Introduction to Computational Linguistics
Session 5: Text Classification

Denise Löfflad

Universität Tübingen

November 29, 2023

Plan

- How computers learn
- Supervised and unsupervised machine learning
- Evaluating your model
- NLP segmentation

We are **not** going to talk about Naives Bayes, the Perceptron and other machine learning algorithms as this is for future semesters!

How Computers Learn

- machine learning: machine induces patterns from observations in data
- model:
    - summary of what the algorithm learned
    - estimates based on seen data, but can be applied to unseen data

How Computers Learn

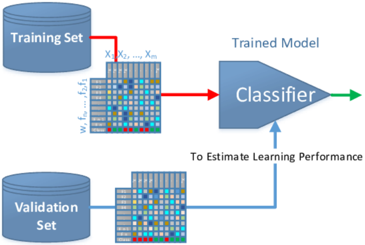
Features

- Dimensions of interest in data
- Quantifications of raw data
- Measured for all records in the data set
- Individual features often combined into a *feature vector*
- Example: Speech Intelligibility scoring
  - Features:
    - rate of speech (syllables per second)
    - total time speaking (seconds)
    - amplitude (decibels)
    - pronunciation nativeness (GOP scores, Link)
    - number of grammatical errors (error count)
  - Feature Vector: $[1.7, 4, 55, -28, 2]$
- Learning algorithms accept feature vectors as input

How Computers Learn

data often split into different sets

- training set
- test set
- development/validation set
  or cross-validation



Source `https://www.researchgate.net/figure/`

`The-learning-process-of-a-classifier-The-classifier-learns-from-the-training-data-The_fig1_333505093`

Supervised Learning

- work with labelled data
- "supervised": labels are determined by humans in the loop
  - e.g. annotations from corpora
- examples:
  - label how intelligible a recording is (good, fair, poor)

Supervised Learning

workflow:

1. carefully formulate the RQ
2. find or create appropriate data
3. define feature extracting mechanism and clean the data
4. decide on algorithm(s)
5. split the data
6. train the model
7. test and cross-validate
8. evaluate the model's performance
9. compare to baselines of StotA

compare with alternative algorithms/feature sets

Supervised Learning

Examples:

- linear regression
- logistic regression
- support vector machine
- decision tree
- classification tasks

Unsupervised Learning

- no pre-d categories
- purpose: automatically detecting structure in data

Unsupervised Learning

workflow:

1. define feature extracting mechanisms
2. decide on algorithms
3. apply on data set
4. inspect resulting inferred structure

Unsupervised Learning

examples:

- k-means clustering
- hierarchical clustering

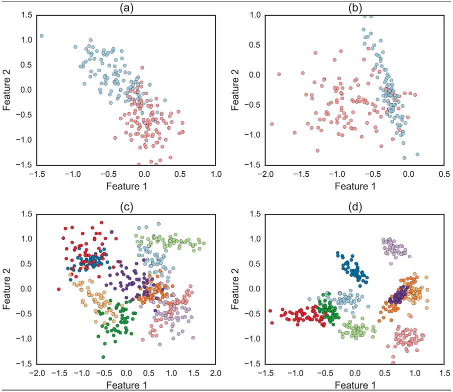


Figure: Taken from ,Rodriguez et al. [2019]

How to evaluate your (supervised) model?

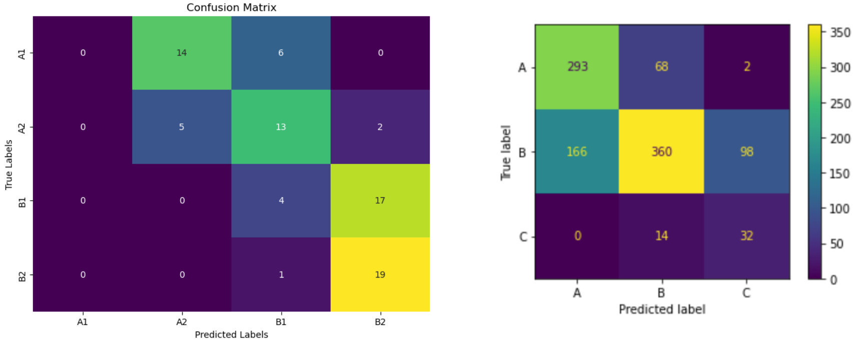


Figure: Confusion matrices on classification data

Measuring Success (binary classification)

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- Recall: $\frac{TP}{TP+FN}$
- Precision: $\frac{TP}{TP+FP}$
- Accuracy: $\frac{TN+TP}{TP+FP+FN+TN}$
- True negative rate (TNR): $\frac{TN}{FP+TN}$

Measuring Success

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- Precision: $\frac{\text{TP}}{\text{TP+FP}}$

Measuring Success

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- Recall: $\frac{\text{TP}}{\text{TP}+\text{FN}}$

Measuring Success - Error Analysis

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- True negative rate (TNR): $\frac{\text{TN}}{\text{FP+TN}}$

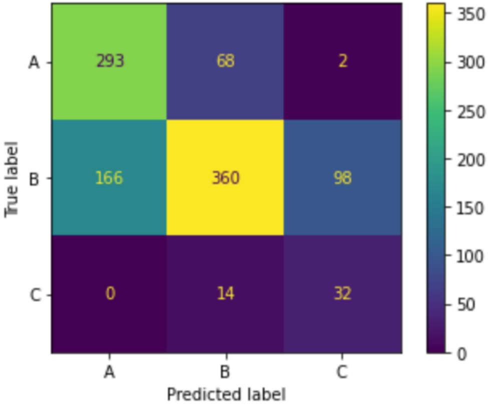
Measuring Success

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- Accuracy: $\frac{\text{TN+TP}}{\text{TP+FP+FN+TN}}$

For multiclass classification you have different options:

- calculate the values for every class
- average (micro or macro) over the classes



before the lecture look at this again https://www.evidentlyai.com/

Measuring Success

Which measures are important?

Measuring Success

Which measures are important?

- Would you rather see spam mails or miss work mails?
- Would you rather see hate speech or get your normal Tweet deleted?

$\rightarrow$ always consider the task you are working on! There is no one-fits-all solution.

NLP Tasks - Segmentation

- raw input: input stream of characters/text
- output: input segmented into smaller units
  - tokens/"words" → tokenization
  - sentence → sentence segmentation
  - chunks: groups of tokens

Tokenization

"A token is an instance of a *sequence of characters* in some particular document that are *grouped together as a useful semantic unit for processing*. A type is the class of all tokens containing the same character sequence. "[1]

[1] `https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html`, last accessed 2021-01-18

Tokenization

- split into smaller units
- not necessarily always a “word”
- many decisions to make to deal with challenges

Tokenization – Challenges

- contracted/enclitic forms
- example: “I don’t know whether he’s going there.
- one/multiple words/tokens ?

Tokenization – Challenges

- hyphenated forms
- example: "An end user device-independent solution - would that be possible?"
- meaning preserved when splitting into two tokens?

Tokenization – Challenges

- forms with/adjacent to periods
- example:
  "You find it on 21st St. opposite of Third Ave., in the U.S.A."

Tokenization – Challenges

- slashes
- <u>example:</u>
  "A helpful/fun e-learning website is https://feedbook.schule"

Tokenization – Challenges

- special characters
  - single/double quotation marks
  - comma/semicolon
  - question mark/exclamation marks
  - round brackets
- example:
  “Yes!” is what I would ’say’, but I don’t say it - I know better (now).

Tokenization – Challenges

- multi-word expressions
- includes compound nouns
- need to decide whether to recognize multiword tokens or not
- example: "hot dog", "in spite of"

Tokenization – Challenges

- named entities
- example: "New York"

Tokenization – Challenges

- abbreviations
- example: "Builders Inc."

Tokenization – Challenges

- integrate morphological analysis to split certain contracted forms?
- usage of lexicon in tokenization?
    - Can help with things like “U.S.A.”, “hot dog”, and so on

Sentence Segmentation

- decide sentence boundaries
- in case of punctuation characters, decide whether they are
  - sentence-internal (e.g., Mr.)
  - sentence-final (e.g., The cat woke up.)
- can be tackled as classification task for punctuation
- many tools perform it in conjunction with tokenization

Further Challenges

- different meaning of characters in different language and writing systems
  - e.g. agglutinative languages vs. ideographic writing systems
- lack of punctuation or spaces in certain writing systems, changes in semantic interpretation depending on tokenization
- under-resourced languages: difficult to find corpora to train models

Getting Started with NLP - The OpenNLP Library

Basic properties of Apache **OpenNLP**:

- library with NLP tools for segmentation, tagging, named entity recognition, chunking, parsing, ...
- **machine learning** algorithms, some rule-based methods for comparison
- includes tools and pre-trained statistical language models
- accessible both as an **API** (**Application Programming Interface**) and as an executable program (**CLI** or **Command Line Interface**)

Downloads

- official website: `https://opennlp.apache.org`
- OpenNLP Package by Chen Xiaobin:

  https://github.com/chxiaobin/opennlp/archive/master.zip
- manuals at `https://opennlp.apache.org/docs/`

OpenNLP: Sentence Segmentation

Deciding whether a punctuation character marks the end of a sentence.

- `$ bin/opennlp SentenceDetector models/en-sent.bin < input/text1.txt`

Tokenization

Tokens: words, punctuation, numbers...

- tokenizer with statistical model
- `$ bin/opennlp TokenizerME models/en-token.bin < input/text1.txt`

Rule-Based vs. Statistical Models

Exercise

The OpenNLP library also includes a simple tokenizer which does not require a model for tokenization. Try to run it with the same input file and see if it gives you the result you want.

```
$ bin/opennlp SimpleTokenizer < input/text1.txt
```

References and Acknowledgments

These slides are largely based on Dickinson et al. [2012] and largely based on slides for Text Technology, 2023 by Stephen Bodnar and on Detmar Meurers' slides on Second Language Acquisition.

Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and computers*. John Wiley & Sons, 2012.

Mayra Z Rodriguez, Cesar H Comin, Dalcimar Casanova, Odemir M Bruno, Diego R Amancio, Luciano da F Costa, and Francisco A Rodrigues. Clustering algorithms: A comparative approach. *PloS one*, 14(1):e0210236, 2019.