
- Multi-class classification tasks
  - Multi-label (any-of) classification: each item can be assigned more than one label
  - Multinomial (one-of) classification: classes are mutually exclusive
- Multi-class model
  - Any-of: one binary classifier for each class
    - each classifier makes its decision **independently** (multiple labels may be assigned)
  - One-of:
    - multi-class classifier (e.g., Multinomial Naïve Bayes, Multinomial Logistic Regression, …)
    - binary classifiers: run all classifiers and choose the label from the classifier with the **highest score/probability**

Macro vs Micro Averaging

- Macro-averaging
  - Compute performance for each class and then average over classes
- Micro-averaging
  - Collect decisions from all classes and compute performance

		gold labels urgent	normal	spam	
system output	urgent	8	10	1	$\textbf{precision}_u = \frac{8}{8+10+1}$
	normal	5	60	50	$\textbf{precision}_n = \frac{60}{5+60+50}$
	spam	3	30	200	$\textbf{precision}_s = \frac{200}{3+30+200}$
		$\textbf{recall}_u = \frac{8}{8+5+3}$	$\textbf{recall}_n = \frac{60}{10+60+30}$	$\textbf{recall}_s = \frac{200}{1+50+200}$	

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60$$

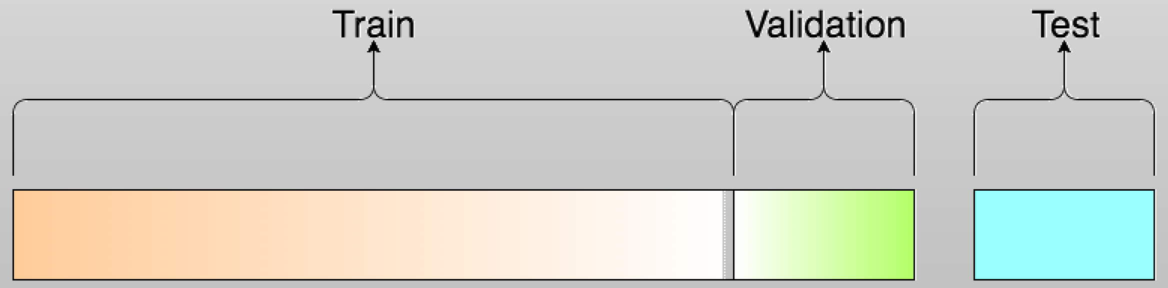
Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = .73$$

Training, Development and Test Sets

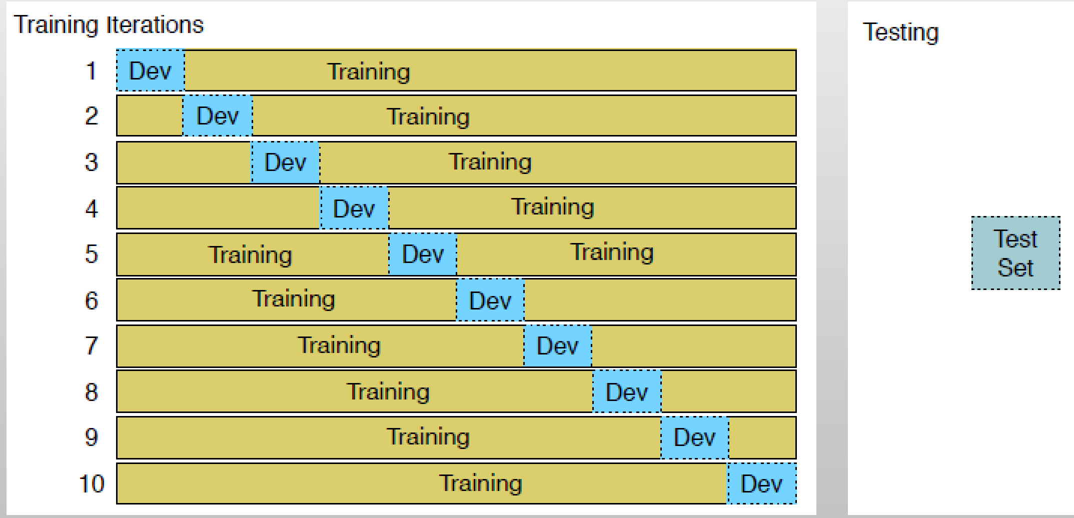
- Training set: used to train the model
- Development (or validation) set: used to tune hyperparameters and decide what the best model is
- Test set: used to test the model's performance



- Problem: is the test set large enough to be representative?

Cross-validation

- Why not use all data both for training and testing?
- K-fold cross-validation
  - Split data into *k* folds
  - Repeat *k* times: train with *k-1* folds and test with the remaining fold
  - Report average scores on the *k* iterations
- If possible, keep a test set so that the model can be optimized in the training/dev sets



Bias in Text Classification Tasks

- Representational harms
  - Caused by a system that demeans a social group, e.g., by perpetuating negative stereotypes
  - Example: sentiment classifiers assign lower sentiment and more negative emotion to sentences with African American names
- Causes:
  - Bias in the training data: machine learning systems typically replicate or amplify bias
  - Bias in the human labelers
  - Bias in the resources used (lexicons, pretrained embeddings, …)
  - Model architecture: what is the model trying to optimize?

➢ Mitigation of these harms is an open research area

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

LIACC

The Python Notebook

jupyter

text-classification_.ipynb

- Loading a dataset for text classification
- Cleanup and normalization
- Generating features: BoW, 1-hot vectors, TF-IDF, n-grams
- Using ML classifiers: NB, LR, DT, RF, SVM, …

NLTK
https://www.nltk.org/

scikit learn
https://scikit-learn.org/