Classification of the 20 newsgroups dataset

ML lab 2024, Prof. Garcke, Dr. Bohn

Jonas Neuschäfer, Lena Siemer

January 31, 2026

Recap and Motivation

The dataset is a collection of newsgroup documents, originially of the form

- From:
- Subject:
- ...
- $\Rightarrow$ newsgroup (there are 20, more or less evenly distributed)

- Training data: 11314
- 18828 messages in total

Motivation: Trying out and combining different leverages to improve the classification (feature extraction + different classifiers vs. fine-tuning of parameters)

Outline

Step 1 Preprocessing the dataset

Step 2 Feature Extraction

- Bag-of-words model
- TF-IDF Vectorizing
- n-gram

Step 3 Comparison of Classifiers used in the lab

3.1 Setting up Feedforward Neural Network

3.2 Support Vector Machine, Nearest Neighbors, other Regression

Step 4 Evaluation of the performance on different tasks

- i.e. Grouping

Step 5: Classification via Fine-Tuning

Step 1: Preprocessing the dataset

Our preprossesing function includes the following steps:

1. Remove metadata
2. Delete symbols and punctuation
3. lower casing
4. Remove digits
5. Remove "very short" words
6. stop words
7. lemmatization

$\Longrightarrow$ Being able to reduce the length of a message, in selected examples, by approximately 50%.

Step 2: Vectorizing

Possible ways to vectorize:

- Bag-of-words
  - token counts
- TF-IDF vectorization
  - Term frequency $*$ Inverse document frequency $= \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} * \log \frac{N}{|\{d : d \in D \text{ and } t \in d\}|}$
- n-gram
  - sequence of n consecutive words in a text
  - possibly better 'understanding' of the sentence

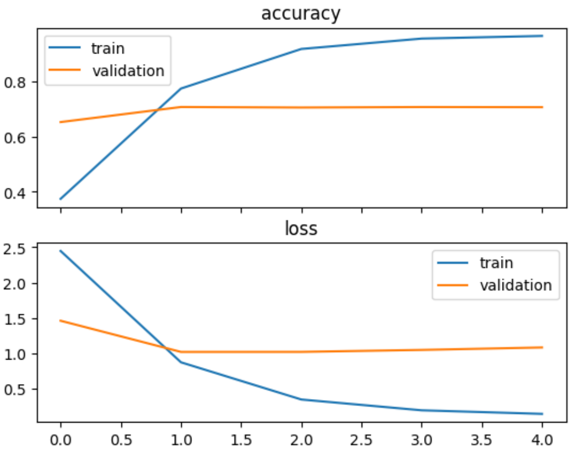
Vectorizing - first attempts to gain intuition of how good we are

- Depending on the steps included during the preprocessing of the dataset and the choice of a classifier we generally achieve a low accuracy with our **Bag-of-words** function.
  - Using **Ridge-Regression as benchmark**: $\approx 50\%$ on the whole dataset.
- Using the **TF-IDF Vectorizer** from sklearn already yields 70% accuracy.
- Including **n-gram** feature had no significant effect.
  - range $[1, 2]$

Hence, we decided to work with the TF-IDF features for the next few proceeding questions we want to address (including stop-word-filtering).

Step 3.1: Setting up a Neural Network for Classification

- Feedforward Neural Network with 2 layers (Dense).



→ Overfitting very fast! → adding Dropout: 71% accuracy

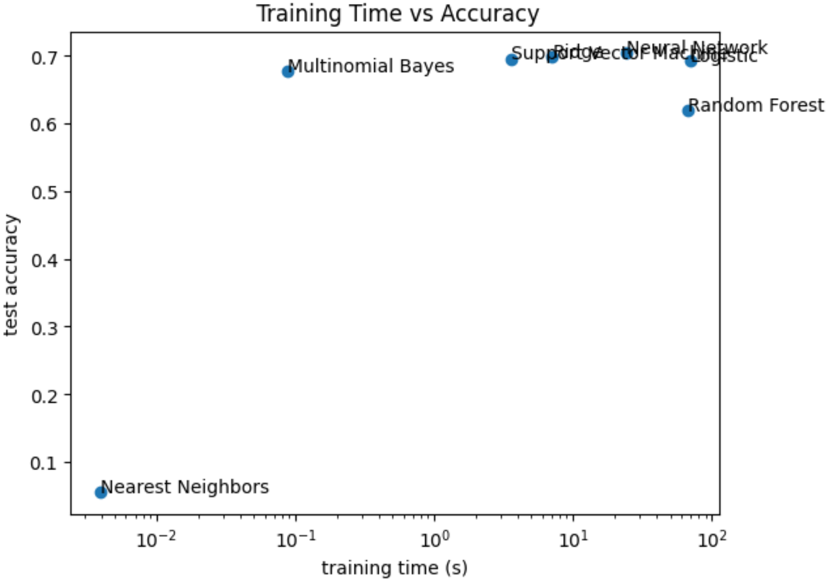
3.2 Comparison of Classifiers

- Multinomial Bayes
  - assumes that the presents of one feature does not affect the other
  - calculates probability distribution of text data
  - $P(D|c) = \frac{T_c!}{\prod_{i=1}^{V} x_i!} \prod_{i=1}^{V} \frac{\theta_{c,i}^{x_i}}{x_i!}$
    T_c is the total number of words in documents of class c
    x_i is the count of word i in document D
    $\theta_{c,i}$ is the probability of word i occurring in a document of class c

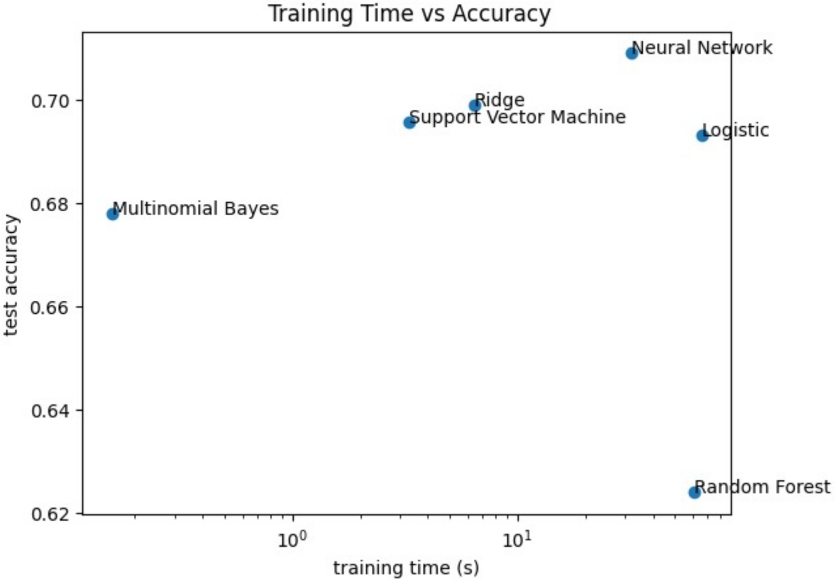
3.2 Comparison of Classifiers

- Ridge Regression
  - special version of linear regression
  - developed to deal with correlated attributes
  - encourages smaller, more evenly distributed weights by adding a penalty based on the square of the coefficients
- Logistic regression
  - special version of linear regression
  - fit data to logistic function $f(z) = \frac{1}{1+e^{-z}}$
  - calculates probabilities
- Random Forest
  - create uncorrelated decision trees
  - train each with a different, random part of the data
  - prediction: aggregate all predictions of the trees

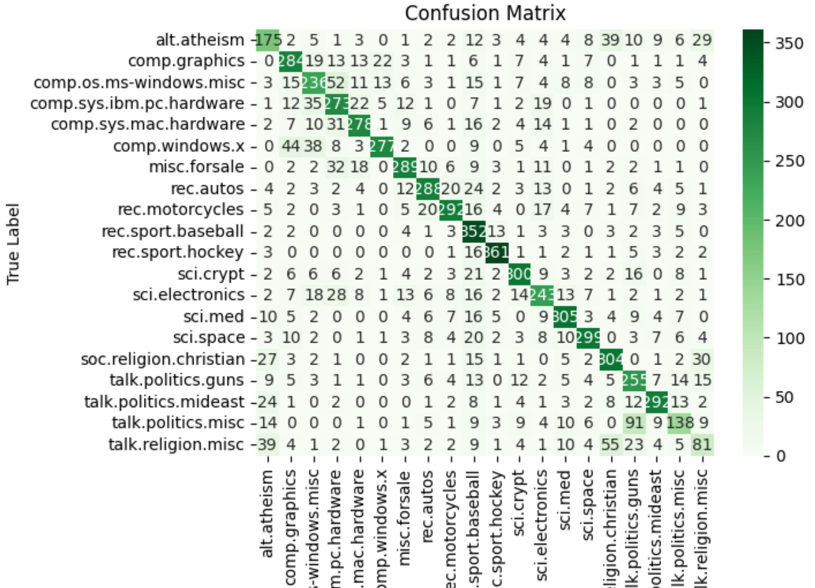
3. Comparison of different classifiers



3. Comparison of different classifiers



Confusion matrix



Accuracy Report

	precision	recall	f1-score	support
alt.atheism	0.62	0.46	0.53	319
comp.graphics	0.71	0.72	0.71	389
comp.os.ms-windows.misc	0.63	0.66	0.65	394
omp.sys.ibm.pc.hardware	0.65	0.67	0.66	392
comp.sys.mac.hardware	0.74	0.71	0.73	385
comp.windows.x	0.85	0.70	0.77	395
misc.forsale	0.44	0.84	0.58	390
rec.autos	0.76	0.76	0.76	396
rec.motorcycles	0.92	0.67	0.78	398
rec.sport.baseball	0.89	0.82	0.85	397
rec.sport.hockey	0.95	0.90	0.92	399
sci.crypt	0.91	0.67	0.77	396
sci.electronics	0.57	0.64	0.60	393
sci.med	0.77	0.78	0.77	396
sci.space	0.87	0.70	0.78	394
soc.religion.christian	0.57	0.84	0.68	398
talk.politics.guns	0.63	0.61	0.62	364
talk.politics.mideast	0.91	0.73	0.81	376
talk.politics.misc	0.57	0.48	0.52	310
talk.religion.misc	0.33	0.35	0.34	251

Top features with scores

talk.politics.guns:
bd: 1.423
nra: 1.426
fbi: 1.444
weapon: 1.464
jmd: 1.591
firearm: 1.794
weapons: 2.051
guns: 2.183
gun: 2.699

talk.politics.misc:
deane: 1.178
taxes: 1.181
homosexuals: 1.183
libertarian:1.247
blacks: 1.728
jobs: 1.287
drugs: 1.403
tax: 1.566
libertarians: 1.576

4. Grouping

alt.atheism

comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x

rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey

sci.crypt
sci.electronics
sci.med
sci.space

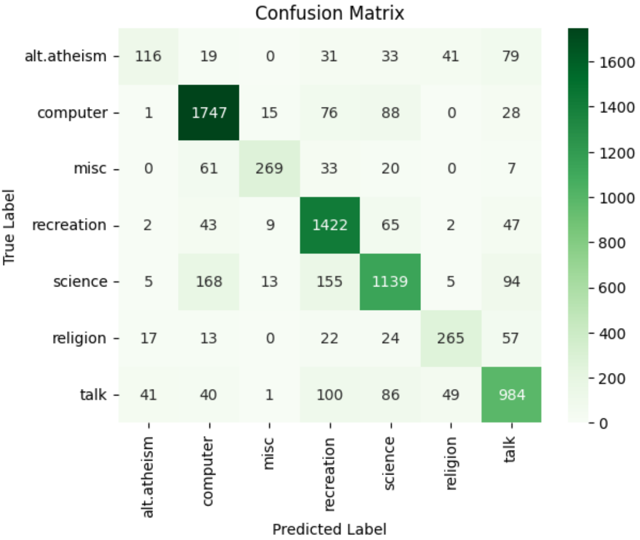
soc.religion.christian

talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc

misc.forsale

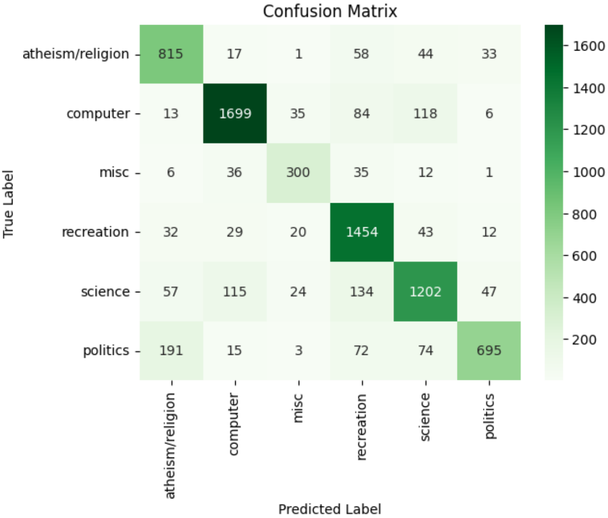
Grouping - Result

- We achieve an accuracy of 79% using Ridge Regression.



Comparison with Monday (not in the notebook)

- We achieve an accuracy of 82% using our Neural Network.



5. Classification via Fine-Tuning

- Working with transformers
  - Introduced by Vaswani et al. in 2017
  - Input Embedding: Word $\mapsto \underbrace{\begin{pmatrix} \\ \\ \\ \end{pmatrix}}_{\text{word vector}} + \underbrace{\begin{pmatrix} \\ \\ \\ \end{pmatrix}}_{\text{position encoding}}$
  - Each of Transformer's encoder layers comprise
    - Attention mechanism, where scores are calculated how relevant each word is to another (attention-weights)
    - Fully Connected Feedforward Neural Network
- We used one version of the BERT base model.
- Sub-word tokenization (encoding) of the messages (assigns $input - ids$ and $attention - mask$)
- Fine-tuned by specifying the training arguments (optimize weighted average of f1-score)

5. Classification via Fine-Tuning

- Trying it with only 4 randomly picked categories (unfortunately similar): ['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware']
- We achieve an accuracy of 80.7% .

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.977100	0.675589	0.744980	0.737853
2	0.501800	0.569718	0.790495	0.790333
3	0.375000	0.552118	0.806560	0.806923

Example Text

Example message after preprocessing: "24bit color dpi fladbed scanner job gif tiff pcx bmp interested please write imagesyzaolcom"

- After encoding we can read out the probabilities: