



THE UNIVERSITY
of EDINBURGH

Text Technologies for Data Science

INFR11145

Text Classification

Instructor:
Björn Ross


20-Nov-2024

1

Pre-Lecture

- Last week
 - Online guest lecture. Recording on Learn!
- Today
 - Lecture 1 (Text classification): Theory
 - Lecture 2 (Text classification): Practical
- Next week
 - Final set of lectures + CW3 announcement

Björn Ross, TTDS 2024/2025



THE UNIVERSITY
of EDINBURGH

2

Lecture Objectives

- Learn about text basics of text classification
 - Definition
 - Types
 - Methods
 - Evaluation

3

Text Classification

- **Text classification** is the process of classifying documents into predefined categories based on their content.
 - Input: Text (document, article, sentence)
 - Task: Classify into predefined one/multiple categories
 - Categories:
 - Binary: relevant/irrelevant, spam .. etc.
 - Few: sports/politics/comedy/technology
 - Hierarchical: patents

4

Classification is and is not

- **Classification** (a.k.a. “**categorization**”): a common technology in data science; studied within pattern recognition, statistics, and machine learning.
- Definition: the activity of **predicting** to which among a **predefined finite set** of groups (“classes”, or “categories”) a data item belongs to
- Formulated as the task of generating a hypothesis (or “classifier”, or “model”)

$$h : D \rightarrow C$$

where $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ is a domain of data items and
 $C = \{c_1, \dots, c_n\}$ is a finite set of classes (the **classification scheme**)

Classification is and is not

- Different from clustering, where the groups (“clusters”) and their number are not known in advance
- Unsuitable when class membership can be determined with certainty (relatively easily)
 - e.g., predicting whether a natural number belongs to *Prime* or *Non-Prime* is not classification
- In text classification, data items are
 - **Textual**: e.g., news articles, emails, sentences, queries, etc.
 - **Partly textual**: e.g., Web pages

Types of Classification

- **Binary:**
item to be classified into one of two classes
 $h : D \rightarrow C, C = \{c_1, c_2\}$
 - e.g., Spam/not spam, offensive/not offensive, rel/irrel
- **Single-Label Multi-Class (SLMC)**
item to be classified into only one of n possible classes.
 $h : D \rightarrow C, C = \{c_1 \dots c_n\}$, where $n > 2$
 - e.g., Sports/politics/entertainment, positive/negative/neutral
- **Multi-Label Multi-Class (MLMC)**
item to be classified into none, one, two, or more classes
 $h : D \rightarrow 2^C, C = \{c_1 \dots c_n\}$, where $n > 1$
 - e.g., Assigning CS articles to classes in the ACM Classification System
 - Usually be solved as n independent binary classification problems

Björn Ross, TTDS 2024/2025



7

Dimension of Classification

- Text classification may be performed according to several dimensions (“axes”) orthogonal to each other
- by **topic**; by far the most frequent case, its applications are global
- by **sentiment**; useful in market research, online reputation management, social science and political science
- by **language** (a.k.a. “language identification”); useful, e.g., in query processing within search engines
- by **genre**; e.g., AutomotiveNews vs. AutomotiveBlogs, useful in website classification and others;
- by **author** (a.k.a. “authorship attribution”), by native language (“native language identification”), or by gender; useful in forensics and cybersecurity
- by **usefulness**; e.g., product reviews
-

Björn Ross, TTDS 2024/2025



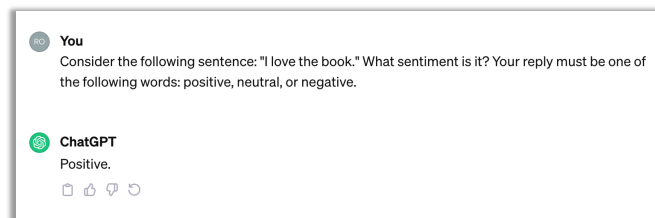
8

Rule-based classification

- An old-fashioned way to build text classifiers was via knowledge engineering, i.e., manually building classification rules
 - E.g., (Viagra or Sildenafil or Cialis) → Spam
 - E.g. (#MAGA or America great again) → support Trump
- Common type: dictionary-based classification
- Disadvantages:
 - Expensive to setup and to maintain
 - Depends on few keywords → bad coverage (recall)

Zero-shot classification

- Using machine learning models for classification without giving training examples
 - A modern approach that can work well and requires little human effort
 - Depends highly on black-box models, limited opportunities for customisation and error analysis (but this is an active research area)
 - Often used as a baseline for performance comparisons



Supervised-learning classification

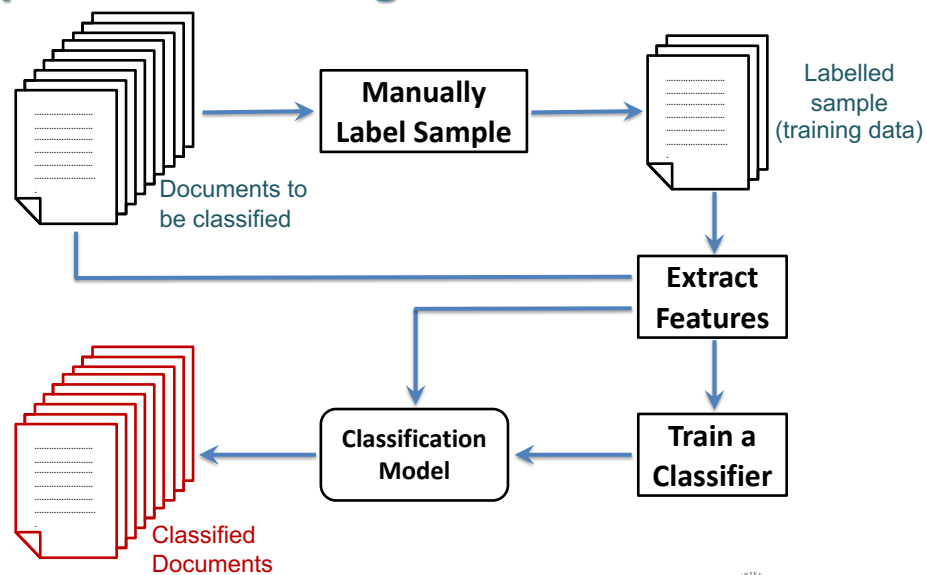
- A generic (task-independent) learning algorithm is used to train a classifier from a set of manually classified examples
- The classifier learns, from these training examples, the characteristics a new text should have in order to be assigned to class c
- Advantages:
 - Generating training examples cheaper than writing classification rules
 - Easy update to changing conditions (e.g., addition of new classes, deletion of existing classes, shifted meaning of existing classes, etc.)

Björn Ross, TTDS 2024/2025



11

Supervised-learning classification



Björn Ross, TTDS 2024/2025



12

Extract Features

- In order to be input to a learning algorithm (or a classifier), all training (or unlabeled) documents are converted into **vectors** in a common **vector space**
- The dimensions of the vector space are called **features**
- In order to generate a vector-based representation for a set of documents D , the following steps need to be taken
 1. Feature Extraction
 2. Feature Selection or Feature Synthesis (optional)
 3. Feature Weighting

Step 1: Feature Extraction

- What are the features that should be different from one class to another?
- Simplest form: Bag-of-words (BOW)
 - Each term in a document is a feature
 - Feature space size = vocabulary in all docs
 - Standard IR preprocessing steps are usually applied
 - Tokenisation, stopping, stemming
- Other simple features forms:
 - Word n-grams (bigrams, trigrams,)
 - Much larger + more sparse
 - Sometimes char n-grams are used
 - Especially for degraded text (OCR or ASR outputs)

Step 1: Feature Extraction

- What other text features could be used?
- Sentence structure:
 - POS (part-of-speech tags)
 - Syntactic tree structure
- Topic-based features:
 - LDA topics
 - NEs (named entities) in text
 - Links / Linked terms
- Non-textual features:
 - Average doc\sentence\word length
 - % of words start with upper-case letter
 - % of links/hashtags/emojis in text

Björn Ross, TTDS 2024/2025



15

Step 1: Feature Extraction

- What preprocessing to apply?
 - Case-folding? **really** vs **Really** vs **REALLY**
 - Punctuation? “?”, “!”, “@”, “#”
 - Stopping? “**he**”, “**she**”, “**what**”, “**but**”
 - Stemming? “**replaced**” vs “**replacement**”
- Other Features:
 - Starts with capital letter, all caps
 - Repeated characters “**congraaaaaats**” “**help!!!!!!!!**”
 - Scores from dictionaries and lexicons (e.g. LIWC)
- Which to choose?
 - Classification task/application

Björn Ross, TTDS 2024/2025



16

Step 2: Feature Selection

- Number of distinctive features = length of feature vector
- Vector can be of length in the order of 10^6 , and might be sparse
 - High computational cost
 - Overfitting
- What are the most important features among those?
 - e.g. Reduce from 10^6 to 10^4
- For each class, find the top representative k features for it → get the Union over all classes → reduced feature space

Björn Ross, TTDS 2024/2025



17

Step 2: Feature Selection Functions

- Document frequency
 - % of docs in class c_i that contain the term t_k
 - Very basic measure. Will select stop words as features
- Mutual Information
 - How much we learn from the presence or absence of term t_k about whether or not a document is in class c_i
 - Often used in feature selection in text classification

$$MI(t_k, c_i) = \sum_{c \in \{c_i, c_i\}} \sum_{t \in \{t_k, t_k\}} P(t, c) \cdot \log_2 \frac{P(t, c)}{P(t) \cdot P(c)}$$

- Pearson's Chi-squared (χ^2)
 - used more in comparisons between classes

Björn Ross, TTDS 2024/2025



18

Step 2: Feature Selection Functions

Function	Denoted by	Mathematical form
Document frequency	$\#(t_k, c_i)$	$P(t_k c_i)$
DIA association factor	$z(t_k, c_i)$	$P(c_i t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{ Tr \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]^2}{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{ Tr \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]}}{\sqrt{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k c_i) + d}{P(\bar{t}_k \bar{c}_i) + d}$
Odds Ratio	$OR(t_k, c_i)$	$\frac{P(t_k c_i) \cdot (1 - P(t_k \bar{c}_i))}{(1 - P(t_k c_i)) \cdot P(t_k \bar{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)$

Björn Ross, TTDS 2024/2025



19

Step 2: Feature Synthesis

- **Matrix decomposition techniques** (e.g., PCA, SVD, LSI) can be used to synthesize new features that replace the features discussed above
- Principle of **distributional semantics**: semantics of a word “is” the words it co-occurs with
 - **Pros**: the synthetic features in the new vector representation do not suffer from problems such as polysemy and synonymy
 - **Cons**: computationally expensive
- **Word embeddings**: the new wave of distributional semantics, modern approaches are based on neural networks
- PCA: Principle component analysis
- SVD: Singular value decomposition
- LSA: latent semantic analysis

Björn Ross, TTDS 2024/2025



20

Step 2: Feature Synthesis

- Deep learning?
- Language modelling “features”
 - Tokenize text and pass to neural network layer
 - E.g., recurrent layer, convolutional layer, self-attention layer
 - Stack on 3+ more layers
 - Train a model to predict the next word (or a missing word) given previous words
 - Penultimate layer of network can be used to generate features for other language-based tasks
 - Basis for many state-of-the-art text classifiers
 - e.g. BERT (DistilBERT, RoBERTa..), XLNet, etc.

Björn Ross, TTDS 2024/2025



21

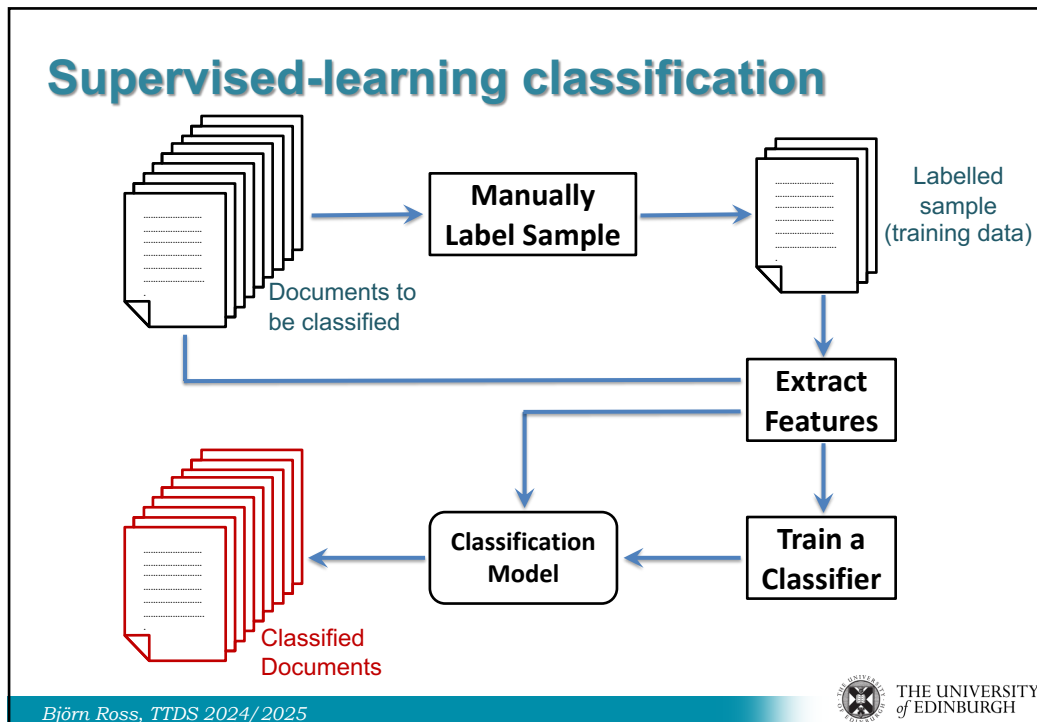
Step 3: Feature Weighting

- Attributing a value to feature t_k in document d_i
This value may be
 - **binary** (representing presence/absence of t_k in d_i);
 - **numeric** (representing the importance of t_k for d_i); obtained via feature weighting functions in the following two classes:
 - **unsupervised**: e.g., tfidf or BM25,
 - **supervised**: e.g., $tf * MI$, $tf * x^2$
- Similarity between two vectors may be computed e.g. via **cosine similarity**
- **Scaling** can be important!

Björn Ross, TTDS 2024/2025



22



23

Training a Classifier

- For **binary** classification, essentially any supervised learning algorithm can be used for training a classifier; classical choices include
 - Support vector machines (SVMs)
 - Random forests
 - Naïve Bayesian methods
 - Lazy learning methods (e.g., k-NN)
 - Logistic Regression
 -
- The “**No-free-lunch principle**” (Wolpert, 1996) → *there is no learning algorithm that can outperform all others in all contexts*
- Implementations need to cater for
 - the very high dimensionality
 - the sparse nature of the representations involved

THE UNIVERSITY
of EDINBURGH

Björn Ross, TTDS 2024/2025

24

Training a Classifier

- For **Multiclass classification**, some learning algorithms for binary classification are “SLMC-ready”; e.g.
 - Decision trees
 - Random forests
 - Naive Bayesian methods
 - Lazy learning methods (e.g., k-NN)
 - Neural networks
- For other learners (notably: SVMs) to be used for SLMC classification, combinations / cascades of the binary versions need to be used
 - e.g. multi-class classification SVM
 - Could be directly used for MLMC as well

Holding out test data

- It's important to avoid overfitting
- Labelled data could be split into **two parts**
 - **Training**: used to train the classifier (e.g. **80%** of the data)
 - **Test**: used to test the performance of the trained classifier on unseen data (e.g. **20%** of the data)

Hyperparameter optimisation

- Most classifiers have some hyperparameters to be optimized (we will usually refer to the ones we set manually as “hyperparameters” to distinguish from the “learned” parameters/weights of the model)
 - The C parameter in soft-margin SVMs
 - The r, d parameters of non-linear kernels
 - Decision threshold for binary SVM
- Usually labelled data is split into **three parts**
 - **Training**: used to train the classifier (typically **80%** of the data)
 - **Development**: used to optimise hyperparameters. Apply the classifier on this data with different values of the hyperparameters and report the one that achieves the highest results (usually **10%** of the data)
 - **Test**: used to test the performance of the trained classifier with the optimal hyperparameters on these unseen data (usually **10%** of the data)
- Optimising the hyperparameters on test data is cheating!

Evaluation

- Effectiveness (e.g. accuracy, precision, recall, F1):
 - Global effectiveness measures
 - Per class effectiveness measures
- Efficiency:
 - Speed in learning
 - SVM with linear kernel is known to be fast
 - DNNs are known to be much slower (specially with large # layers)
 - Speed in classification
 - K-NNs are known to be one of the slowest
 - Speed in feature extraction
 - BOW vs POS vs Link analysis features
- Importance of baselines

Evaluation: Baselines

- There are standard methods for creating baselines in text classification to compare your classifier with
- Most popular/simplest baselines
 - Random classification
 - Classes are assigned randomly
 - How much better is the classifier doing than random?
 - Majority class baseline
 - Assign all elements to the class that appears the most
 - How much better you are doing than if you always picked the same thing output regardless of input?
- Simple algorithm, e.g. BOW
 - Usually used when you introduce new interesting features

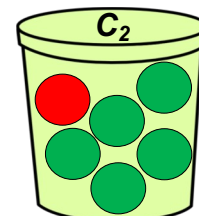
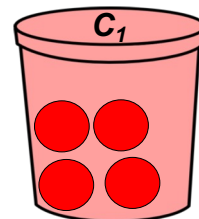
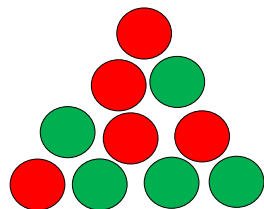
Björn Ross, TTDS 2024/2025



30

Evaluation: Binary Classification

- Accuracy:
 - How many of the samples are classified correctly?
- $A = (4+5)/10 = 0.9$



Björn Ross, TTDS 2024/2025



31

Evaluation: Binary Classification

- $A = (1+6)/10 = 0.7$ System 1
- $A = (0+7)/10 = 0.7$ System 2
- When classes are highly unbalanced
 - Precision/recall/F1 for the **rare class**
 - e.g. Spam classification (detection)

Björn Ross, TTDS 2024/2025
THE UNIVERSITY of EDINBURGH

32

Evaluation: Binary Classification

	System 1	System 2
Precision	$1/3 = 0.33$	$0/1 = 0$
Recall	$1/2 = 0.5$	$0/2 = 0$
F1	0.4	0

Björn Ross, TTDS 2024/2025
THE UNIVERSITY of EDINBURGH

33

Evaluation: Multi-class

- Accuracy = $(3+3+1)/10 = 0.7$
- Good measure when
 - Classes are nearly balanced
- Preferred:
 - Precision/recall/F1 for each class

	Green	Red	Blue
P	0.75	1	0.333
R	0.75	0.75	0.5
F1	0.75	0.86	0.4

- Macro-F1**
 $= (0.75+0.86+0.4)/3$
 $= 0.67$

Björn Ross, TTDS 2024/2025

34

Evaluation: Multi-class

- Majority class baseline
 - Accuracy = 0.8
 - Macro-F1 = 0.296
- Macro-F1:
 - Should be used in binary classification when two classes are important
 - e.g.: males/females while distribution is 80/20%

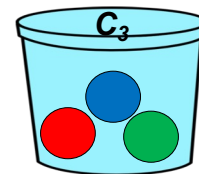
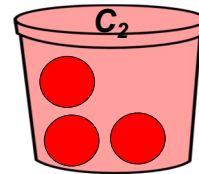
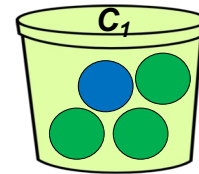
Björn Ross, TTDS 2024/2025

35

Error Analysis

- Confusion Matrix

		Predicted class		
		●	●	●
Actual class	●	3	0	1
	●	0	3	1
	●	1	0	1



- Useful:
 - Find classes that are confused with others
 - Develop better features to solve the problem

Björn Ross, TTDS 2024/2025



36

Summary

- Text Classification tasks
- Feature extraction/selection/synthesis/weighting
- Learning algorithms
- Baselines
- Evaluation measures
 - Accuracy/precision/recall/Macro-F1

Björn Ross, TTDS 2024/2025



37

Resources

- *Fabrizio Sebastiani*
Machine Learning in Automated Text Categorization
ACM Computing Surveys, 2002
Link: <https://arxiv.org/pdf/cs/0110053>
- *Yoav Goldberg*
A Primer on Neural Network Models for Natural Language Processing
Link: <https://arxiv.org/abs/1510.00726>