
FABLE FILTER

Administrative

- Name: Fable Filter
- CAS Machine Intelligence 23.10E
  - Text Analytics
  - Advanced Big Data
- Group
  - Marco Gsponer
  - Tobias Ambühl
- Link to Github: https://github.com/2bonahill/fable-filter

Goal and problem statement

- The digital information landscape is cluttered with fake or misleading news articles
- A dataset with articles, specifically labeled as "fake", "conspiracy", "rumor", “clickbait” etc. provides a unique opportunity for training an AI-based detection system
- The primary aim is to develop an AI system capable of effectively detecting and categorizing new articles

First problem

- A dataset of almost 10 million labelled news articles is too much to handle using a traditional approach
- So how can we find a better way to master a vast quantity of text?

Parallelization and final dataset

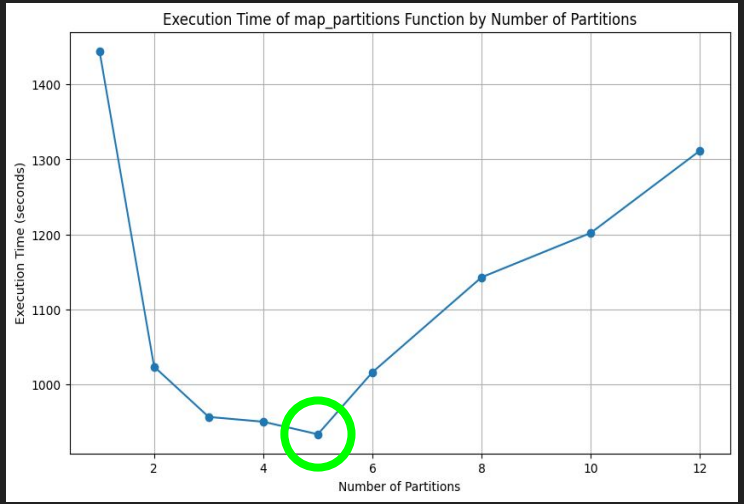
- We use dask's parallelization mechanisms to optimize the work of data preprocessing
- And we use Python's time and memory profiling capabilities for measuring
- Final dataset of roughly 70k truncated samples.

reliable	18655
political	12673
bias	8578
fake	7985
conspiracy	6489
rumor	4550
unknown	2961
clickbait	2276
unreliable	1202
junksci	974
satire	932
hate	744

Name removal

Tuning the number of partitions resulted in almost 50% efficiency gain for name removal -> From 28 min to 16 min

```python
import time

# Function to remove names
def remove_names(text):
    doc = nlp(text)
    for ent in doc.ents:
        if ent.label_ in ['PERSON']:
            text = text.replace(ent.text, '')
    return text

for i in range(1,12):
  dfc = df.copy()
  dfc = dfc.repartition(npartitions=i)
  # dfc = dfc.persist()

  # Start timing
  start_time = time.time()

  dfc['content'] = dfc.map_partitions(lambda partition: partition['content'].apply(remove_names))
  dfcp = dfc.persist()

  # End timing
  end_time = time.time()

  # Calculate and print the duration
  duration = end_time - start_time
  print(f"The map_partitions function took {duration} seconds to execute for {i} partitions.")
```

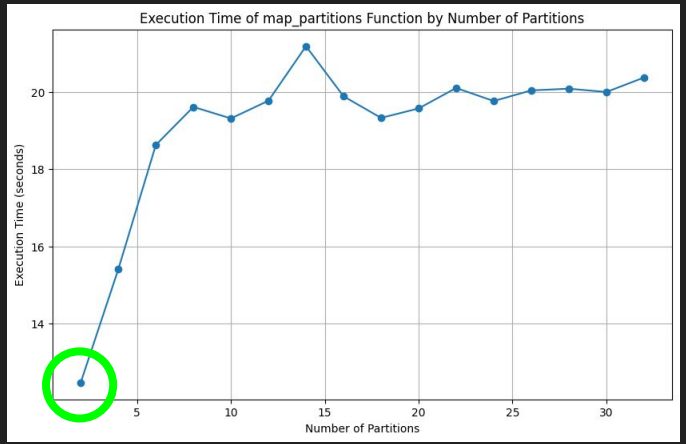


Text cleanup

Adding more partitions had a negative effect for the text cleanup process. Possible reasons:

1. Scheduling Overhead: The Dask scheduler has to manage more tasks, which can introduce significant overhead, especially if the tasks are small and fast. This overhead can sometimes outweigh the benefits of parallelism.
2. Communication Overhead: More partitions mean more data needs to be shuffled across the cluster, which can increase network traffic and latency.

```python
import re
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')

def clean_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Remove stopwords (common words that may not add much meaning)
    stop_words = set(stopwords.words('english'))
    text = ' '.join([word for word in text.split() if word not in stop_words])

    return text
```



Training different models for the classification task

- TF-IDF
- LSTM - Using Optuna as the optimization framework to automate hyperparameter search
- DistilBERT - A small, fast, cheap and light Transformer model trained by distilling BERT base (Hugging Face)

Baseline model with TF-IDF vectorization

1. Basic Model using Naive Bayes Classifier

	precision	recall	f1-score
bias	0.77	0.13	0.22
clickbait	1.00	0.00	0.00
conspiracy	0.98	0.10	0.18
fake	0.99	0.25	0.40
hate	1.00	0.00	0.00
junksci	1.00	0.00	0.00
political	0.41	0.50	0.45
reliable	0.38	0.96	0.54
rumor	0.99	0.18	0.30
satire	1.00	0.00	0.00
unknown	1.00	0.04	0.07
unreliable	1.00	0.22	0.36
accuracy			0.43

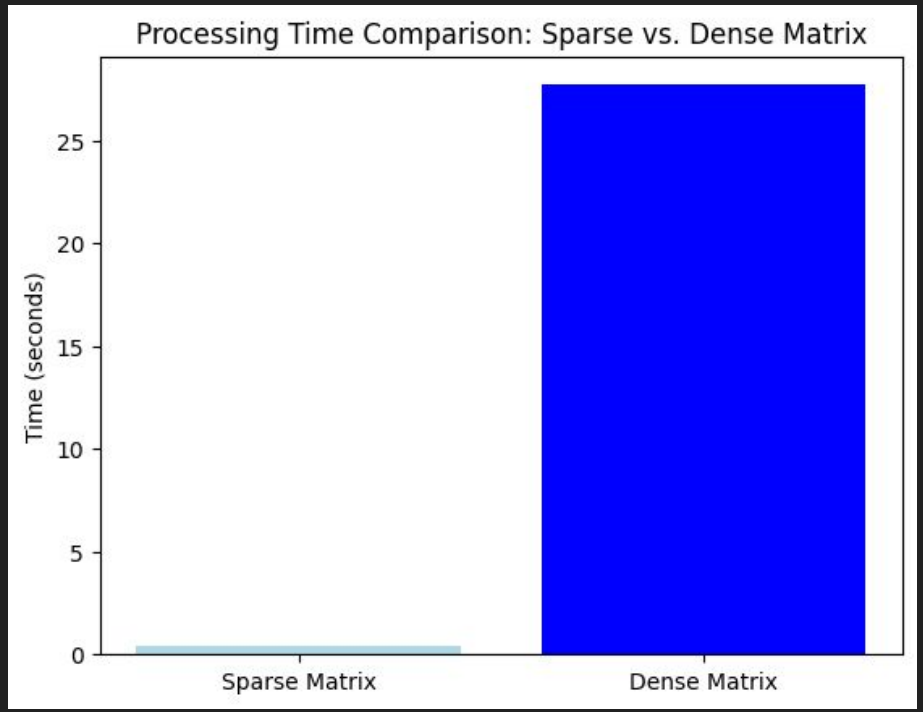
-> Problem: Imbalanced classes!

2. Naive Bayes with oversampling of minorities
   -> accuracy rises to 0.53

3. Use logistic regression with class weights
   -> accuracy goes up to 0.6

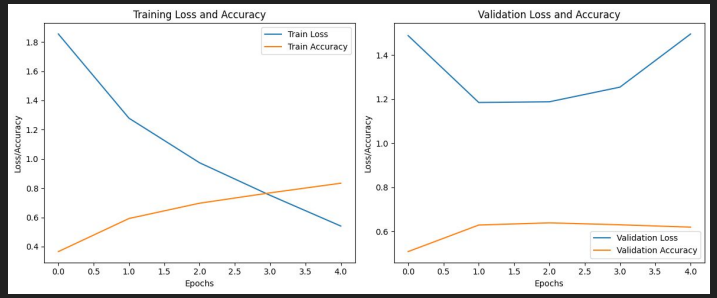
	precision	recall	f1-score
bias	0.54	0.50	0.52
clickbait	0.27	0.43	0.33
conspiracy	0.50	0.51	0.51
fake	0.81	0.72	0.76
hate	0.16	0.35	0.22
junksci	0.35	0.72	0.47
political	0.58	0.47	0.52
reliable	0.81	0.74	0.77
rumor	0.70	0.83	0.76
satire	0.11	0.19	0.14
unknown	0.34	0.44	0.38
unreliable	0.45	0.38	0.41
accuracy			0.60

Effects of sparsity in TF-IDF on processing time



LSTM

Optimizing parameters with optuna (vocab_size, embedding_dim, lstm_units, dropout_rate, batch size and number of epochs)



The model seems to be overfitting (adding dropout layers, early dropout didn't help), its learning well on the Trainset. Maybe training with more data would help.

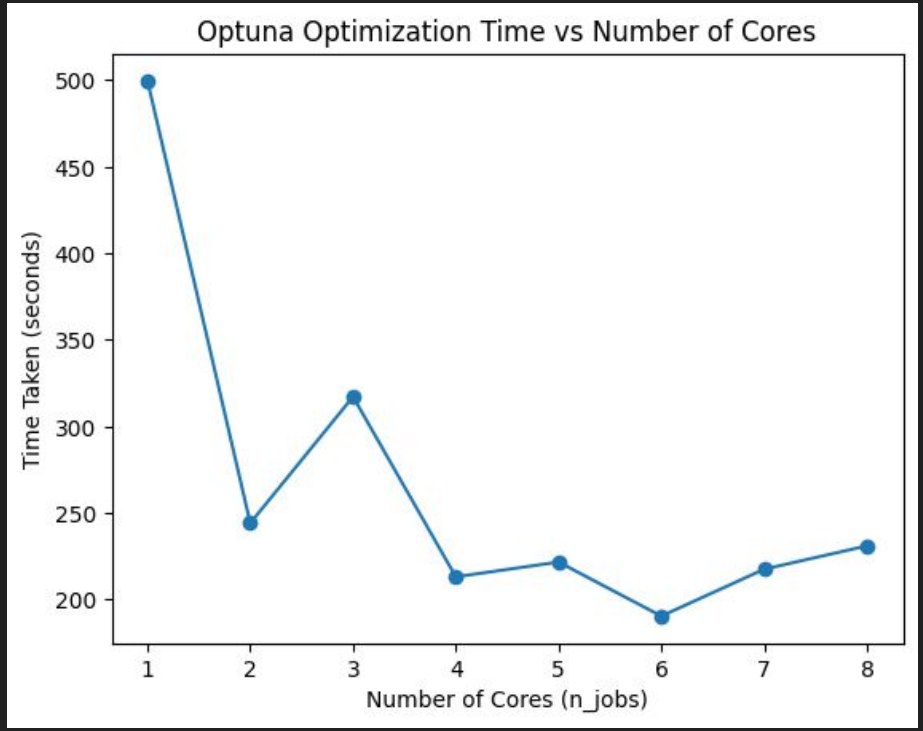
Adding layers didn't make the results better

Using on Testset:

	precision	recall	f1-score
bias	0.44	0.57	0.50
clickbait	0.40	0.27	0.32
conspiracy	0.57	0.46	0.51
fake	0.88	0.80	0.84
hate	0.19	0.14	0.16
junksci	0.66	0.60	0.63
political	0.49	0.56	0.52
reliable	0.78	0.77	0.78
rumor	0.79	0.69	0.74
satire	0.17	0.12	0.14
unknown	0.41	0.41	0.41
unreliable	0.67	0.36	0.47
accuracy			0.62

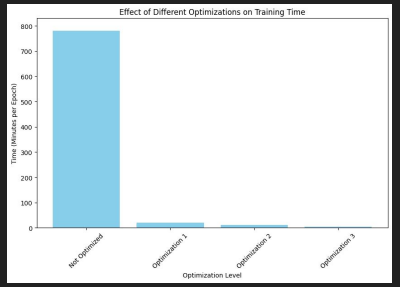
-> Predictions seem to be fairly good for fake, reliable and rumor

Effects of parallelization on the use of optuna

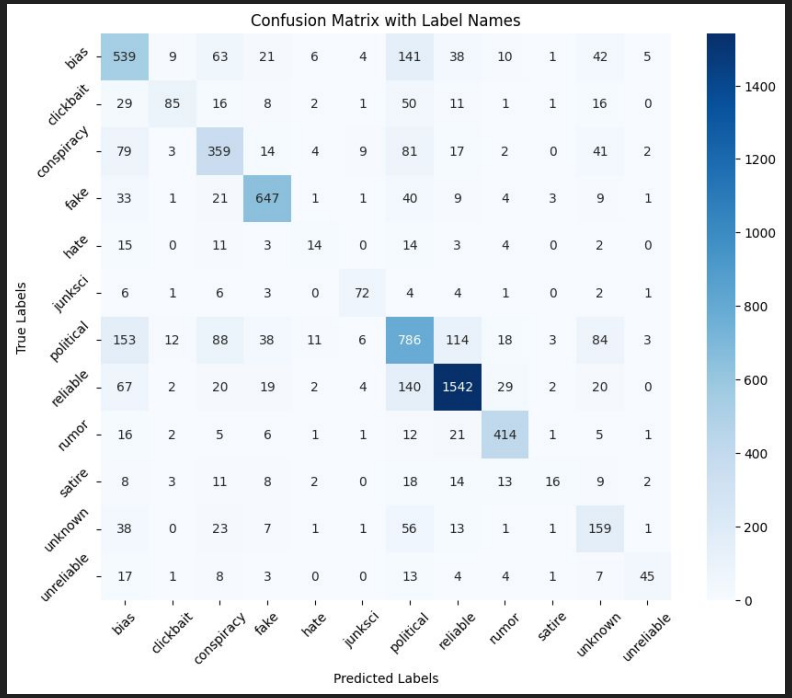


DistilBERT

- Not optimized
  - CPU
  - Batch Size: 32
  - Approximately 13 hours! per Epoch
- Optimization 1
  - GPU
  - Batch Size 32
  - Approx: 20 minutes per Epoch
- Optimization 2
  - GPU
  - Batch Size 128
  - Approx. 12 minutes per Epoch
- Optimization 3
  - GPU
  - Prefetching and Parallel Data Loading
  - Batch Size 128
  - Approx. 4 minutes per Epoch!!




```
54/54 [==============================]
Validation results:
loss: 1.9270278215408325
accuracy: 0.6950896978378296
```



Inference

- Testing with some German news articles
- Translation into English via ChatGPT
- Little prediction method to classify input

```
# SRF News
make_prediction("Ukrainian President Volodymyr Zelenskyy is traveling to the World Economic Forum (WEF) in Davos next week. This was confirmed by the organizers…")
```

BIAS

```
# SRF News
make_prediction("A court in Moscow has imprisoned Russian-born US citizen Robert Woodland Romanov. Analysts suspect a connection to other US citizens in Russian custody…")
```

BIAS

```
# SRF News
make_prediction("Marco Odermatt also dominates the first training for the downhill races at the Lauberhorn, achieving the fastest time…")
```

RELIABLE

```
# Weltwoche
make_prediction("Gabriel Attal becomes France's new Prime Minister. At 34 years old, he will be the youngest and first openly homosexual head of government of the Grande Nation…")
```

HATE

```
# Weltwoche
make_prediction("Swiss style is farmer’s style: Why rural mass protests in Switzerland, unlike in Germany, are absent…")
```

POLITICAL

Conclusions

- Through the use of parallelization with Dask we were able to handle the initial data and get a usable sample
- We were able to reduce processing time with the use of sparsity in tf-idf vectorization, parallelizing parameter optimization tasks with optuna of our lstm model and parallel data loading during DistilBERT model training
- The best accuracy was achieved by fine tuning the DistilBERT model (0.696)
- Fake and reliable news were recognized the best
- To further improve our models we suggest to use a bigger training dataset with a more even distribution of categories