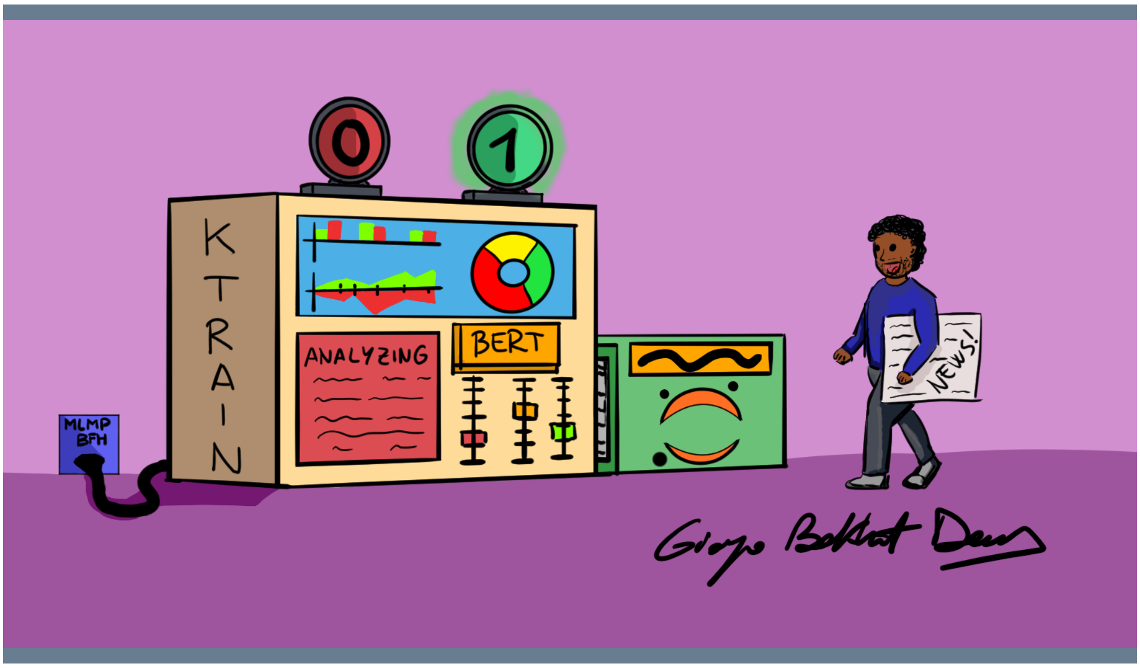


Sentiment Analysis on Swiss Newspapers

Project documentation

Bachelor thesis

Today's newspapers have the power to shape one's entire perspective on the world. For that reason, it is very important to be able to differentiate between newspapers that keep one in a rather pessimistic state of being, or newspapers that offer a more optimistic outlook on the world. The idea is to create, train, adjust and improve a model so that it is fit to analyze various Swiss newspapers and to determine which papers are written with the most negative and positive attitude.

Degree course:	BSC Computer Science
Author:	Giorgio Bakhiet Derias
Tutor:	Prof. Dr. Mascha Kurpicz-Briki
Expert:	Andreas Dürsteler
Date:	June 15, 2021

Versions

Version	Date	Status	Remarks
0.1	22.02.2021	Draft	First draft
0.1.1	23.02.2021	Draft	Contents added
0.2	14.03.2021	Addition	Chapter 1 added
0.3	18.03.2021	Addition	Chapter 2 added
0.4	27.03.2021	Addition	Chapter 3 added
0.4.1	03.04.2021	Addition	Chapter 3 extended
0.5	05.04.2021	Addition	Chapter 6,7 added
0.6	11.04.2021	Correction	Chapter 1,2,3,5 corrected
0.7	19.04.2021	Addition	Chapter 4,5 added
0.8	10.05.2021	Correction	Chapter 4,5,6 corrected
0.9	02.06.2021	Correction	Chapter 5,6,7 corrected
1.0	15.06.2021	Final	Completed

Contents

1 Introduction

> *"Things like chatbots, machine learning tools, natural language processing, or sentiment analysis are applications of artificial intelligence that may one day profoundly change how we think about and transact in travel and local experiences."*
>
> Gillian Tans

Sentiment Analysis or Opinion Mining is a way of finding out the polarity or strength of the opinion (positive or negative) that is expressed in written text [1].
It is used in business to understand social sentiment for their brand, a particular product or service.

1.1 Previous work

In the last semester, my project partner and I have for the first time stepped into the world of sentiment analysis, making use of various supports such as tutorials, explanations, and lots of research. We then decided to use all the material we had found to build something of our own, so we built a tool to categorize the sentiment of hotel reviews. We wanted to split the sentiment of the reviews into 5 categories: from awful to excellent. At the end of the project, we managed to have an accuracy of about 0.48, taking into consideration the 5 categories was a good result.

1.2 Project description

The thesis project I am working on this semester will still be on sentiment analysis. This time the model created will be used to make predictions about newspaper news, to understand the polarity (positive or negative) of an article or the newspaper itself. A newspaper can change people's way of thinking and mood because of its polarity. That is why it is vital to understand its importance. By having the polarity of a newspaper, I can also calculate a lot of things such as: the polarity of a newspaper over time, the various categories, and which of these newspapers affect us the most.

1.3 Motivation

As already mentioned, this technology allows to crystallize negative and positive feelings from a text. Besides sentiment classification, sentiment analysis brings many other benefits in areas such as marketing and customer satisfaction. Some of these benefits could be, for example:

- **Customer classification**
  sentiment analysis allows customers to be classified according to their emotional mood. This offers the opportunity to find customers who are more willing to buy.
- **Chatbot training**
  With the results of the sentiment analysis tool, it is possible to train chatbots to recognize and respond to specific customer sentiments.
- **Scalability and automation**
  As a digital tool, sentiment analysis can be easily extended or integrated into an automated system.

From these points, a strongly increasing tendency regarding the application of sentiment analysis can be assumed. This makes it even more relevant for developers to get to grips with it. From the developer's perspective, learning sentiment analysis also promotes new experiences in areas such as:

- Data analytics
- Data science
- Machine learning
- Predictive modelling

1.4 Goal

The core of this project is with the help of different tools like Tensorflow [2], Keras [3] and BERT [4] to create a model that takes text as input and classifies it in "positive" or "negative". To calculate the polarity of the article, I will use different datasets to create different models, in the end I will use the model and dataset that most reflects my purpose. Unfortunately, due to an impediment, my partner from the previous project was not able to take part in this assignment, which means that I must continue this project alone. For a matter of timing and difficulty, I have resized the project and the categories on which to make a prediction will no more be 5 as in the last project, but 2. This model should be available as a single component that can be integrated into arbitrary applications.
For the implementation of the project, I have used several websites and tutorials:

- Tutorial for Keras [5]
- Deep-Learning-For-Hackers [6]
- Sentiment Analysis with TensorFlow 2 and Keras [7]
- Deep Learning LSTM for Sentiment Analysis [8]
- Natural Language Processing and Sentiment Analysis using Tensorflow [9]
- Sentiment Analysis: First Steps With Python's NLTK Library [10]
- Sentiment Analysis using Deep Learning with Tensorflow [11]
- Practical Text Classification With Python and Keras [12]
- Classify text with BERT [13]
- Text Classification with BERT using Transformers for long text inputs [14]
- Simple Transformers — Multi-Class Text Classification with BERT, RoBERTa, XLNet, XLM, and DistilBERT [15]
- Fine-tuning BERT with Keras and tf.Module [16],
- The 1cycle policy [17],
- Ktrain tutorials [18].

1.5 Project organization

Table 1 describes who occupies which role in our project organization:

Project organization		
Role in the project organization	**Name**	**BFH Abbreviation**
Tutor	Mascha Kurpicz-Briki	kim3
Expert	Andreas Dürsteler	-
Developer	Giorgio Bakhiet Derias	bakhg1

Table 1: Project organization

1.6 Tools

In Table 2 is possible to find the different tools I used for this project:

Tools	
Tool	**Description**
Jupyter Notebook	A document that can store code, diagrams, graphics and much more.
Google Colaboratory	Online platform to host Jupyter Notebook.
Kaggle	Is an online machine learning environment and community.
Anaconda	Application to manage libraries of larger projects.
Virtual Machine	Anaconda is installed on the virtual machine to run the project.
PyCharm	IDE used for the Python language. It is developed by the company JetBrains.
GitHub	GitHub is a hosting service for software projects.
Overleaf	Overleaf is a cloud-based collaborative LaTeX editor used to write, edit, and publish scientific papers.
MLMP	MLMP is a cloud-based environment for machine learning created by BFH.
Scikit-learn	Open-source machine learning software library for the Python programming language.
Tensorflow	Open-source software library for machine learning.
Pandas	Open-source data analysis and tool for data manipulation, built in Python.

Table 2 continued from previous page

Tool	Description
Keras	Tensorflow library specifically for creating neural networks.
Ktrain	Ktrain is a library used to build, train, debug, and deploy neural networks in the deep learning software framework Keras.
BERT	Is a recent paper published by researchers at Google AI Language.
Hugging Face	Is an open-source provider of natural language processing (NLP) technologies.
Plotly	Plotly provides online graphing, analytics, and statistics tools for individuals and collaboration, and libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST.

Table 2: Tools

1.7 Project pipeline

The Project pipeline is shown in Figure 1, the pipeline goes from top to bottom. The pipeline is categorized by color, in fact each color is a key component of the project:

- in orange there is the part that concerns the creation of the dataset (Chap:3,4),
- in red the part related to Ktrain (Chap:5.2),
- orange and red together will create the model (Chap:5.2.2),
- in green the data coming from the APIs, this data will be given to the trained model to make predictions (Chap:5.4),
- in blue the components related to the calculation of polarity, analysis of results and creation of visualization (Chap:5.5).

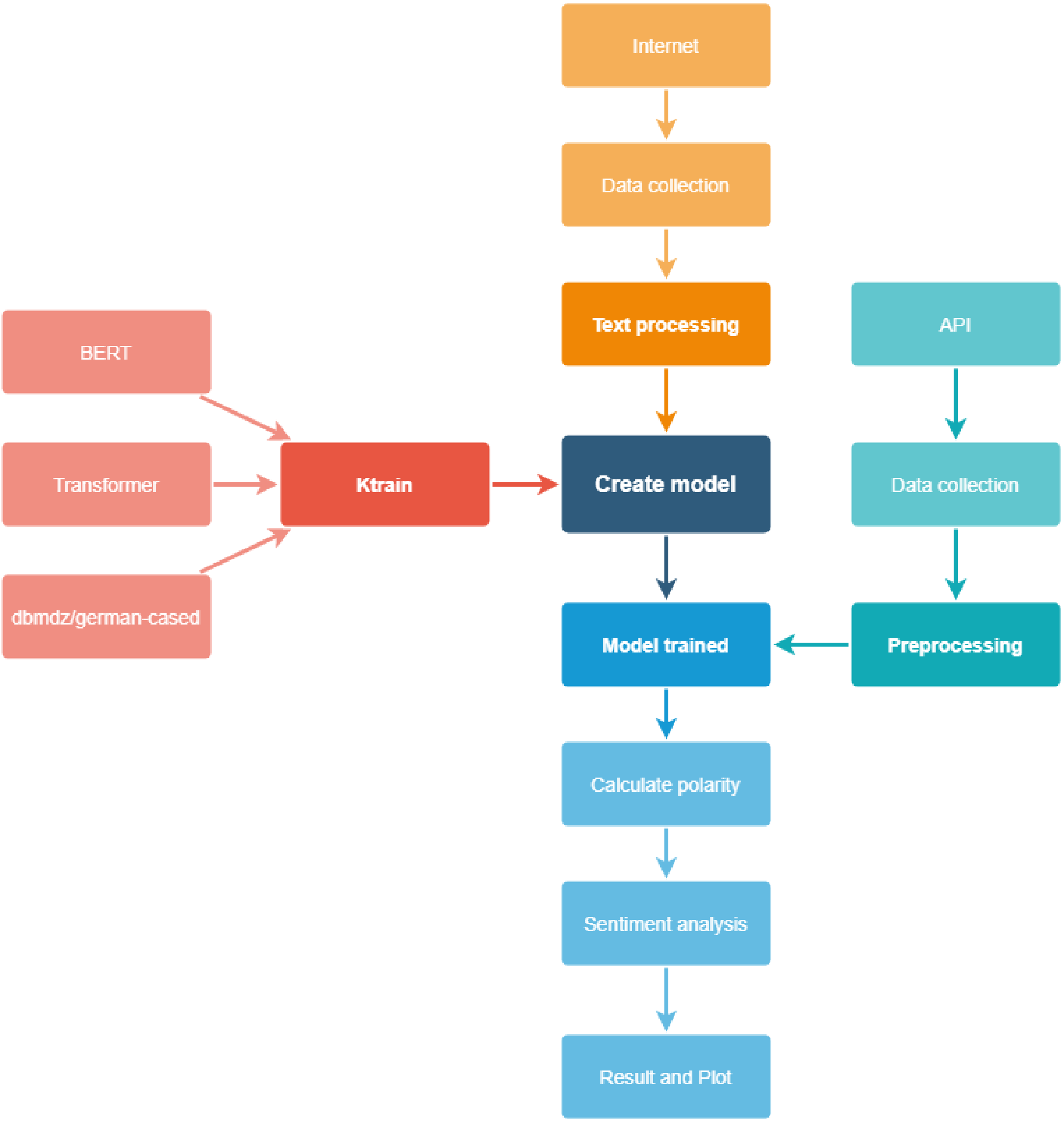


Figure 1: Project pipeline

2 Technology and setup

This section will explain all the components used for the development environment.

2.1 Jupyter Notebook

The model is developed in the form of a Jupyter Notebook [19] using the Python programming language. Jupyter Notebooks are documents that can contain code, text, images, diagrams and explanations. This is especially suitable for this project, because in Machine Learning it is often useful to explain the source code with graphs and diagrams.

2.2 Google Colaboratory

The notebook was initially hosted on Google Colaboratory [20]. This brought the following advantages:

Synchronization By storing the notebook in a central location, Google Drive, and having everyone access the same notebook, all changes are immediately visible to all users of the notebook.

Project structure The Jupyter Notebook allows to map the whole project in a structured way in a single file.

Project dependencies Since the project is hosted on Google Colab, it is not necessary as a developer to install libraries and other dependencies locally.

However, as it turned out, Colab removes from the platform external resources that were imported into the project after a certain time. This meant that I had to re-import the datasets, which are essential for the development of my model, before each work. This forced me to set up a Virtual Machine on which the notebook can then be run locally.

2.3 Virtual machine

To have the tools installed in one place, I opted for a virtual machine. Not only this, the VM has precisely other advantages, which are:

- more computing power
- do not use my local machine
- is always on, so it can always work
- the models are trained on a special machine with special software
- being online I can work from any location
- always working I can train my models also at night
- if by chance something goes wrong, I can backup and create another VM

2.4 Kaggle

What is Kaggle [21]? As it is possible to find written in the documentation:

> ”Kaggle is an AirBnB for Data Scientists - this is where they spend their nights and weekends.” – Zeeshan-ul-hassan Usmani [22]

Founded in 2010 by Anthony Goldbloom (CEO) and Ben Hamner (CTO), and acquired by Google in 2017, Kaggle enables data scientists and other developers to engage in running machine learning contests, write and share code, and to host datasets.

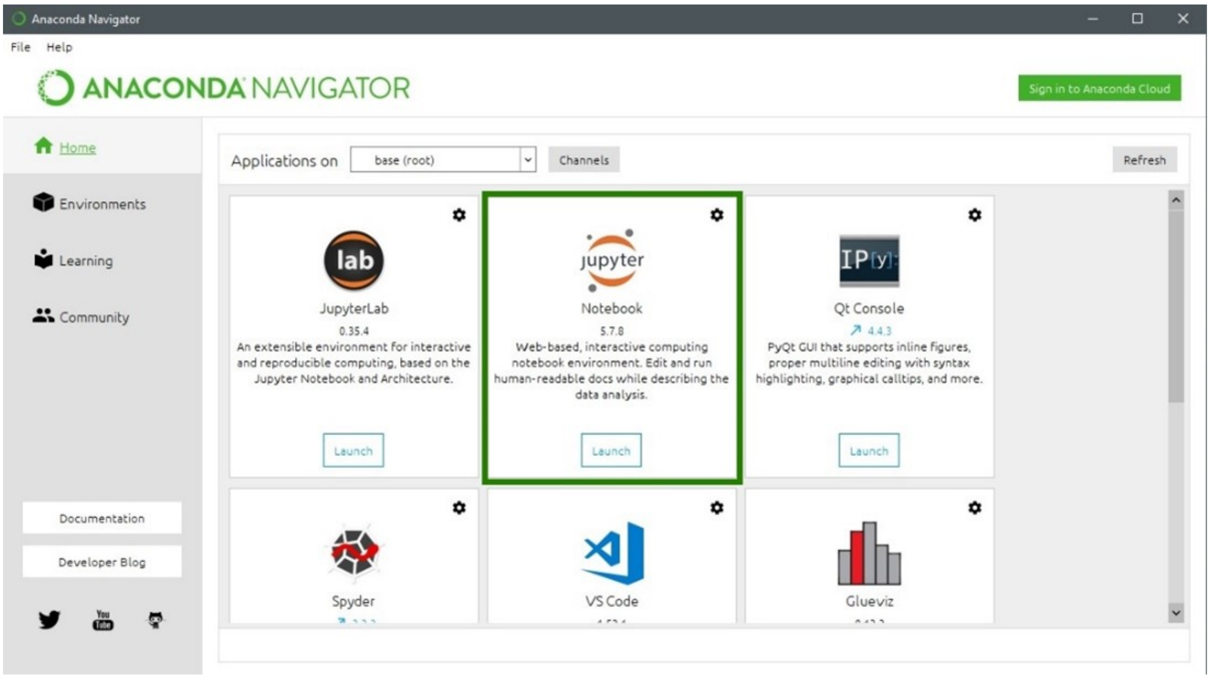
Kaggle is an online community for data scientists that offers:

- machine learning competitions,
- datasets,
- notebooks,
- access to training accelerators,
- education.

Thanks to its 35k datasets, Kaggle is the perfect place to start a search for a datasets, notebooks or information.

2.5 Anaconda

Anaconda [23] is a distribution of the Python and R programming languages, used for data science, machine learning etc. Anaconda is used to simplify package management and deployment of various libraries. To run Jupyter Notebook with all the needed libraries, Anaconda has turned out to be a suitable solution. Anaconda is a platform that allows to set up large projects locally by creating so-called ”environments” that contain all the required configurations and libraries for the project. To my advantage, Anaconda already provides a pre-built environment for Jupyter Notebook projects, which is already equipped with all my needed tools, as shown in Figure 2.



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Figure 2: Anaconda Navigator

Theoretically, it would also have been possible for each developer to install Anaconda on their computer and work on the notebook, which is maintained on a GitHub repository. However, since the project is very hardware-heavy and compiling a machine learning model can take up to several hours, a Virtual Machine turns out to be the better option.

2.6 PyCharm

PyCharm [24] is an integrated development environment (IDE) used for programming in the Python language, created by the Czech company JetBrains. I used it because it supports anaconda, GitHub and as a debugger. It natively supports jupyter notebooks is perfect because it is much more comfortable than jupyter itself, then it has several extensions that make programming easier and more intuitive.

2.7 Github

GitHub [25] is a provider with the function of hosting for software development and version control using Git. Thanks to its free of charge nature it is used for most of the open-source projects. That is why it is loved by the community of programmers, even I use it for this project to save files and use them from different locations.

2.8 Overleaf

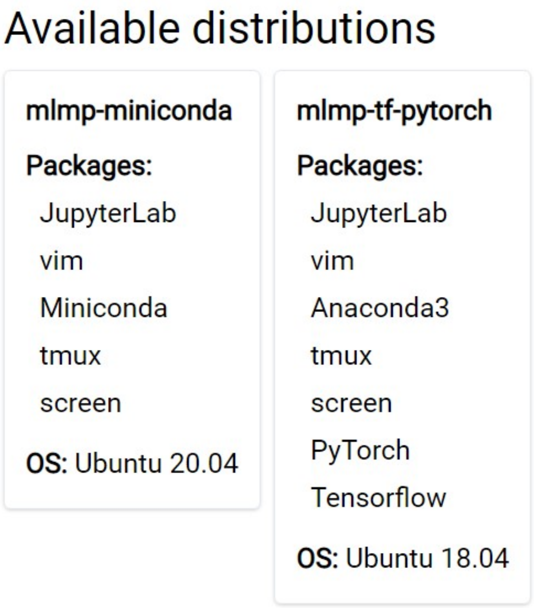
Overleaf [26] is cloud-based LATEX editor, which makes writing a scientific paper collaborative. LATEX is a software widely used in the academic world to create scientific texts, with useful formatting for mathematical, statistical, computer science, physics texts, which can be problematic on other text editors. I preferred Overleaf to other solutions for the convenience, in fact there is no need to install anything, everything is accessible via the internet as with

Colab. Overleaf was useful for me to write this documentation, thanks also to its versioning nature.

2.9 MLMP

MLMP [27] is a cloud-based environment in which machine learning models can be trained. Similar to Colab, this environment was created by Department of Engineering and Information Technology of the Bern University of Applied Sciences (BFH), to allow its students to be able to train models, with a more performing machine. It is possible to use it in the BFH network or through a connection to the network via VPN, the advantage lies in being able to use not only the CPU, as in normal virtual machines, but also very powerful GPUs, designed precisely for this purpose.This way I can train my model much faster, having much more computing power at my disposal.

Figure 3 shows the available environments on MLMP:



Available distributions

mlmp-miniconda

Packages:
JupyterLab
vim
Miniconda
tmux
screen

OS: Ubuntu 20.04

mlmp-tf-pytorch

Packages:
JupyterLab
vim
Anaconda3
tmux
screen
PyTorch
Tensorflow

OS: Ubuntu 18.04

Figure 3: MLMP distributions

Which distribution to use? To work I chose the distribution **mlmp-tf-pytorch** because it has the most suitable components for the work I need to do such as: Tensorflow and Anaconda in full version.

As can be seen from this blog [28] the GPU takes much less time to work than the CPU.

2.10 Scikit-learn

Scikit-learn [29] is a free machine learning library for Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

2.11 Pandas

Pandas [30] (”Python Data Analysis Library”) is an open-source library written in Python, for data analysis and data manipulation tasks. Pandas can be useful for example:

- Convert Python lists or dictionaries, Numpy arrays, to Pandas data frames.
- Inspecting data frames with a lot of functions.
- Data manipulation like filter, sort or group by and also data cleaning.
- Import local datasets that can be in different formats like CSV, TSV, Excel, etc.
- Access remote files or CSV databases or even websites in JSON format or read SQL tables or database.

2.12 Tensorflow

Tensorflow [31] is a machine learning framework from Google, which facilitates the process of capturing data, training models, making predictions, and refining future results.

Tensorflow is an open-source library for large-scale numerical computing and machine learning, it bundles a number of machine learning and deep learning algorithms and models. All this is provided through the Python language; since it is easy to learn and implement. The actual mathematical operations, however, are performed in high-performance C++.

2.13 Keras

Keras [3] is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the Tensorflow library.

A model with Keras consists of multiple layers, each of them performs a new data transmutation on the result from the layer above, at the end we will have a series of layers connected like a neural network.

Neural networks do not process raw data, like text files, encoded JPEG image files, or CSV files. They process vectorized and standardized representations.

For this reason, Keras comes to our support, in fact we can use Keras for all those tasks of data loading and data preprocessing.

Not only this, with Keras can also perform many other tasks such as:

- Build a model
- Train a model with the method fit()
- Evaluate a model
- Customize the method fit() for fine-tuning

See references for more details [32]

2.14 BERT

BERT [4] (Bidirectional Encoder Representations from Transformers) is an open-source deep learning model developed by Google and stat of the art for NLP tasks.

The biggest difficulty is finding the part of BERT that suits best. In fact, this task needed a lot of time and research to be able to figure out which BERT is the right one for this project.

In the list below are the different versions there are of BERT:

- BERT-Base, uncased and seven more models with trained weights released by the original BERT authors.
- Small BERTs have the same general architecture but fewer and/or smaller Transformer blocks, which lets to explore tradeoffs between speed, size and quality.
- ALBERT [33]: four different sizes of "A Lite BERT" that reduces model size (but not computation time) by sharing parameters between layers.
- BERT Experts [34]: eight models that all have the BERT-base architecture but offer a choice between different pre-training domains, to align more closely with the target task.
- ELECTRA [35] has the same architecture as BERT (in three different sizes), but gets pre-trained as a discriminator in a set-up that resembles a Generative Adversarial Network (GAN).
- BERT with Talking-Heads Attention [36] and Gated GELU [37] [base, large] has two improvements to the core of the Transformer architecture.

How BERT works? BERT uses Transformer, that learns contextual relations between words in a text. Transformer is divided in two different mechanism:

- encoder - that reads the text input,
- decoder - that produces a prediction.

The detailed workings of Transformer are described in a paper by Google [38].

Transformer encoder reads the entire sequence of words at once.

BERT and Fine-tuning BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model, for classification tasks like sentiment analysis one only needs to add a layer on top of the Transformer output.

2.15 Ktrain

Ktrain [18] is a lightweight wrapper open-source for the deep learning library Tensorflow Keras, and many pre-trained deep learning architectures like BERT. Ktrain helps to build, train, and deploy neural networks and other machine learning models. It is designed to make deep learning and AI more accessible and easier to apply for both newcomers and experienced practitioners.

For my task, I will use the implementation of pre-trained BERT provided by Ktrain and fine-tune it to classify the sentiment of the reviews.

2.16 Hugging Face

The Hugging Face transformers package is a very popular Python library, that provides thousands of pre-trained models for task text classification, information extraction, question answering, etc. in more than 100 languages. Thanks to Hugging Face NLP has become easy and accessible for everyone. As of 2019, it also supports Tensorflow2 and not just PyTorch [39] as before. More information about Hugging Face [40], the related paper [41] and its source code [42].

2.17 Plotly

Plotly [43] is a company that develops tools for creating data visualizations for data analysis. It is also possible to have these tools online for statistical data for individuals or collaborations. It has a library of different plots [44] and it is also possible to create dashboards thanks to Plotly Dash [45]. Having Python compatible libraries was easy for me to import into my Jupyter Notebooks.

3 Dataset selection

The most complicated step to start with is to search for a dataset, it is the crucial part because a wrong dataset means a wrongly trained model, so searching for the dataset is important and takes a lot of time. Being in fact so important I spent several hours to research, analyze different datasets, to finally find what I needed. Initially for my work I chose the IMDb dataset, but not being ideal I then replaced it with the Hotel Review dataset. The problem with both datasets is that they are in English, and it turned out in the project that the articles to be analyzed would be only in German. I had to search for a dataset in German, and I found the Filmstarts one. Therefore, in the rest of the work, I will continue with the Filmstarts dataset.

3.1 IMDB review description

"Large Movie Review Dataset" [46] is a dataset about movie reviews, consisting of 50k reviews which are divided into positive and negative reviews (no neutral). This is a very famous and used dataset in the world of sentiment classification, other information can be found through the publication "Learning Word Vectors for Sentiment Analysis" [47].

Motivation of the choice

This dataset provides a categorization, i.e., it is a labelled dataset. Another advantage is that with Tensorflow this dataset is particularly good for the methods I can use that we will see later.

3.2 Hotel review description

The dataset consists of approximately 515,000 customer reviews on over 1493 luxury hotels throughout Europe. Each review has a score ranging from 1 to 10. The dataset [48] is hosted on Kaggle and is provided by Jiashen Liu.

Motivation of the choice

The dataset provides an interesting set of datasets for sentiment analysis. These are provided with plenty of attributes. The structure of the datasets is kept very simple and understandable. Furthermore, Kaggle has rated this Dataset with a usability score of 8.2.

3.3 Filmstarts description

The Filmstarts dataset [49] consists of 71,229 user written movie reviews in the German language. The dataset is a collection from the German website "filmstarts.de". The users can label their reviews in the range of 0.5 to 5 stars. With 40,049 documents the majority of the reviews in this data set are positive and only 15,610 reviews are negative.

Motivation of the choice

The choice of this dataset fell on the German language, unfortunately there are not many datasets in this language. Fortunately having a score already made it easier to define a sentiment.

4 Preprocessing the Filmstarts dataset

4.1 Dataframe structure

The dataset is a tab-separated values (TSV) file. A TSV file is a simple text format for storing data in a tabular structure.

First, I need to import the file using Pandas, without importing lines with errors:

```
# Load the data using Pandas
film_de = pd.read_csv("filmstarts.tsv", sep = '\t',encoding='utf8',␣
 ↪error_bad_lines=False, warn_bad_lines=True, header=None)
```

The TSV file contains 3 columns, this means that one customer rating contains 3 attributes. In Table 3 all attributes are explained in detail:

Dataframe structure	
0	Url of the review.
1	Score Rating.
2	Text of the review.

Table 3: Dataframe structure

4.2 Create an input and response dataframe

I cleaned the dataframe to remove all the columns that I did not need. After that I renamed the columns and reordered.

```
film_de = film_de.rename(columns={2: 'Review', 1: 'Score'})
```

```
                                                 Review  Score
10                                     alle teile gemeint    0.0
11      ALSO:    Ich habe in meinem Leben schon viel S...    0.0
55      der Vermarktung!     Ich frage mich allen Erns...    1.0
58      Ey isch hab gestern Lordof the Weed gesehen ne...    0.0
73      Also ich muss ehrlich sagen das ich es total l...    1.0
...                                                   ...    ...
71073  Zwei Stunden Lebenszeit vergeudet. Hier wird v...    0.0
71078  Traumfrauen. Für viele von uns unerreichbar, e...    1.0
71081  So einen drecks film sorry aber mir fällt leid...    0.0

[7744 rows x 2 columns]
```

Now that I have cleaned up the dataframes with what I needed, I see how they are composed, as can be seen from the chart in Figure 4 :

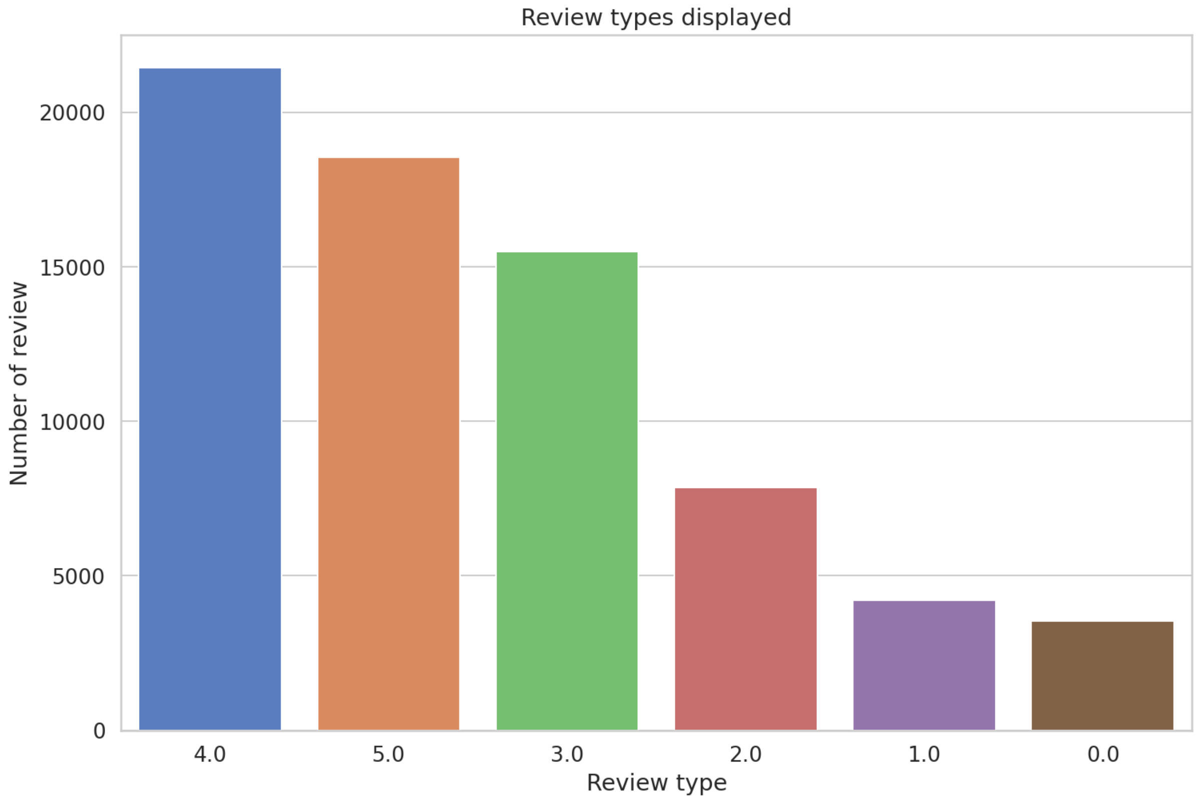


Figure 4: Number of ratings per category

The "Polarity" of a review is determined by the following criteria:

- **"0"**: for the category with score 0.0
- **"1"**: for the category with score 5.0

I did this to have only the worst and best reviews, so that I could have a clear distinction of words. And thus make it easier for the model to work.

The code for this operation is shown here below:

```
# Get review type by aforementioned method
def get_review_type(review_score):
    if review_score <= 0:
        return 0
    elif review_score >= 5:
        return 1
    else:
        return None


film_de["Positive"] = film_de["Score"].apply(
  lambda x: get_review_type(x)
)

# Combine only the useful columns
film_df_de = film_de[["Review", "Positive"]]
```

Now we have the two categories 1 ("good") and 0 ("bad"). As can be seen from the chart in

Figure 5 the "good" category has many more values than the "bad" category.

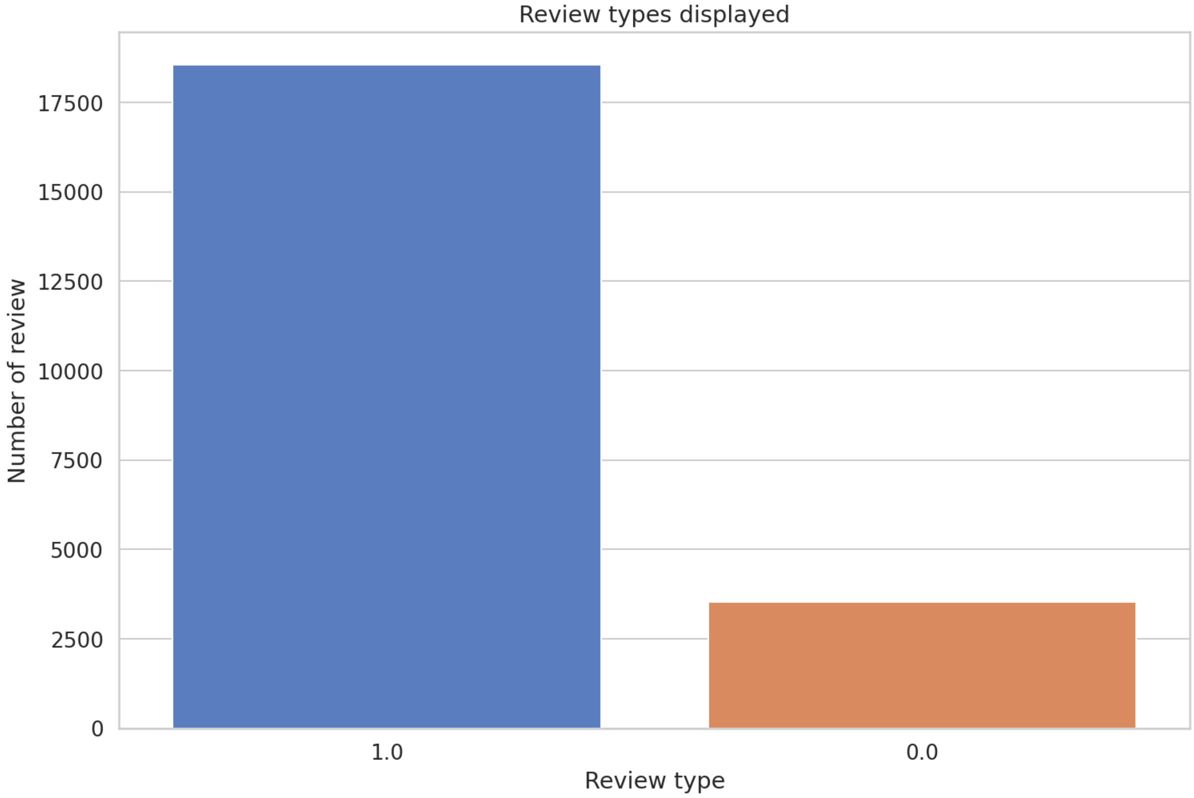


Figure 5: Number of ratings per category

4.3 Resample reviews

In order to be able to train my model later on, I need to have the same amount of test data for each category. For this reason, I should limit the larger category to the value of the smaller one. The code for this operation:

```
# Get same number of reviews for each type
bad_reviews = film_df_de[film_df_de.Positive == 0]
good_reviews = film_df_de[film_df_de.Positive == 1]

sample_len = len(bad_reviews)

bad_df = bad_reviews
good_df = good_reviews.sample(n=sample_len, random_state=RANDOM_SEED)

film_review_df = good_df.append(bad_df).reset_index(drop=True)
film_review_df.shape
```

By doing so, the data will have the same number of entries as in Figure 6.

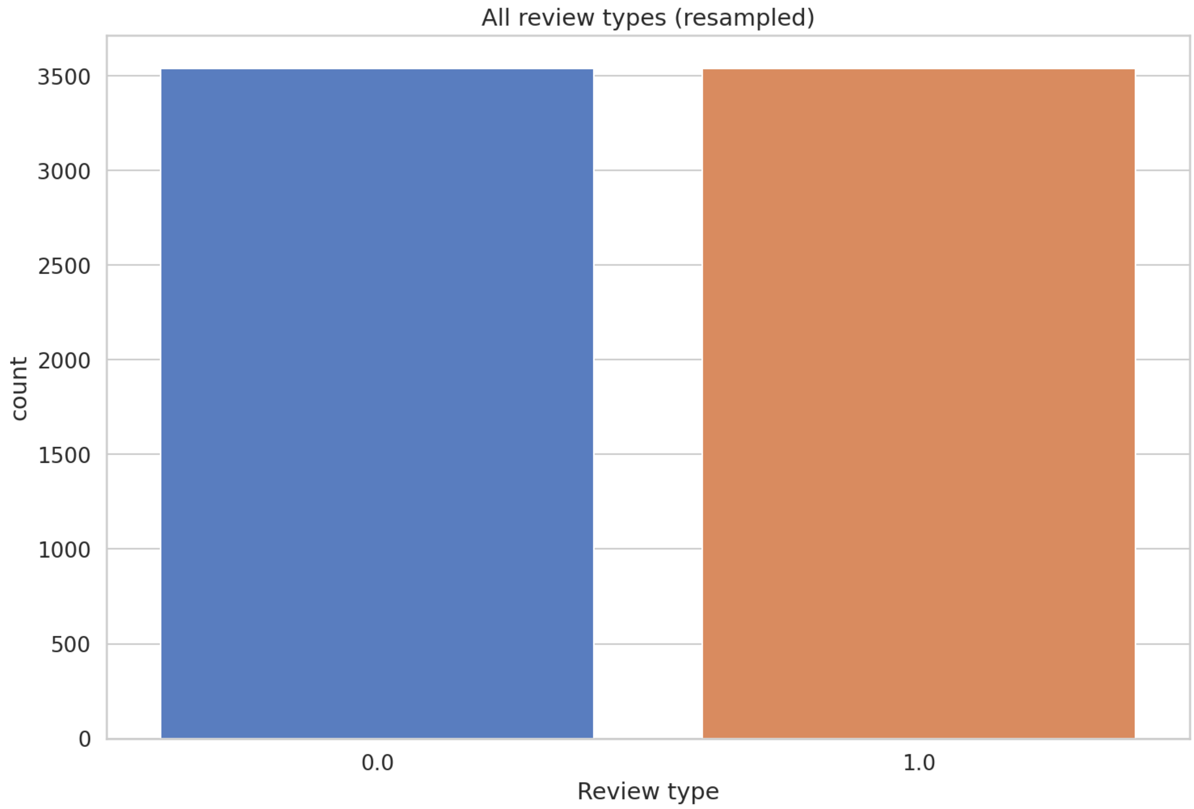


Figure 6: Uniform size of the categories

4.4 Preprocessing

In this section I deal with the preparation of the data to give then to the model. Before I start with splitting the dataset, I randomize the entire dataset by shuffling the data in a totally random way. The first thing to do is to split the dataframes I got after cleaning into 2 parts:

- training set
- test set

Training set is a sample of data used to fit the model, the model trains and learns from this data.
Test set is used to evaluate the trained model. With the evaluation of the model, it is possible to make a fine-tune of the model and to modify the hyperparameters.
Real test data provides the gold standard used to evaluate the model.

Figure 7 shown an example of the split of a dataset[1]:

[1] Reference figure https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

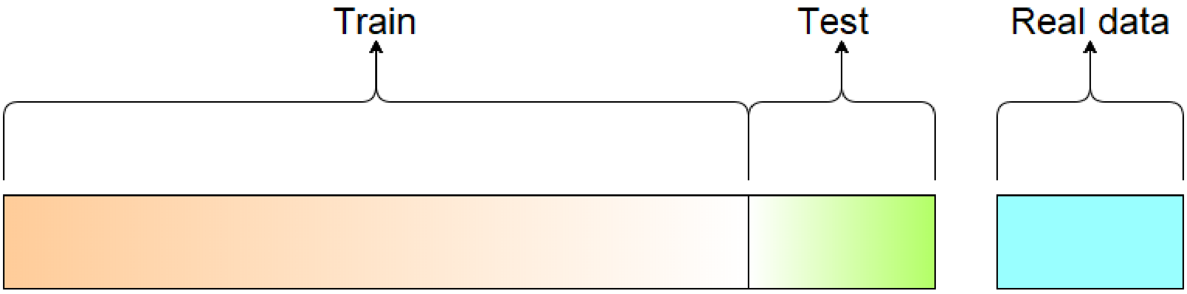


Figure 7: Data split

Thanks to the library sklearn.model_selection.train_test_split I can easily split my dataset in training and test. The division is done with a ratio 80%(training)/20%(test)

```
train, test = train_test_split(film_review_df,test_size=0.2)
```

```
size of training set: 5660
size of test set: 1416
```

Finally, in order to have data that is usable by the Ktrain library, I need to transform both sets into lists, and eliminate rows with null values.

Next I will use the real data from the journals, to do some testing. This test is to see, if with the real data the model behaves as it should.

5 Sentiment Analysis

This is the main section as far as the project is concerned, in fact here will be treated the sentiment analysis. It describes the creation of the model and work on the data predicted by the model. My model can basically be imagined as a function that takes arbitrary text as input, evaluates it, and classifies it into negative and positive. To develop and train a model with this capability requires many prebuilt datasets of real examples from the Internet. These are then used to feed the model so that it can learn from these datasets and deduce correlations. Based on these correlations, the model will then be able to process arbitrary texts and assign sentiment values to them.

5.1 The first approach to the thesis

Before I started doing any work on sentiment analysis, I had to do many hours of research. At the end of the 5th semester, we saw in class how to do text analysis, so with techniques of:

- regex,
- sentence segmentation (character sequence into sentences),
- tokenization (character sequence into tokens (words)),
- stop word elimination (the, a, to, of, etc),
- normalization (U.S.A. or USA as no difference),
- lemmatization,
- stemming,
- using NLTK library.

So, the first approach was to work with an NLTK library, and do text analysis. However, doing research I have seen that in recent times another method has become more famous and has become the state of the art in all respects with regard to the task of text classification, text summarisation, question answering and sentiment analysis. I am talking about BERT. As I continued in my research I saw not only how famous BERT had become in a short time, but also how much it made machine learning work easier, thanks to BERT, Natural Language Processing (NLP) became easy and accessible for everyone. Becoming so used and famous, it caused in a short time the creation of several libraries and methods related to BERT. After working on the datasets, the biggest difficulty was finding the version of BERT and related libraries that would best suit them. For this task, I invested a lot of time and research to figure out which BERT is right for me. Finally, my choice fell on BERT-base with Hugging Face's Transformes as library. The convenience of these transformers is that you can use thousands of pre-trained templates for task text classification, information extraction, question answering, etc. in over 100 languages.

5.2 Ktrain

Now that I am clear on the tools, all I have to do is set up the work. To be able to do something of my own I thought to work also on the dataset, so with Ktrain I did not use the IMDB dataset, but I opted for the Filmstarts dataset.

Ktrain can be used thanks to two different APIs:

- automatic API,
- manual API.

The automatic API has a limited choice of classifiers (6), having to create a model that works on the German language I need a specific classifier, so I am going to use the manual API because I can use any pre-trained model that you can find at the Hugging Face website [40]. The manual model is an improvement of the automatic model, the part about the automatic model can be found in subsection A.4.

Process the data The data has been cleaned previously as seen in section 4, the data then only needs to be tokenized in order to be used.

What is tokenization?[50] Tokenization is one of the most common tasks when it comes to working with text data, in a nutshell is the division of a text (sentence, paragraph or a whole text) into individual words . Each word is called a token. However, BERT cannot use words directly, so there will be a need to encode each token into numbers so that it is usable by the machine. BERT only digests tokens with a maximum length of 512. It does not make sense to use the maximum length when the tokens are shorter, so in order to save model work and to speed up preprocessing work I need to find the maximum length of each review.

What I need to do now is to figure out the maximum length of each review, meaning that I just need to find what is the max_len I am interested in to create the model:

```
textToCheck = film_review_df.Review[1]
```

```
for txt in textToCheck:
    tokens = tokenizer_hugg.encode(txt, max_length=512)
    token_lens.append(len(tokens))
```

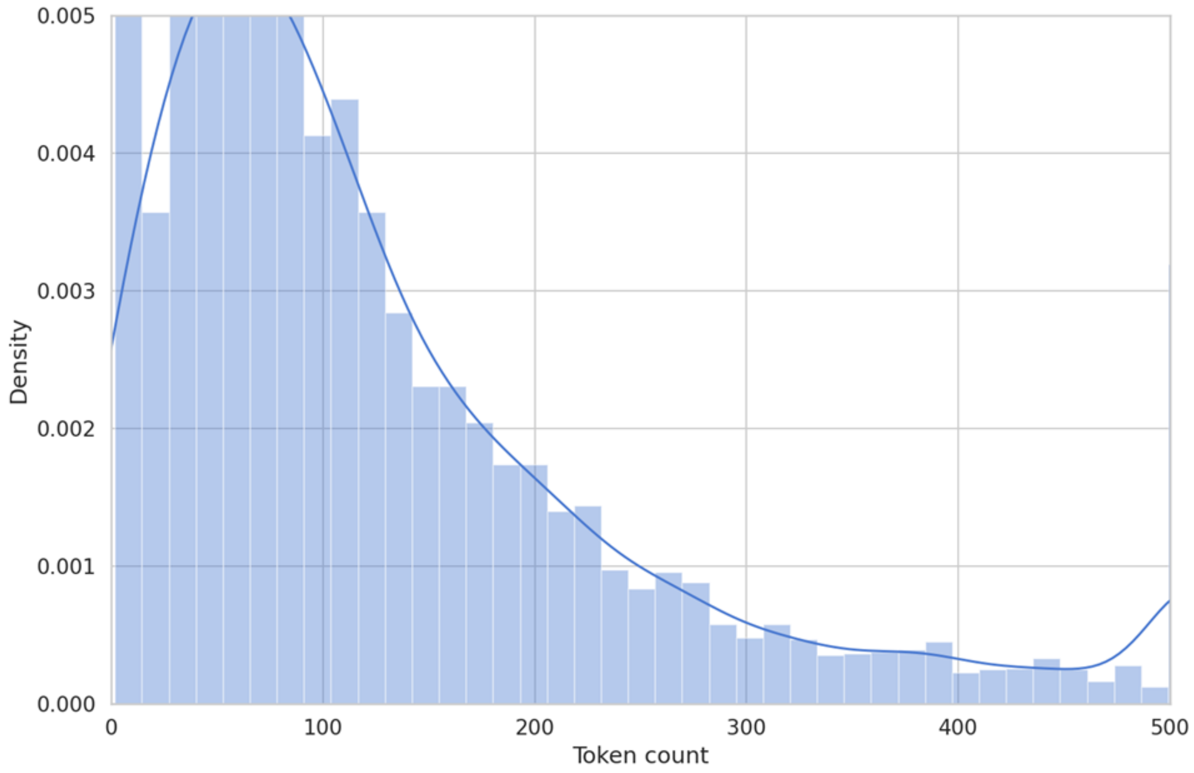


Figure 8: Ktrain learner plot

From Figure 8 it is possible to see that after 400 tokens the density is less than 0.001. For this reason I can set max len to 400.

5.2.1 Build a model and wrap in learner

The section describes the construction of the model using the Ktrain library.

Model My choice fell on dbmz/bert-german-cased. This is one of the best models I found on Hugging Face [51], from the site it is possible to read that:

> The source data for the model consists of a recent Wikipedia dump, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl and News Crawl. This results in a dataset with a size of 16GB and 2,350,234,427 tokens.

As the git name suggests dbmz [52] is available as open-source from the MDZ Digital Library team at the Bavarian State Library [53] based in Munich. Dbmz is one of the leading exponents of the German language and more.

In order to work with Ktrain, I then had to set the model's name:

```
MODEL_NAME = 'dbmdz/bert-base-german-cased'
```

From the Ktrain documentation it is possible to understand how to use the library. The text.Transformer() function allows me to do text classification using a Hugging Face transformers.

```
t = text.Transformer(MODEL_NAME, maxlen=400, class_names=['0','1'])
```

The t.preprocess() function allows me to use the transformer to tokenize and encode the train and test datasets.

```
trn = t.preprocess_train(xtrain_list,ytrain_list)
```

```
preprocessing train...
language: de
train sequence lengths:
        mean : 105
        95percentile : 317
        99percentile : 623

Is Multi-Label? False
```

```
val = t.preprocess_test(xtest_list, ytest_list)
```

```
preprocessing test...
language: de
test sequence lengths:
        mean : 111
        95percentile : 342
        99percentile : 619
```

My model is nothing more than a wrap of all the previous steps, added to the get_classifier() function:

```
model = t.get_classifier()
```

get_learner Now that I have a model, I can create a learner. A learner object will be used to help tune and train my network.

```
learner = ktrain.get_learner(model, train_data=trn, val_data=val, batch_size=12)
```

5.2.2 Model training

learner.lr_find and plot() With a learner object, I can simulate a workout on different learning rates, so I can see the best one, I will use the function:

```
learner.lr_find()
```

With the learner.lr_plot() function I can plot the graph of what I just trained, so can I visually inspect the loss plot to help identify the maximal learning rate associated with falling loss. Figure 9 shows how it works:

```
learner.lr_plot()
```

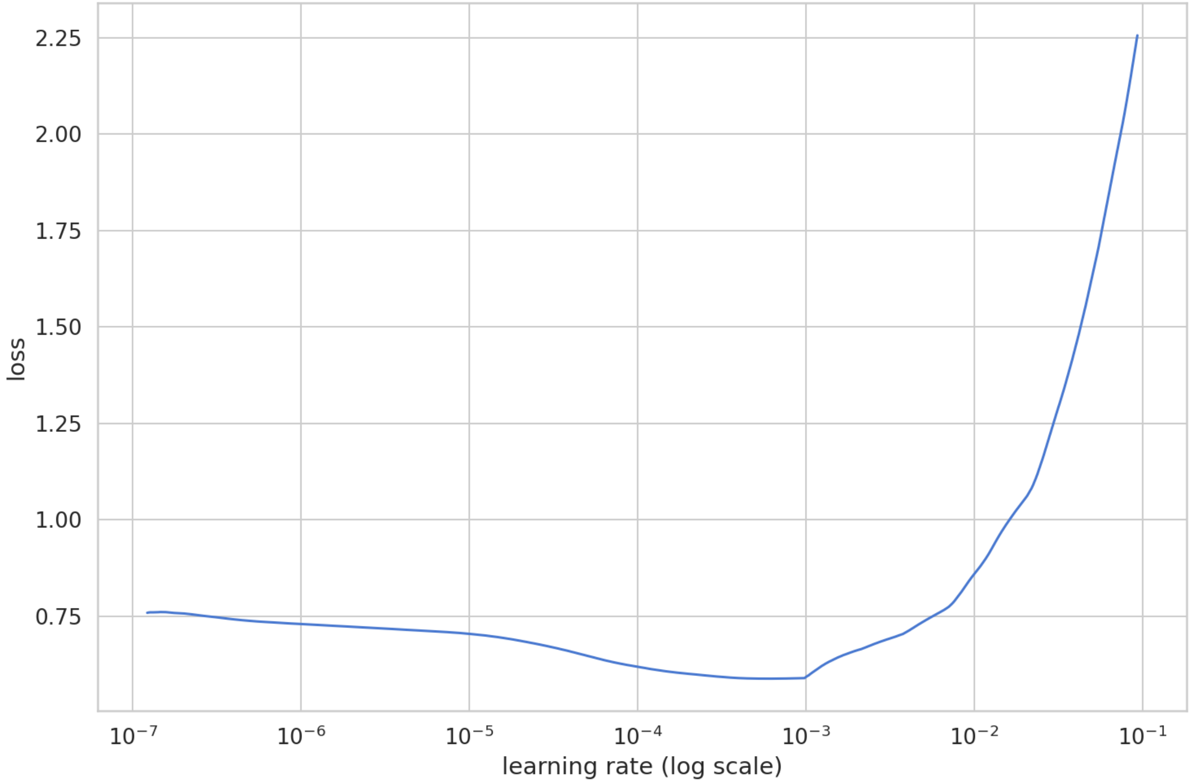


Figure 9: Ktrain learner plot

Autofit Now I am going to use the autofit [54] method to train the model with the parameters I found earlier. The autofit method uses a cyclical learning rate schedule. The default learning rate is the triangular learning rate policy [55]. As before, I invoked the help function to better understand how the autofit function works. The autofit method accepts two primary arguments, learning rate and epochs. If epochs are not defined then the method will train until the validation loss no longer improves after a certain period. This period is also configurable using the early_stopping argument.

I also used another two arguments, reduce_on_plateau and checkpoint_folder.
reduce_on_plateau check if the validation loss fails to improve after a specified number of epochs. checkpoint_folder folder path in which to save the model weights for each epoch.
Next, I called the function as below:

```
learner.autofit(3e-5, reduce_on_plateau=3, checkpoint_folder='./checkpointNewModel25.
↪04/')
```

```
early_stopping automatically enabled at patience=5


begin training using triangular learning rate policy with max lr of 3e-05...
Epoch 1/1024
472/472 [==============================] - 190s 378ms/step - loss: 0.4936 -
accuracy: 0.7200 - val_loss: 0.1846 - val_accuracy: 0.9300
Epoch 2/1024
472/472 [==============================] - 178s 375ms/step - loss: 0.1627 -
accuracy: 0.9432 - val_loss: 0.1748 - val_accuracy: 0.9350
...
```

```
...
Epoch 00005: Reducing Max LR on Plateau: new max lr will be 1.5e-05 (if not
early_stopping).
Epoch 6/1024
472/472 [==============================] - 178s 376ms/step - loss: 0.0299 -
accuracy: 0.9921 - val_loss: 0.2663 - val_accuracy: 0.9307
Epoch 7/1024
472/472 [==============================] - 179s 376ms/step - loss: 0.0178 -
accuracy: 0.9949 - val_loss: 0.2605 - val_accuracy: 0.9314
Restoring model weights from the end of the best epoch.
Epoch 00007: early stopping
Weights from best epoch have been loaded into model.
```

5.2.3 Results

Validate Using the:

```
learner.validate()
```

method I can create a Confusion matrix on the newly trained data in order to get a more detailed picture:

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.94       725
           1       0.93      0.94      0.93       690

    accuracy                           0.93      1415
   macro avg       0.93      0.94      0.93      1415
weighted avg       0.94      0.93      0.93      1415


 array([[14883,  2497],
        [ 2610, 14915]])
```

What is a Confusion matrix?

Confusion matrix is one of the easiest and most intuitive metrics to check the accuracy of a model. It is mostly used for classification problems, as in my example.

The Confusion matrix is calculated as shown in Figure 10:

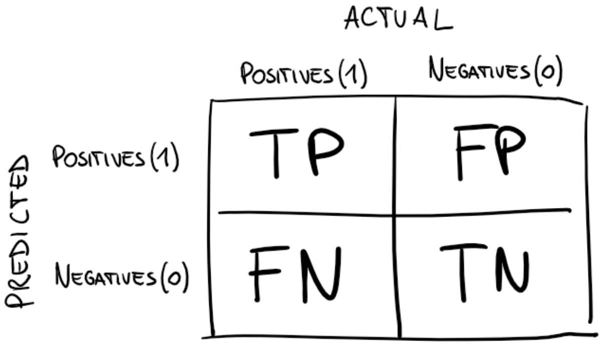


Figure 10: Confusion matrix

In my case I have the two classes 0 (negative) and 1 (positive).

True Positive(TP): number of cases in which the value of the real data is 1 (positive) and the prediction of the model is also 1. Correct prediction.

True Negative(TN): number of the real cases is 0 (negative) and the cases predicted from the model are 0. Correct prediction.

False Positive(FP): number of the real cases are 0 (negative) and the cases predicted from the model are 1 (positive). Wrong prediction.

False Negative(FN): number of the real cases are 1 (positive) and the cases predicted from the model are 0 (negative). Wrong prediction.

Thanks to the Confusion matrix, several factors can be calculated:

- accuracy
- precision
- recall or sensitivity
- F1 score.

Accuracy [56]:
This metric, when talking about tasks such as classification, is the number of correct predictions made by the model, out of all the predictions made. Figure 11 shows how this metric is calculated:

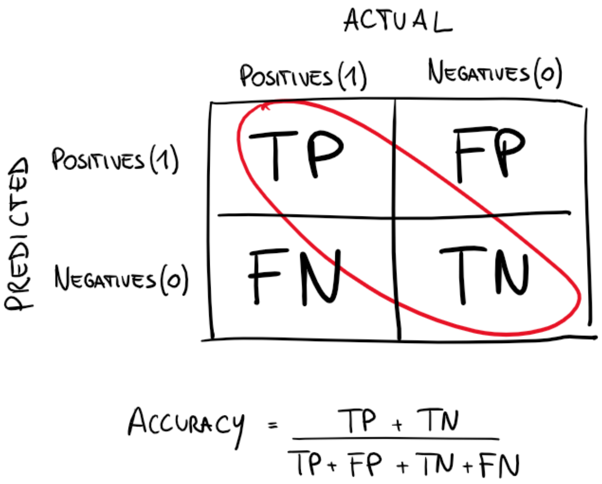


Figure 11: Accuracy

Accuracy is a good measure when the classes of targets in the data are nearly balanced. For example class A= 60% and class B= 40%. If instead I have a big difference like, class A = 85% and class B=15% the accuracy cannot be used because it is not a reliable value.

For this reason I made a resample of my dataset, to have small difference between the two classes. See 4.3.

In my case I have a great accuracy (93%) counting that the model is based on a language where punctuation, cased and uncased words have a great influence, I can be satisfied.

Precision [56]:
Precision is the ratio of correctly predicted positive observations to the total predicted positive

observations. The question that this metric answers is: of all the articles labelled as positive, how many of them are actually positive?

High precision relates to the low false positive (FP) rate. Figure 12 shows how this metric is calculated:

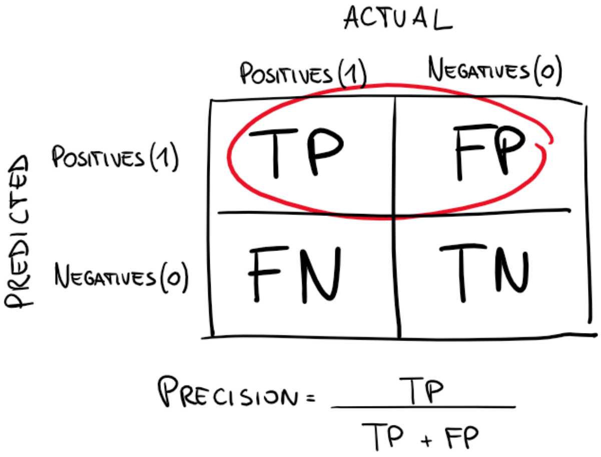


Figure 12: precision

Precision is used in cases where we need to be certain, so it is not the quantity that matters but the quality. For example, if I must make a prediction about cancer cases, I must be 100% sure that it is correct even if it is only one case. In my case it is better to have great accuracy, but if I have mistaken, I do not have repercussions like for example in the medical field.

In my case precision ($0 = 94\%$, $1 = 93\%$) is aligned with all other values.

Recall (or Sensitivity) [56]:
Recall is the ratio of correctly predicted positive observations to all observations in actual class. It answers the question: of all the articles that are truly positive, how many of them were predicted correctly? Figure 13 shows how this metric is calculated:

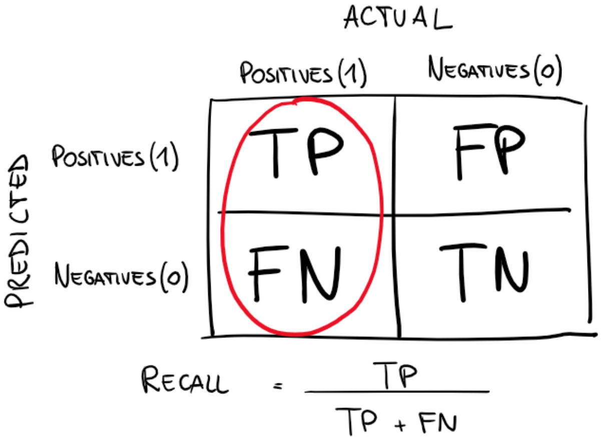


Figure 13: Recall/Sensitivity

Recall is used not so much to figure out which cases were predicted correctly, but more the capture of all "positive sentiment" cases with the answer as "positive".

In my case recall (0 = 93%, 1 = 94%) is aligned with all other values.

F1 score [56]:
We need a metric that takes precision and recall into account, that's why F1 score exists. F1 score is the weighted average of precision and recall. Figure 14 shows how this metric is calculated:

$$\text{F1 SCORE} = \frac{2 \cdot (\text{RECALL} \cdot \text{PRECISION})}{(\text{RECALL} + \text{PRECISION})}$$

Figure 14: F1 score

F1 score is usually more useful than accuracy, especially if we have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it is better to look at both precision and recall.

In my case F1 score (0 = 94%, 1 = 93%) is aligned with all other values.

5.2.4 Save and reload model

Thanks to the last update of Ktrain in March, to save the trained model and its weights I will simply use:

```
predictor.save('./modelsave/bertDe_predictor_93')
```

Even simpler is to load a model, in fact used the function load_predictor() by specifying from which path to take the model, I can already in the next line make predictions.

```
# reload predictor
predictor = ktrain.load_predictor('./modelsave/bertDe_predictor_93')
predictor.predict('Heute ist ein schöner Tag.')
```

5.3 Newspaper API's

Now that I have a trained and working model, what I am missing is data from Swiss newspapers. To do this I used some Application Programming Interfaces (API):

- NewsAPI [57]
- GNewsAPI [58]

I have used both APIs in developer mode for free.

NewsAPI allows me to:

- search all articles and get live top headlines,
- new articles available with 1 hour delay,
- search articles up to a month old,
- 100 requests per day,
- No extra requests available,

- No uptime SLA,
- Basic support.

GnewAPI allows me to:

- 100 requests per day,
- basic support,
- up to 10 articles returned per request,
- maximum of 1 request per second.

The main idea was to use the API in a Python file, so that I have the news of the day every time I launch the file. I used two APIs because I have limitations on the number of requests per day and the number of items per request. To avoid these limitations I imported both APIs in a single Python file, for each API I made a request per category and saved the response in json. Categories are:

- world,
- nation,
- business,
- technology,
- entertainment,
- sports,
- science,
- health.

I transformed the jsons into dataframes, for each dataframe I checked to make sure they were not empty and deleted the columns I did not need. I noticed that NewsAPI returns only newspapers in German language even if I change the settings.

Once I have the different clean dataframes I put them in a list and merge them into one dataframe thanks to the Pandas concat() function. I then check the list that there are no duplicates, and if there are I delete them, always thanks to Pandas with the drop_duplicates() function.

By doing so I expect to have about 30 articles in German for each category.

At this point still using Pandas to_csv, I export and save the dataframe in csv format with its timestamp.

In order to have more freedom I export not only the concatenated dataframe, but also each individual dataframe category.

Now that I have a working Python script what I need is just to make it automatic in order to make one request per day. Using the virtual machine always active, I created, thanks to the windows task scheduler, a task that would start my script once a day and save the files in a certain folder.

5.4 Model test with newspaper data

5.4.1 Ground truth

After a few days of scraping data thanks to my script, I decided to do some tests with the model I had trained. First, I imported the data into my environment, then thanks to Pandas I started to analyze it and then I made the predictions. The dataset I imported has the following columns:

- source
- title
- description

Before I can look at the model results, I need to create a ground truth. In the data science world, it is called "ground truth" when I compare a result created by a user "by hand" with what the model predicted. This is to be able to compare the real data with the data from the model. So, I took a sample of a dataset, and hand classified a positive or negative sentiment for each newspaper article. I started by ranking the sentiment at the description of each article without reading the title first, and then did the same at the title, independent of the description. I did this because some descriptions without reading the title were of one sentiment, but with title attached were of the opposite sentiment. One problem with sentiment prediction I had when the article was "a fact". Normally it can be interpreted as neutral. In that case it can be both, I kept right whether the model predicted positive or negative.

Once I have a ground truth, I tested the model and had it make predictions on the description column, and then on the title column.

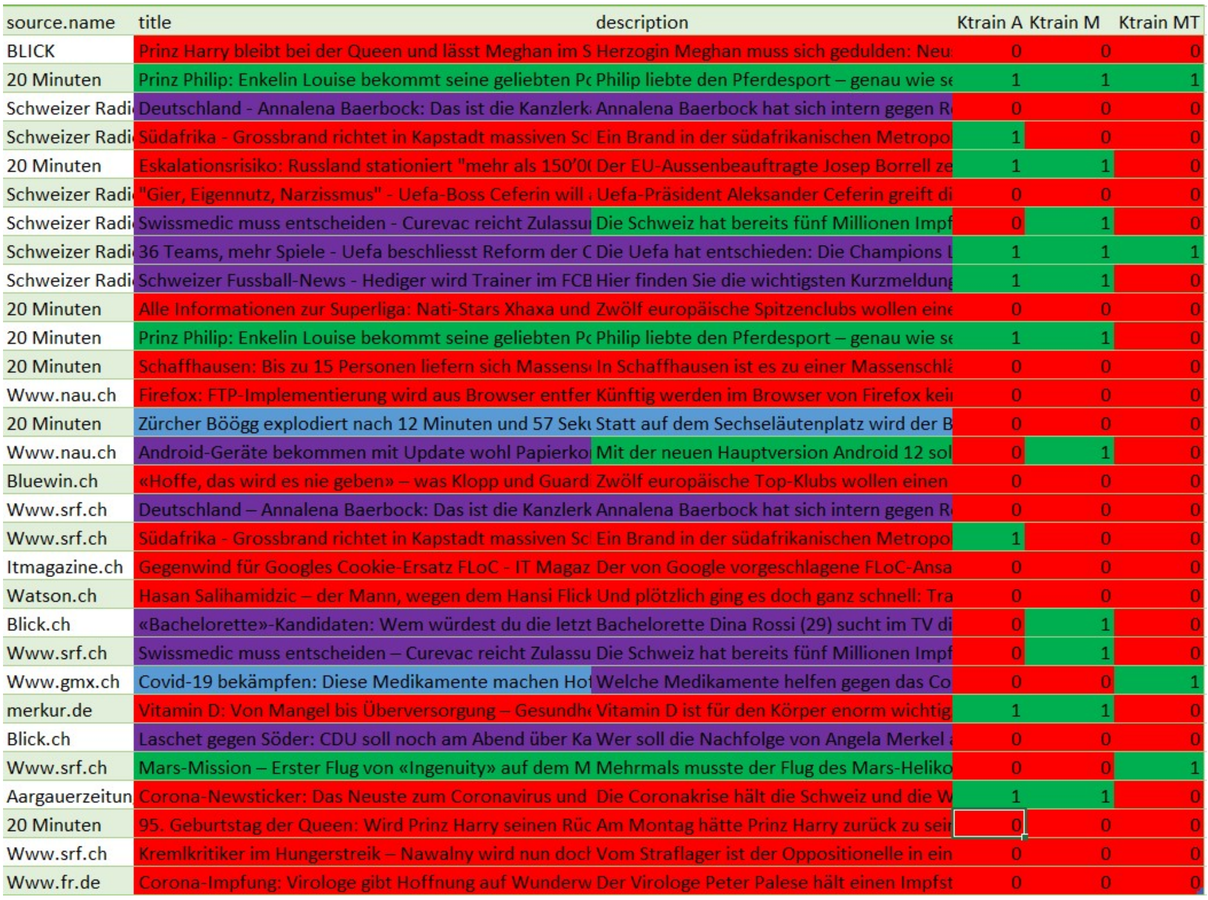
These were the results:

Model	**Accuracy**	**Target**	**Score**
Ktrain Automatic	90%	description	22/30
Ktrain Manual	93%	description	26/30
Ktrain Manual	93%	title	29/30

Table 4: Model comparison

5.4.2 Interesting case studies

In Figure 15 it is possible to see the comparison between the models:



source.name	title	description	Ktrain A	Ktrain M	Ktrain MT
BLICK	Prinz Harry bleibt bei der Queen und lässt Meghan im S	Herzogin Meghan muss sich gedulden: Neu:	0	0	0
20 Minuten	Prinz Philip: Enkelin Louise bekommt seine geliebten Pc	Philip liebte den Pferdesport – genau wie se	1	1	1
Schweizer Radi	Deutschland - Annalena Baerbock: Das ist die Kanzlerk	Annalena Baerbock hat sich intern gegen R	0	0	0
Schweizer Radi	Südafrika - Grossbrand richtet in Kapstadt massiven Sc	Ein Brand in der südafrikanischen Metropo	1	0	0
20 Minuten	Eskalationsrisiko: Russland stationiert "mehr als 150'0(	Der EU-Aussenbeauftragte Josep Borrell ze	1	1	0
Schweizer Radi	"Gier, Eigennutz, Narzissmus" - Uefa-Boss Ceferin will	Uefa-Präsident Aleksander Ceferin greift di	0	0	0
Schweizer Radi	Swissmedic muss entscheiden - Curevac reicht Zulassu	Die Schweiz hat bereits fünf Millionen Impf	0	1	0
Schweizer Radi	36 Teams, mehr Spiele - Uefa beschliesst Reform der C	Die Uefa hat entschieden: Die Champions L	1	1	1
Schweizer Radi	Schweizer Fussball-News - Hediger wird Trainer im FCB	Hier finden Sie die wichtigsten Kurzmeldung	1	1	0
20 Minuten	Alle Informationen zur Superliga: Nati-Stars Xhaxa und	Zwölf europäische Spitzenclubs wollen eine	0	0	0
20 Minuten	Prinz Philip: Enkelin Louise bekommt seine geliebten Pc	Philip liebte den Pferdesport – genau wie se	1	1	0
20 Minuten	Schaffhausen: Bis zu 15 Personen liefern sich Massens	In Schaffhausen ist es zu einer Massenschlä	0	0	0
Www.nau.ch	Firefox: FTP-Implementierung wird aus Browser entfer	Künftig werden im Browser von Firefox kei	0	0	0
20 Minuten	Zürcher Böögg explodiert nach 12 Minuten und 57 Sek	Statt auf dem Sechseläutenplatz wird der B	0	0	0
Www.nau.ch	Android-Geräte bekommen mit Update wohl Papierko	Mit der neuen Hauptversion Android 12 sol	0	1	0
Bluewin.ch	«Hoffe, das wird es nie geben» – was Klopp und Guard	Zwölf europäische Top-Klubs wollen einen	0	0	0
Www.srf.ch	Deutschland – Annalena Baerbock: Das ist die Kanzlerk	Annalena Baerbock hat sich intern gegen R	0	0	0
Www.srf.ch	Südafrika - Grossbrand richtet in Kapstadt massiven Sc	Ein Brand in der südafrikanischen Metropo	1	0	0
Itmagazine.ch	Gegenwind für Googles Cookie-Ersatz FLoC - IT Magaz	Der von Google vorgeschlagene FLoC-Ansa	0	0	0
Watson.ch	Hasan Salihamidzic – der Mann, wegen dem Hansi Flick	Und plötzlich ging es doch ganz schnell: Tra	0	0	0
Blick.ch	«Bachelorette»-Kandidaten: Wem würdest du die letzt	Bachelorette Dina Rossi (29) sucht im TV di	0	1	0
Www.srf.ch	Swissmedic muss entscheiden – Curevac reicht Zulassu	Die Schweiz hat bereits fünf Millionen Impf	0	1	0
Www.gmx.ch	Covid-19 bekämpfen: Diese Medikamente machen Hoi	Welche Medikamente helfen gegen das Co	0	0	1
merkur.de	Vitamin D: Von Mangel bis Überversorgung – Gesundh	Vitamin D ist für den Körper enorm wichtig	1	1	0
Blick.ch	Laschet gegen Söder: CDU soll noch am Abend über Ka	Wer soll die Nachfolge von Angela Merkel	0	0	0
Www.srf.ch	Mars-Mission – Erster Flug von «Ingenuity» auf dem M	Mehrmals musste der Flug des Mars-Heliko	0	0	1
Aargauerzeitun	Corona-Newsticker: Das Neuste zum Coronavirus und	Die Coronakrise hält die Schweiz und die W	1	1	0
20 Minuten	95. Geburtstag der Queen: Wird Prinz Harry seinen Rüc	Am Montag hätte Prinz Harry zurück zu sei	0	0	0
Www.srf.ch	Kremlkritiker im Hungerstreik – Nawalny wird nun doch	Vom Straflager ist der Oppositionelle in ein	0	0	0
Www.fr.de	Corona-Impfung: Virologe gibt Hoffnung auf Wunderw	Der Virologe Peter Palese hält einen Impfst	0	0	0

Figure 15: Model Comparison

The colors in the columns title and description refer to the manual labelling. The red and green colors refer to negative(0) or positive(1), purple instead are the neutral cases where a sentiment cannot be defined, so both values from the model I considered them correct. In blue are in interesting cases, because the text without a context comes across as negative, in fact it is not.

Example: "Statt auf dem Sechseläutenplatz wird der Böögg dieses Jahr im Kanton Uri verbrannt. Der Wind sorgte dafür, dass der Kopf schnell explodierte. Der Sommer kann kommen!"

The model classifies this sentence negative for obvious reasons: something is burned, head exploded, I personally had to inform myself because even I without context was not sure how to classify this news. This shows that the machine is not perfect and cannot understand shades that are difficult even to humans.

5.4.3 Find the bug

What can be seen is that the description alone is often predicted positively compared to the title which is predicted negatively. For this reason the next test sees the Ktrain Manual model doing prediction on "title+description" together, to see if the results improve.

The accuracy of this prediction improved considerably, although I often encountered trivial errors. So what I did was to use predictor.explain() so I could better understand how the model works and I did several tests.

With predictor.explain() I can generate a visualization. The visualization technique is called "Explain like I'm 5" (ELI5)[59] and uses the Local Interpretable Model-agnostic Explanations (LIME) algorithm [60][61], in which the importance of each single word can be seen based on the final prediction, using an interpretable linear model. The green words are inferred as contributing to the classification. The red (or pink) words detract from our final prediction. (Shade of color denotes the strength or size of the coefficients in the inferred linear model.)

I created a dataset on purpose with the newspaper header with more data, to be able to make a good comparison. So, I chose "20 Minuten" to be able to do some tests.

This dataframe has on both "title+description" and "description" 26 negative predictions and 13 positive predictions, while on "title" it has 33 negatives and 6 positives, perfect for understanding what is going on.

Since on both "title+description" and "description" the results match, I focus on the "title" column.

Looking in the dataframe it can be seen that NewsAPI adds at the end of the headline "- newspaper name".

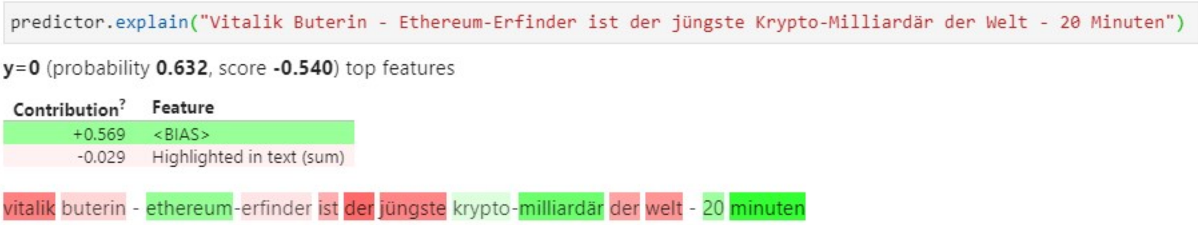
Before making the tests, there are to explain some data that help to understand better how the model works.

- **y=** refers to how the text has been classified, it can be 0 (negative) or 1 (positive),
- **score** is the score that the sentence has obtained looking at the Feature contribution. If the score is negative then y=0 if positive y=1. Higher is the score, higher is the probability value.
- **Contribution** = (weight * feature value) is calculated for each value. For feature values we can have 2 types of features:
  - **Highlighted in text(sum)**,
  - **<BIAS>**

  Both of these values can be positive or negative, green or red. "Highlighted in text(sum)" indicates the correct feature, while "BIAS" incorrect. These two values are summed together, they can have the same or opposite sign.

I did 4 tests on the same sentence with positive ground truth:

1. Title with source at the end = **negative 0.632**



```
predictor.explain("Vitalik Buterin - Ethereum-Erfinder ist der jüngste Krypto-Milliardär der Welt - 20 Minuten")
```

y=0 (probability **0.632**, score **-0.540**) top features

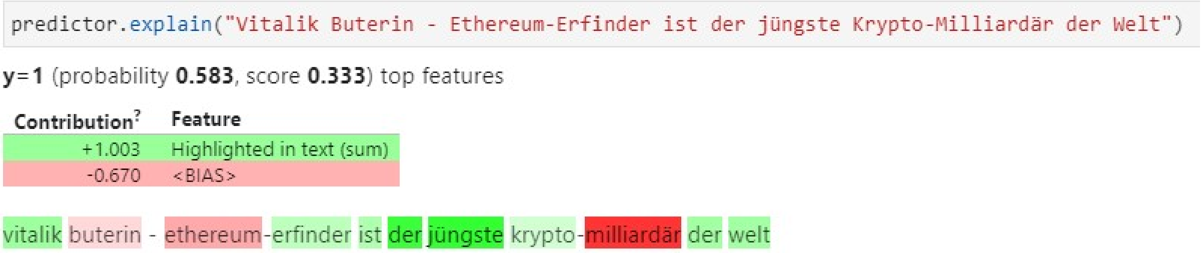
Contribution?	Feature
+0.569	<BIAS>
-0.029	Highlighted in text (sum)

vitalik buterin - ethereum-erfinder ist der jüngste krypto-milliardär der welt - 20 minuten

Figure 16: predictor.explain() test 1

The model classifies it as negative, why? BIAS = +0.569, (incorrect)
Highlighted in text(sum) = -0.029, (correct)
Score will then be = 0.029 - 0.569 = -0.540.
Having negative score, y = 0 with probability = 0.632.

2. Title without source at the end = **positive 0.583**



```
predictor.explain("Vitalik Buterin - Ethereum-Erfinder ist der jüngste Krypto-Milliardär der Welt")
```

y=1 (probability **0.583**, score **0.333**) top features

Contribution?	Feature
+1.003	Highlighted in text (sum)
-0.670	<BIAS>

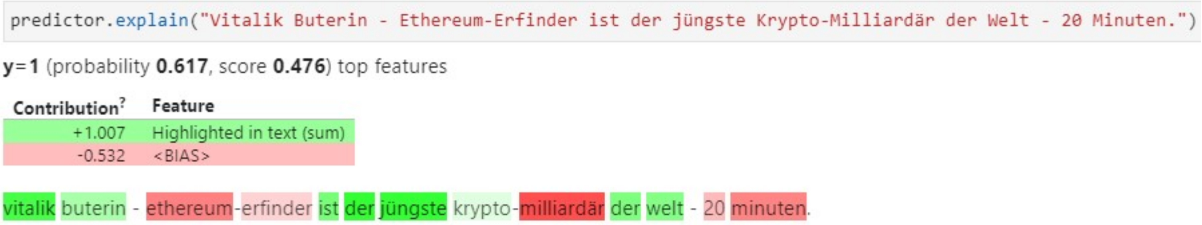
vitalik buterin - ethereum-erfinder ist der jüngste krypto-milliardär der welt

Figure 17: predictor.explain() test 2

Highlighted in text(sum) = +1.003, (correct)
BIAS = -0.670, (incorrect)
Score will then be = 1.003 - 0.670 = 0.333.
Having positive score, y = 1 with probability = 0.583.

I tried adding punctuation:

3. Title with source at the end and no dot = **positive 0.617**



```
predictor.explain("Vitalik Buterin - Ethereum-Erfinder ist der jüngste Krypto-Milliardär der Welt - 20 Minuten.")
```

y=1 (probability **0.617**, score **0.476**) top features

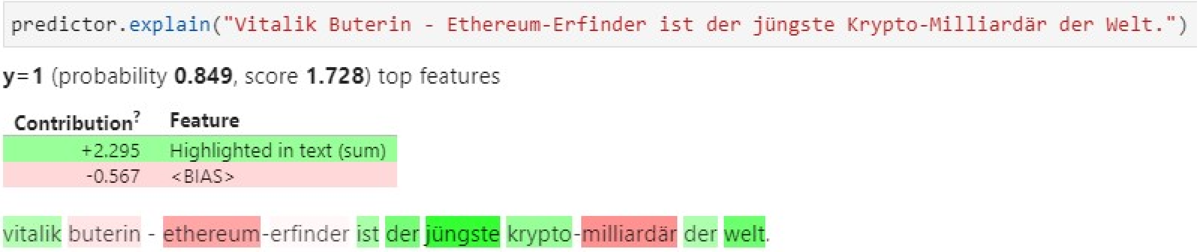
Contribution?	Feature
+1.007	Highlighted in text (sum)
-0.532	<BIAS>

vitalik buterin - ethereum-erfinder ist der jüngste krypto-milliardär der welt - 20 minuten.

Figure 18: predictor.explain() test 3

Highlighted in text(sum) = +1.007, (correct)
BIAS = -0.532, (incorrect)
Score will then be = 1.007 - 0.532 = 0.476.
Having positive score, y = 1 with probability = 0.617.

4. Title without source at the end with dot = **positive 0.849**



```
predictor.explain("Vitalik Buterin - Ethereum-Erfinder ist der jüngste Krypto-Milliardär der Welt.")
```

y=1 (probability 0.849, score 1.728) top features

Contribution?	Feature
+2.295	Highlighted in text (sum)
-0.567	<BIAS>

vitalik buterin - ethereum-erfinder ist der jüngste krypto-milliardär der welt.

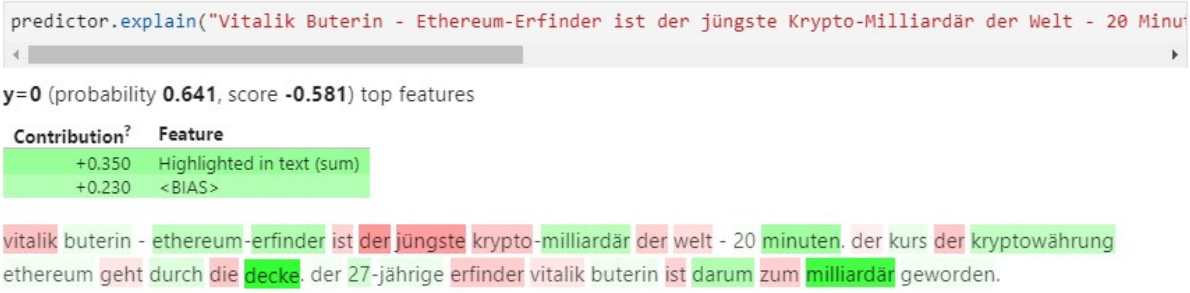
Figure 19: predictor.explain() test 4

Highlighted in text(sum) = +2.295, (correct)
BIAS = -0.567, (incorrect)
Score will then be = 2.295 - 0.567 = 1.728.
Having positive score, y = 1 with probability = 0.849.

It can be understood that punctuation is very important to be able to make predictions correctly, the source already changes the meaning, but the punctuation is at the base.

I then wanted to try "title+description" with what I found previously so:

1. "title+description" with "- source" with dot = **negative 0.641**



```
predictor.explain("Vitalik Buterin - Ethereum-Erfinder ist der jüngste Krypto-Milliardär der Welt - 20 Minu
```

y=0 (probability 0.641, score -0.581) top features

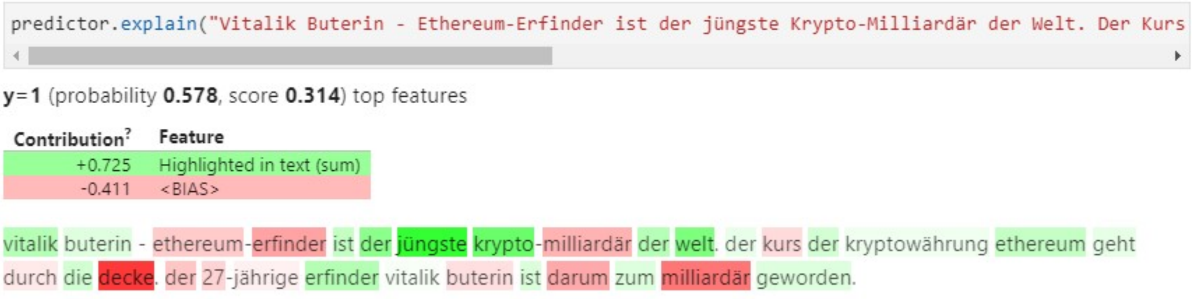
Contribution?	Feature
+0.350	Highlighted in text (sum)
+0.230	<BIAS>

vitalik buterin - ethereum-erfinder ist der jüngste krypto-milliardär der welt - 20 minuten. der kurs der kryptowährung ethereum geht durch die decke. der 27-jährige erfinder vitalik buterin ist darum zum milliardär geworden.

Figure 20: predictor.explain() "title+description" with source

In this case both BIAS and (sum) have the same sign, it means that "Highlighted in text(sum)" goes in the same direction as BIAS, so it is incorrect. Highlighted in text(sum) = +0.230, (incorrect)
BIAS = +0.350, (incorrect)
Score will then be = -0.230 - 0.350 = -0.581.
Having negative score, y = 0 with probability = 0.641.

2. "title+description" without "- source" with dot = **positive 0.578**



```
predictor.explain("Vitalik Buterin - Ethereum-Erfinder ist der jüngste Krypto-Milliardär der Welt. Der Kurs
```

y=1 (probability **0.578**, score **0.314**) top features

Contribution?	Feature
+0.725	Highlighted in text (sum)
-0.411	<BIAS>

vitalik buterin - ethereum-erfinder ist der jüngste krypto-milliardär der welt. der kurs der kryptowährung ethereum geht durch die decke. der 27-jährige erfinder vitalik buterin ist darum zum milliardär geworden.

Figure 21: predictor.explain() ”title+description” without source

Highlighted in text(sum) = +0.725, (correct)
BIAS = -0.411, (incorrect)
Score will then be = 0.725 - 0.411 = 0.314.
Having positive score, y = 1 with probability = 0.578.

Result: Sentence with punctuation totally changes how the sentiment is predicted, while sentence without source increases the accuracy of the predicted result.

5.4.4 Fix the bug

Found this ”bug” there is only to understand how to proceed. The best solution is to clean the data at the root, since NewsAPI gives me unsuitable data, I have two alternatives:

1. do not use NewsAPI,
2. try to solve the problem on data import.

Although the first alternative is the fastest and most immediate, I opted for the second to have a good amount of data on which to make analysis.

I modified the Python script to do data scarping[2]. From the previous version I added having different categories and the timestamp of each article. Next, I created a Python dictionary, filled with all the sources of the various newspapers, so you can clean from each title its source, if the source is not in the dictionary then a warning will notify it. The code to do this:

```python
def cleanup_title(source, title):
    if source not in SOURCES:
        logging.warn("Unknown source %s, leaving as-is", source)
        print("'"+source+"': postfix_title,")
    return SOURCES.get(source, noop)(title)
```

```python
def clean_dataframe(df):
    # clean title from '- newspaper Name'
    title_apply = df.apply(
        lambda row : cleanup_title(row['source.name'], row['title']),
        axis = 1
    )
    # reassign column title
    df['title'] = title_apply
    return df
```

[2] *Link GitHub ”gnewsapi.py”*

I also created a second script to concatenate all files in the same category from all days into one dataframe. This was useful for me to be able to aggregate all the data and make the different plots.

5.5 Sentiment analysis on newspaper data

After managing to import, clean the data and make the prediction, I can finally do the sentiment analysis. The sentiment analysis I will be doing will be on newspaper articles that I have previously imported and concatenated into a single dataframe, thanks to the script[3] I created in Python.

Now I can use my model on all the journal articles I have grouped together over time. I first analyzed the imported data, counting the positive and negative values that the model predicted. By scraping from the API once a day it is very likely to have the same articles on two different days, so to have more accuracy in the prediction I removed all duplicates.

Now I have a dataframe ready to be analyzed with the purpose of creating graphs for a better visual understanding of the data. The first obstacle I found was to normalize the data. This means to transform the positive and negative article count, into a percentage between 0 and 100. To do this I created two functions, the first function is related to the normalization of the entire dataframe:

```python
def normalize(df):
    # copy the data
    df_max_scal = df.copy()

    # apply normalization techniques
    for column in df_max_scal.columns:
        df_max_scal['sentiment %'] = (df_max_scal['count'] / df_max_scal['count'].sum())*100

    df_max_scal['sentiment %'] = df_max_scal['sentiment %'].round(decimals=2)
    return df_max_scal
```

The second is used after a group by:

```python
def norm(x):
    x['sentiment %'] = (x['count'] /x['count'].sum())*100
    x['sentiment %'] = x['sentiment %'].round(decimals=2)
    return x
```

I understood the purpose of normalization through the first plot I made. In fact, the purpose of the plot was to see all the papers with all the positive and negative articles. The problem is that without normalization, I just rely on article count, thus losing focus on the sentiment of the newspaper. After normalization I could then set up the plots.

Below is the list of plots I made:

- plot of sentiment positive vs negative,
- plot of all newspapers positive vs negative

[3] *Link GitHub "Combine_CSV.py"*

- plot top 10 newspapers positive,
- plot top 10 newspapers negative,
- plot per category positive vs negative,
- plot per category of the top 3 newspaper (by top 3 I mean those that have more articles),
- spider plot per category of a single newspaper,
- plot in time per newspaper,
- plot in time per category.

For almost all the plots I used the library Plotly. This library is very famous in the world of data analysis and is widely used to make visualizations.

For each plot I created a single dataframe to leave the original dataframe intact. From the original dataframe, I then used the "group by" function of Pandas. This function allowed me to group in a dataframe based on the values that I needed, then on the grouping calculate the normalization.

The dataset of newspaper articles, which I used to make all the plots, have a time range from the first to the 31th of May 2021.

5.5.1 Plot of sentiment positive vs negative

The plot in Figure 22 describes the percentage of positivity and negativity of all journal articles I obtained.

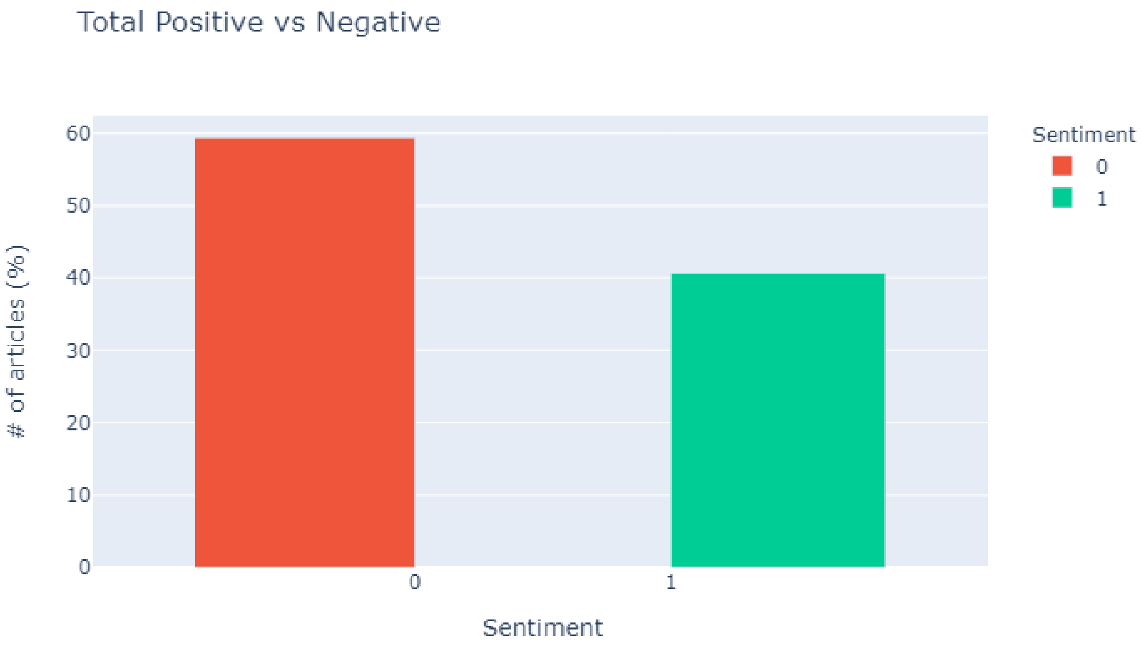


Figure 22: Total positive vs negative

The y-axis describes the percentage of all articles with a positive or negative sentiment, which are located on the x-axis. It is visible that almost 60% of the articles were classified with negative sentiment.

5.5.2 Plot of all newspapers positive vs negative

The plot in Figure 23 describes the positive and negative percentage of all newspapers that have more than 5 articles, I used this filter to get more accurate statistics. Newspapers with a few articles in fact would have biased the analysis.

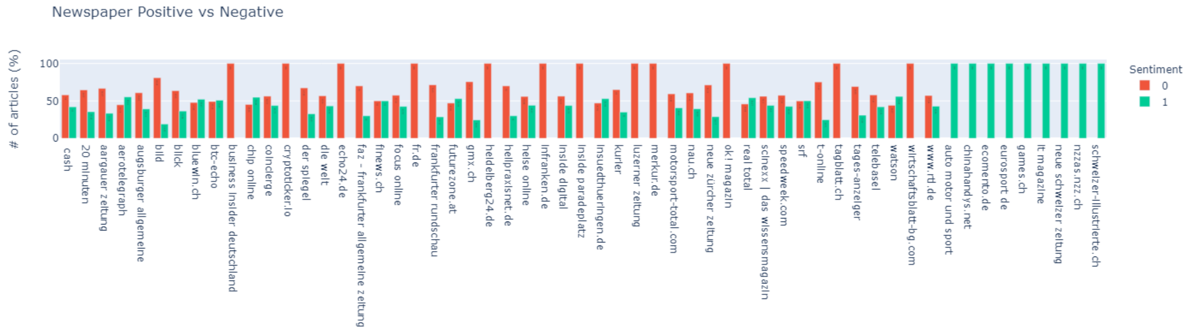


Figure 23: Newspaper sentiment

The y-axis indicates the percentage of positive or negative sentiment, categorized by the sources located on the x-axis.It is interesting to see that some newspapers have a negativity or positivity of 100%. For most cases, however, the negative rate is higher than the positive rate. So not only articles but also most of the newspapers have negative sentiment. With the next two plots it is possible to see more details.

5.5.3 Plot top 10 newspapers positive

The plot in Figure 24 shows the best positive papers. The dataframe from section 5.5.2 was used here, so newspapers below a certain number of articles are not considered.

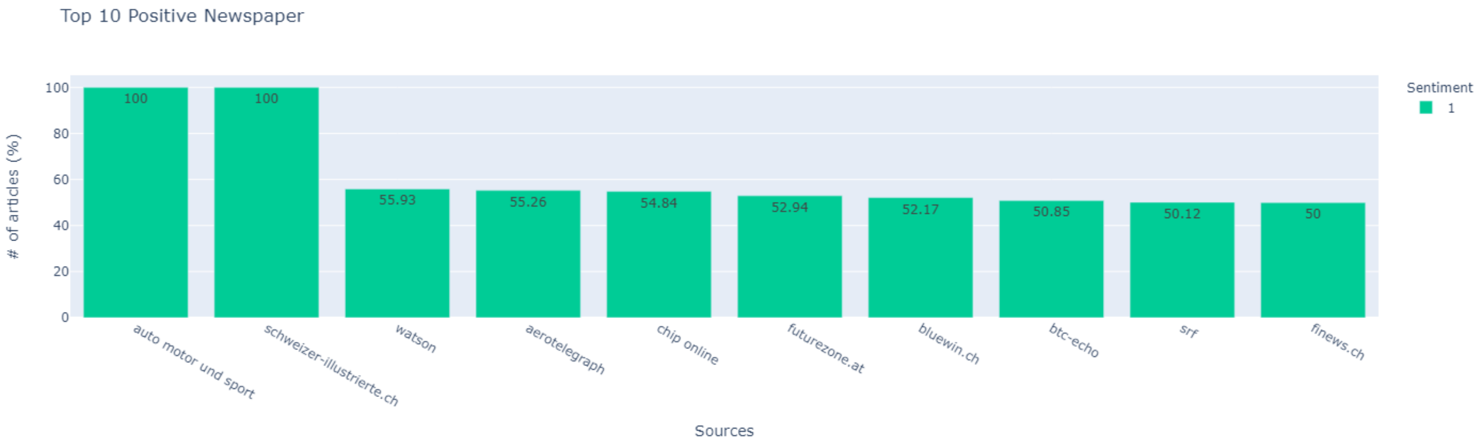


Figure 24: Top 10 positive

The y-axis shows the percentage of all positive items, divided by the sources located on the x-axis.Newspapers with a car, technology or magazine theme have higher positivity.

5.5.4 Plot top 10 newspapers negative

The plot in Figure 25 shows the 10 most negative newspapers. Also here the dataframe from section 5.5.2 was used.

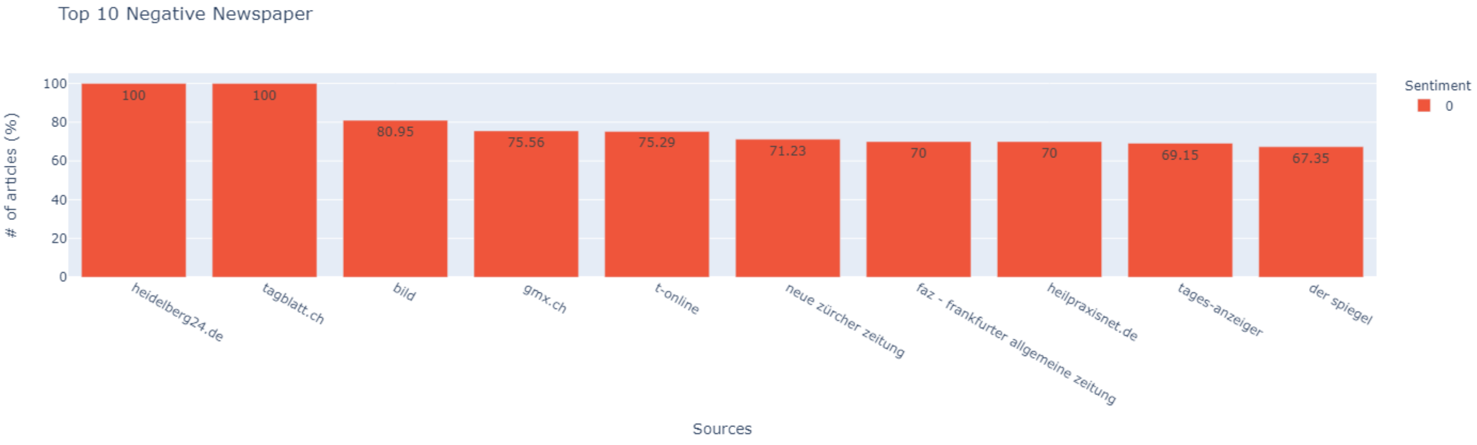


Figure 25: Top 10 negative

The y-axis shows the percentage of all negative items, divided by the sources located on the x-axis.Newspapers with a regional range are the most negative.

5.5.5 Plot per category positive vs negative

The plot in Figure 26 is used to see the percentage of positivity and negativity of each category without considering the newspapers. Thanks to the grouping function I was able to group each individual category and then normalize it.

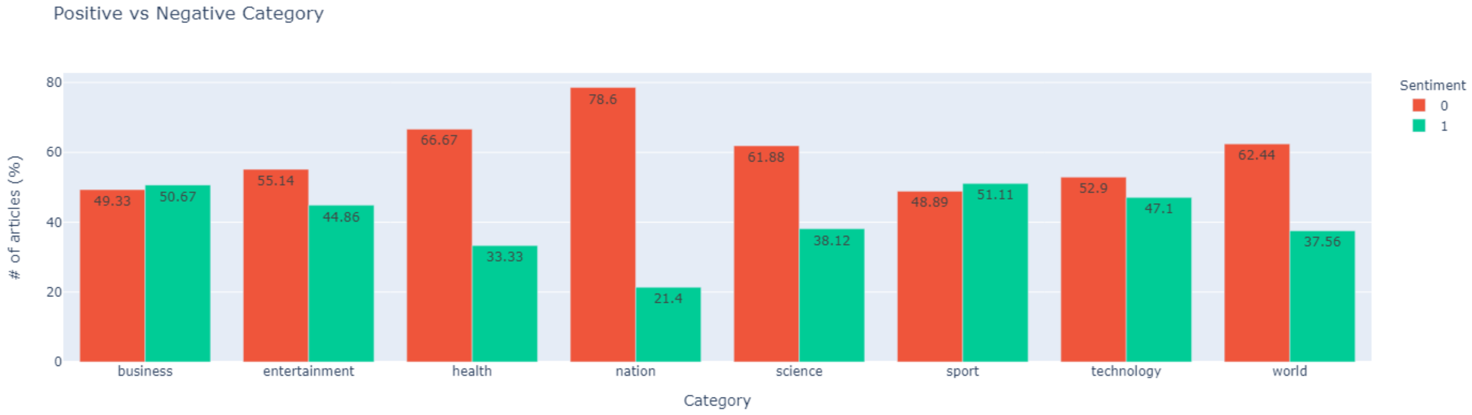


Figure 26: Category positive vs negative

The y-axis indicates the percentage of positive or negative sentiment, sorted by the categories located on the x-axis. The category with the most negatives is "nation". Being German language newspapers, I take into account that nation means not only Switzerland, but also Germany and Austria. Despite this, it is evident the type of negative media assault a person must endure.

5.5.6 Plot per category of the top 3 newspapers

The plot in Figure 27 is used to see the percentage of positivity and negativity of the three largest newspapers by number of articles, all grouped by category.

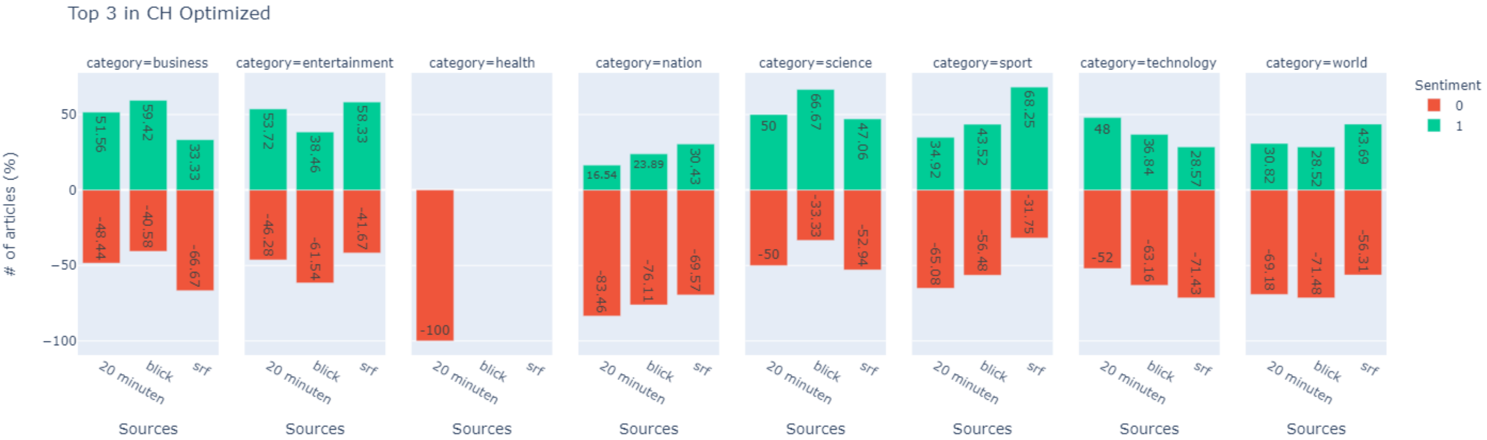


Figure 27: Top 3 newspaper improvement

In detail, "20 Minuten" is the newspaper with the highest negativity in several categories. "Health" category only has 2 articles, so it is not one to be overly considered. While the category with the most negativity is once again "nation". This is a high negativity, considering that this newspaper is free and can be found everywhere.

5.5.7 Spider plot per category of a single newspaper

The plot in Figure 28 gives an idea of the area that each newspaper covers based on the categories. I then created a function to be able to easily plot a list of given newspapers, without having to write everything by hand.

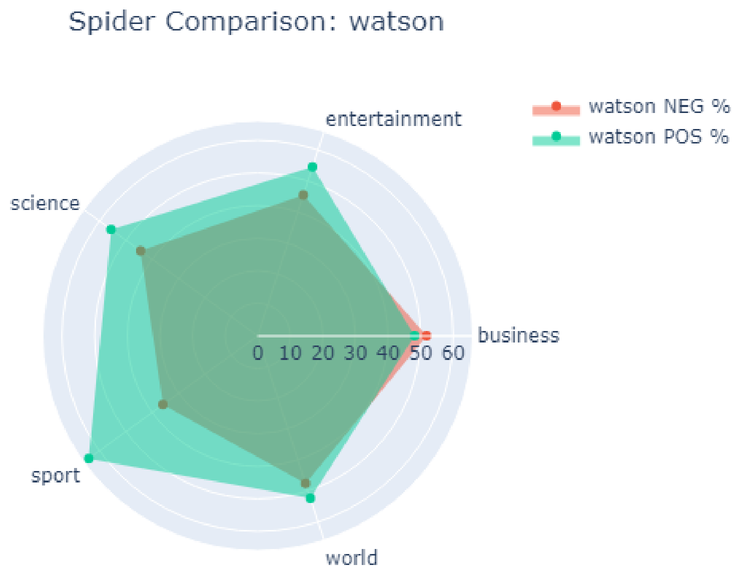


Figure 28: Newspaper spider

Each point on the plot equals the percentage of positivity (in green) or negativity (in red) of a given category. By connecting all the points I define the area of positivity and negativity of each newspaper.

Not just negative newspaper, fortunately. Watson is the newspaper that expresses articles with the most positivity. In fact, in almost every category it has a higher percentage of positivity than negativity.

5.5.8 Plot in time per newspaper

The plot in Figure 30 shows the trend of a newspaper over time. The top 3 newspapers were chosen because they are the ones that influence the most in the Swiss territory. The time span is from May 1, 2021 until today.

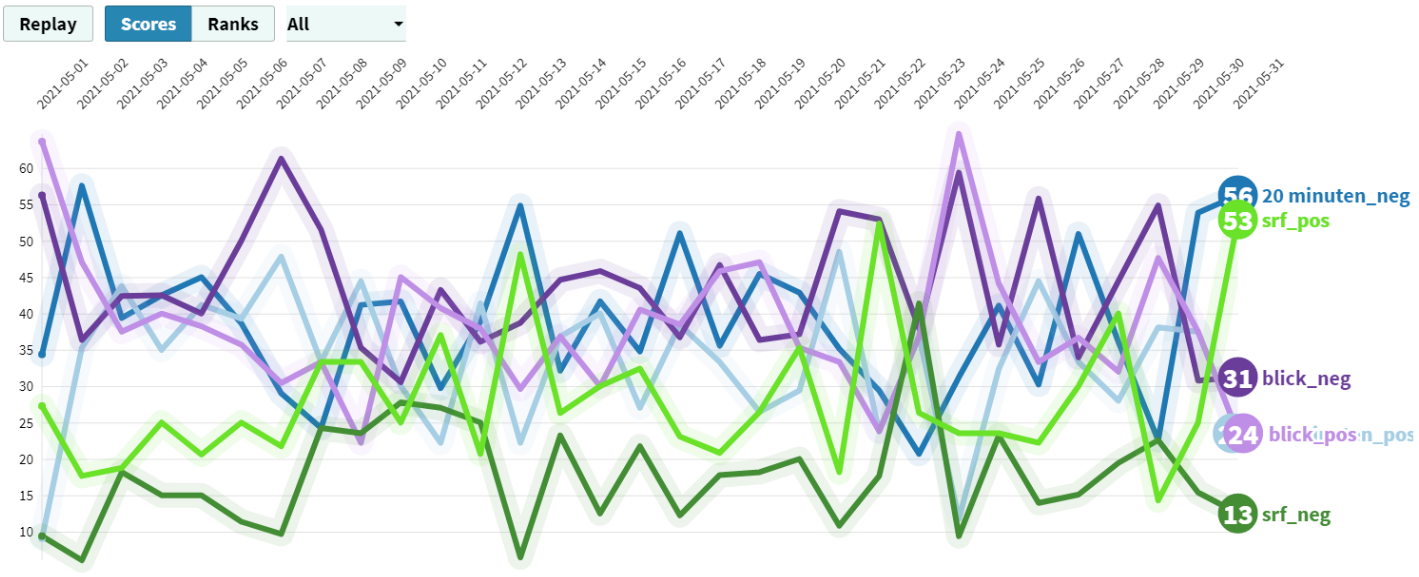


Figure 29: Newspaper in time

The plot presents on the x-axis the time range, and on the y-axis the percentage of positivity or negativity that a paper had over a single day. Each newspaper then is divided into two colors in order to distinguish the positive and negative sentiment. For the division of colors I took into consideration per newspaper two similar colors. The darker color of the two represents negative sentiment, the lighter color represents positive sentiment.

It can be seen that:

- ”blick” has the highest peaks of negative values over time, often exceeding 50%.
- ”srf” remains fairly constant, with more positive articles,
- ”20 minuten” as well as ”blick” have a tendency to be more negative over time.

5.5.9 Plot in time per category

The plot in Figure **??** is an interesting plot because it shows the trend over time as events occur.

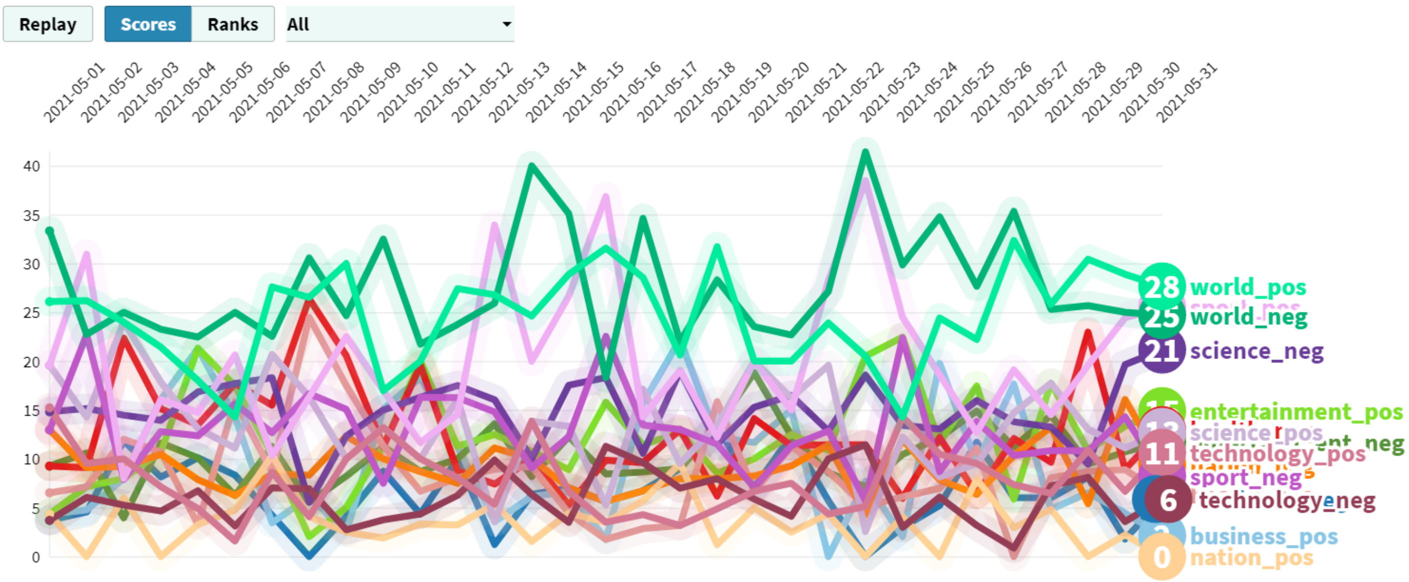


Figure 30: Newspaper in time

The plot presents on the x-axis the time range, and on the y-axis the percentage of positivity or negativity that a category had over a single day. Each category is divided into two colors in order to distinguish positive and negative feeling. For the subdivision of colors I have taken into consideration per category two similar colors. The lighter color of the two represents positive sentiment, the darker color represents negative sentiment.

Categories are:

- business pos(light blue)/neg(blue)
- entertainment pos(light green)/neg(dark green)
- health pos(light red)/neg(red)
- nation pos(light orange)/neg(orange)
- science pos(violet)/neg(lila)
- sport pos(pink)/neg(fuchsia)
- technology pos(light brown)/neg(brown)
- world pos(aqua green)/neg(dark aqua green)

The category that immediately jumps out is the negative "world" one. Recently, negative events have occurred between Palestine and Israel. As of May 17, the category world has had a boost in negative sentiment. On the 23rd of May there was a cableway accident in Italy in which several people died, for this reason also on that day the category "world" had a peak.

It would have been nice to see the performance of the newspapers before and during the pandemic. Being the end of May, maybe I am lucky enough to see the change in direction of the pandemic, like last year in fact in summer it has less effect. Taking into consideration also the vaccination campaign, I speculate that it will be possible to see an improvement in the situation in almost all categories of newspaper.

6 Challenges

In this section, I look at the challenges that emerged during the work on the project and how I covered them.

6.1 Project scaling

As written in the introduction, my partner in the team could not continue this project with me so under the advice of the professor, I decided to scale the project. In doing so, the sentiment analysis went from 5 categories in the previous project to 2.

6.2 Finding the appropriate datasets

My second problem was to find a suitable data set for my needs. To train a model, I needed a data set that was relevant for sentiment analysis, but also large enough to split into training and testing sets without losing model quality. In my opinion, on Kaggle and the internet in general everything can be found.

6.3 Working environment

6.3.1 Virtual machine

My third problem was on the virtual machine, the machine that was given to me unfortunately is not very fitting for the work that must be performed in terms of performance, in fact I have made a request for a more powerful machine. In the time in which I got the new machine, I moved to Colab, a service offered by google, in which a lot of dependencies are already installed.

6.3.2 Google Colab

The first problem I had with Colab is that every time we opened the notebook, I had to load the data, so this implied a big waste of time since the dataset is very large. I also tried to leave it in background on the virtual machine, but Colab after a certain time stops the runtime, so quite useless for my purpose, since to train the model takes several hours. In the end, Anaconda turned out to be the appropriate solution.

6.3.3 Anaconda and Jupyter Notebook

After the difficulties in Colab I decided to go back to the virtual machine, this time more powerful. I created a requirements file from Colab, so I could install all the libraries needed in a new environment. I also tried to create an environment just for the project, and after installing all the dependencies I found errors in the notebook. After several tests on different env's I also had problems installing some libraries. I wanted to avoid working on the base env of anaconda, but it was the only solution that worked. Probably due to system administration privileges. But on the virtual machine I solved all the problems about imports, the dataset remained loaded and the Tensorflow library was much faster.

6.3.4 Tensorflow and Jupyter

Unfortunately, even after troubleshooting Anaconda, I had several problems with Tensorflow in Jupyter including:

- ERROR: tensorflow-gpu 2.3.0 has requirement numpy<1.19.0,>=1.16.0, but you'll have numpy 1.19.5 which is incompatible. I solved it with:
  - https://github.com/tensorflow/models/issues/9200
  - upgrade all the package with: !python -m pip install –upgrade pip
- ERROR: Failed building wheel for pycocotools I solved it with:
  - https://github.com/cocodataset/cocoapi/issues/169
  - by installing Microsoft Visual C++ 14.0
- ERROR: AttributeError: 'float' object has no attribute 'split'. The problem is that some reviews were null,I solved it with:
  - test = test[test['Review'].notnull()]
  - xtest_list = x_test.values.tolist()
  - ytest_list = y_test.values.tolist()

6.3.5 GPU

After setting everything up and starting with my work, I realized that unfortunately the machine upgrade was not enough for my work. In fact, in the virtual machine was present only the CPU but not a GPU, extending considerably the time it took the CPU to train the model. Fortunately, BFH has set up an online environment specifically for machine learning tasks, this env allows students to use jupyter notebooks supported by very powerful GPUs.

6.3.6 Plotly

MLMP uses Jupyter-Lab and not Jupyter-Notebook, this difference is substantial regarding Plotly. In fact, by installing and importing Plotly as I would on a normal Jupyter-Notebook file I would have no problems. The problem I had was that I was unable to plot anything despite the imports. I solved it thanks to these command lines in the MLMP terminal:

1. jupyter labextension list
2. jupyter labextension install jupyterlab-plotly@4.8.2
3. jupyter lab build

In addition, I had to activate the Jupyter-Lab extensions and install:

1. jupyterlab-chart-editor,
2. make a rebuild.

Once these steps are done Plotly is set up correctly and running.

6.4 Finding the appropriate model

This task took me a long time, as for the sentiment analysis problem the field is really huge. After several hours of research, I realized that BERT is the state of the art, as far as this field is concerned. The problem however is that even BERT being a new technology is still not perfect and there is only one way to implement it. Another obstacle is that BERT is a macro-set of other models, so I spent more time looking for the BERT model that was best suited. However, when I found the right model, I had the opposite problem, that is, I had no room for improvement, as it was already perfect. I solved it by taking BERT as a model to compare and creating something of my own even if not perfect.

6.5 NewsApi

NewsApi's dev(free) mode has limitations such as:

- the content of articles is truncated to 200 characters,
- a limited number of requests per day,

So, these two limitations have blocked the work, especially the first one. I wrote an email requesting a student-key just until the project is finished, but unfortunately no response. I solved that I do the same calculation on polarity but not on all the text but on its description, which is a bit the summary of the article. For the issue of the number of requests I solved with a script that I launch once a day.

The API adds at the end of the title "- name of the source", thus throwing off all the results of the model, I solved it by cleaning the title as described in subsection 5.4.

7 Conclusion and future work

7.1 Conclusion

A lot of work and hours were spent creating and optimizing all the processes. These processes include gathering data, extracting details, and display them in a user-friendly way. Right now, a user can get data from the API automatically and concatenate it thanks to a script. After that, all the user has to do is to load the data into a Jupyter Notebook, which will automatically do the sentiment analysis. The plot functions are ready, so the user will be able to plot all the data he imported.

This work has opened my eyes to the world of data. I had never considered sentiment analysis and having it in newspapers made me realize the importance of data in everyday life.

This work could very well be part of studying a psychological thesis on the masses, to see how the polarity of the news can affect the masses. Specifically today, where the news is mostly negative, see the influence it has on the individual person.

Personally, I find it very important how the news is given to us, whether it is true or not, and whether it is possible to improve this situation.

Perhaps through this work, people can realize the negativity of certain news outlets and become aware that the world is not as bad as they portray it.

7.2 Future Work

There is a lot of work that can be tackled in the future. Some ideas are listed in this section, but there is by far more potential.

7.2.1 Back-End

Improve data quality

For the project I used two APIs to get data from the internet, for the future it would be better to do a manual scraping of the data. By doing a manual scraping it is possible to choose many more newspapers, and especially take the ones that are more specific to the area. Besides that, manual scraping, it allows to have much more data from a single newspaper.

Improve accuracy model

The model has an accuracy of 93%, this is because a specific dataset on movie reviews was used. One way to increase the accuracy is to have different datasets, the best option would be to have a dataset on newspaper articles, so that the model already trains on different categories and specific topics, also learning the words better.

Improve technology

Technology is always moving, for this reason it is better to keep updated the model or to create it with the technologies present at the moment of speaking. In fact, just today has been released the new version of Tensorflow with many improvements.

7.2.2 Front-End

User interface

The creation of an application with a user interface, would allow the user to be able to better use this data as a tool.

In addition, the user interface or website should better explain the statistics obtained, making it easier for the user.

A Appendix : Models created

In addition to the documentation, I have described in this section all the models I have created over time. From each model I learned something and realized what I could improve. The models are listed from first to second to last to show the learning journey.

A.1 First model: IMDb dataset

The first model I created was with BERT, without any framework I simply followed the official Tensorflow tutorial. I trained this model the IMDb dataset. This model was purely for educational purposes, once trained the model the only work that actually there was to do was a little fine tuning. I used this model only as an approach to get into the world of BERT, in fact I followed the tutorial of Tensorflow itself to get to this result. For a thesis work I wanted to do something of my own, and not using code and tutorial that it can already find on the net, so I tried something different. After several searches I came across Ktrain.

A.1.1 Lesson learned

I learned how to configure a model with the optimizer, loss and metric.

What is optimizer?
Optimizers are used for improving speed and performance for training a specific model [62]. For my model I chose Adam optimizer. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first order and second-order moments [63].

What is loss?
We use a loss function to determine how far the predicted values deviate from the actual values in the training data. We change the model weights to make the loss minimum, and that is what training is all about [64].

What is a metric?
Calculates how often predictions matches integer labels [65].

A.2 Second model: Ktrain - Hotel dataset

To be able to do something of my own I thought to work also on the dataset, so with Ktrain I did not use the IMDB dataset, but I opted for the Hotel Reviews dataset.

A.2.1 Hotel Review Dataset

Dataframe structure The CSV file contains 17 columns, this means that one customer rating contains 17 attributes. In Table 5 all attributes are explained in detail:

Dataframe structure	
Hotel_Address	Hotel address.
Review_Date	Date on which the customer left his comment.
Average_Score	Average rating. Calculation by all comments of the last year.

Table 5 continued from previous page

Dataframe structure	
Hotel_Name	Hotel name.
Reviewer_Nationality	Client nationality.
Negative_Review	Negative review of the customer. If there is no negative review, it says: ”No Negative”.
ReviewTotalNegativeWord Counts	Number of words in the negative review.
Positive_Review	Positive review of the customer. If there is no positive review, it says: ”No Positive”.
ReviewTotalPositiveWord Counts	Number of words in the positive review.
Reviewer_Score	Score Rating.
TotalNumberofReviews ReviewerHasGiven	Total number of reviews left by the customer.
TotalNumberof_Reviews	Number of reviews of the hotel.
Tags	Tags left by the customer for the review.
dayssincereview	Number of days between the evaluation date and creation of the Dataset.
AdditionalNumberof _Scoring	The number of reviews of the customer, which consist only of a score rating and do not include a comment.
lat	Latitude of the location of the hotel.
lng	Longitude of the location of the hotel.

Table 5: Dataframe structure

A.2.2 Process the data

To do this first I worked on the dataset so:

- cleaned up the columns I did not need,
- divided the dataset in positive and negative thanks to the score table, so for a value below 7 is negative and above 7 is positive,
- resample so that there are the same number of reviews for positive and negative.

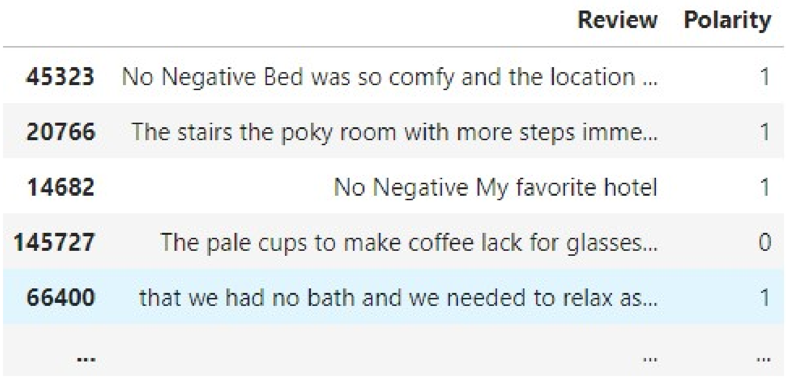
Ktrain provides me with a lot of useful features for my purpose. The first Ktrain function I used is text.texts_from_df. As it is written in the documentation [66], this function allows me to:

"Loads text data from Pandas dataframe file. Class labels are assumed to be one of the following formats:

- one-hot-encoded or multi-hot-encoded arrays representing classes
- labels are in a single column of string or integer values representing class labels
- labels are a single column of numerical values for text regression."

The advantage is that I do not have to do every single preprocessing step manually, and all the steps are followed by the library itself.

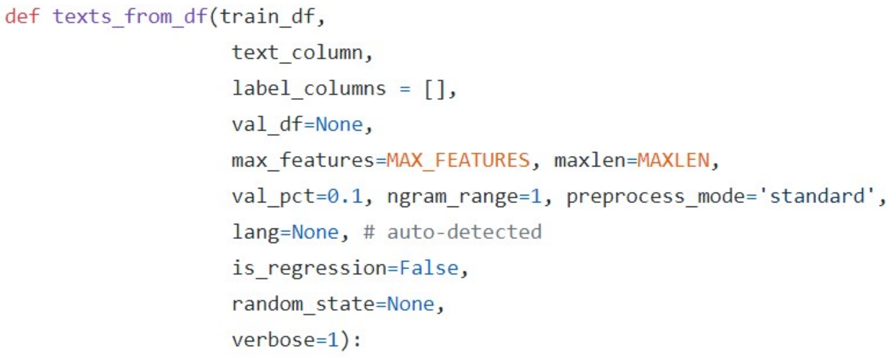
Mine is case number two, in fact my dataframe in Figure 31, is composed as follows:



	Review	Polarity
45323	No Negative Bed was so comfy and the location ...	1
20766	The stairs the poky room with more steps imme...	1
14682	No Negative My favorite hotel	1
145727	The pale cups to make coffee lack for glasses...	0
66400	that we had no bath and we needed to relax as...	1
...	...	...

Figure 31: My resample dataframe

In Figure 32 shown the function from the documentation and the code function I used:



```
def texts_from_df(train_df,
                  text_column,
                  label_columns = [],
                  val_df=None,
                  max_features=MAX_FEATURES, maxlen=MAXLEN,
                  val_pct=0.1, ngram_range=1, preprocess_mode='standard',
                  lang=None, # auto-detected
                  is_regression=False,
                  random_state=None,
                  verbose=1):
```

Figure 32: Ktrain text function from documentation

The function will take as feature the column "Review" and as target "Polarity", it will also automatically download an already pretrained DistilBERT model with its vocabulary. The data for the DistilBERT model has to be preprocessed in a certain way, so I have to indicate this at the preprocess_mode line.

```
train, val, preproc = text.texts_from_df(
train_df=train,
text_column='Review',
label_columns='Polarity',
val_df=test,
maxlen=400,
preprocess_mode='distilbert'
)
```

```
['not_Polarity', 'Polarity']
        not_Polarity  Polarity
128508           1.0       0.0
78547            0.0       1.0
131921           1.0       0.0
16602            0.0       1.0
134008           1.0       0.0
['not_Polarity', 'Polarity']
        not_Polarity  Polarity
45323            0.0       1.0
20766            0.0       1.0
14682            0.0       1.0
145727           1.0       0.0
66400            0.0       1.0
preprocessing train...
language: en
train sequence lengths:
        mean : 39
        95percentile : 120
        99percentile : 227

<IPython.core.display.HTML object>

Is Multi-Label? False
preprocessing test...
language: en
test sequence lengths:
        mean : 40
        95percentile : 123
        99percentile : 225
```

So with this function I load the dataframe that I processed (hotel), I use the Review column for the text to process, while I use the Polarity column to get my target.

A.2.3 Build a Model and Wrap in Learner

Classifier In this section is described the construction of the model, to do this first I had to see what classifier Ktrain allows me to have:

```
text.print_text_classifiers()
```

```
fasttext: a fastText-like model [http://arxiv.org/pdf/1607.01759.pdf]
logreg: logistic regression using a trainable Embedding layer
nbsvm: NBSVM model [http://www.aclweb.org/anthology/P12-2018]
bigru: Bidirectional GRU with pretrained fasttext word vectors
[https://fasttext.cc/docs/en/crawl-vectors.html]
standard_gru: simple 2-layer GRU with randomly initialized embeddings
bert: Bidirectional Encoder Representations from Transformers (BERT) from
keras_bert [https://arxiv.org/abs/1810.04805]
distilbert: distilled, smaller, and faster BERT from Hugging Face transformers
[https://arxiv.org/abs/1910.01108]
```

The classifier I need is distilbert. DistilBERT [67] is a distilled version of BERT, in fact it reduces the size of BERT by 40%, while retaining 97% of its language understanding capabilities and being 60% faster.

So it is smaller and faster implemented by Hugging Face transformer [68]. To be able to use the text.text_classifier function I helped myself with its documentation using some help directly in Jupyter file:

```
help(text.text_classifier)
```

```
Help on function text_classifier in module ktrain.text.models:

text_classifier(name, train_data, preproc=None, multilabel=None,
metrics=['accuracy'], verbose=1)
    Build and return a text classification model.

    Args:
        name (string): one of:
                    - 'fasttext' for FastText model
                    - 'nbsvm' for NBSVM model
                    - 'logreg' for logistic regression using embedding layers
                    - 'bigru' for Bidirectional GRU with pretrained word vectors
                    - 'bert' for BERT Text Classification
                    - 'distilbert' for Hugging Face DistilBert model

        train_data (tuple): a tuple of numpy.ndarrays: (x_train, y_train) or
ktrain.Dataset instance
                        returned from one of the texts_from_* functions
        preproc: a ktrain.text.TextPreprocessor instance.
                 As of v0.8.0, this is required.
        multilabel (bool):  If True, multilabel model will be returned.
                            If false, binary/multiclass model will be returned.
                            If None, multilabel will be inferred from data.
        metrics(list): metrics to use
        verbose (boolean): verbosity of output
    Return:
        model (Model): A Keras Model instance
```

Now that I know how it works, I can build my model so, I chose the classifier (distilbert), train data and preproc are the data I created in the previous function, putting everything together I have as shown in code:

```
model = text.text_classifier('distilbert', train_data=train, preproc=preproc)
```

get_learner Now that I have a model, I can create a learner:

```
learner = ktrain.get_learner(model,
                             train_data=train,
                             val_data=val,
                             batch_size=6)
```

learner.lr_find and plot() Now I simulate a workout on different learning rates and its plot in figure 33, using the functions:

```
learner.lr_find()
```

```
learner.lr_plot()
```

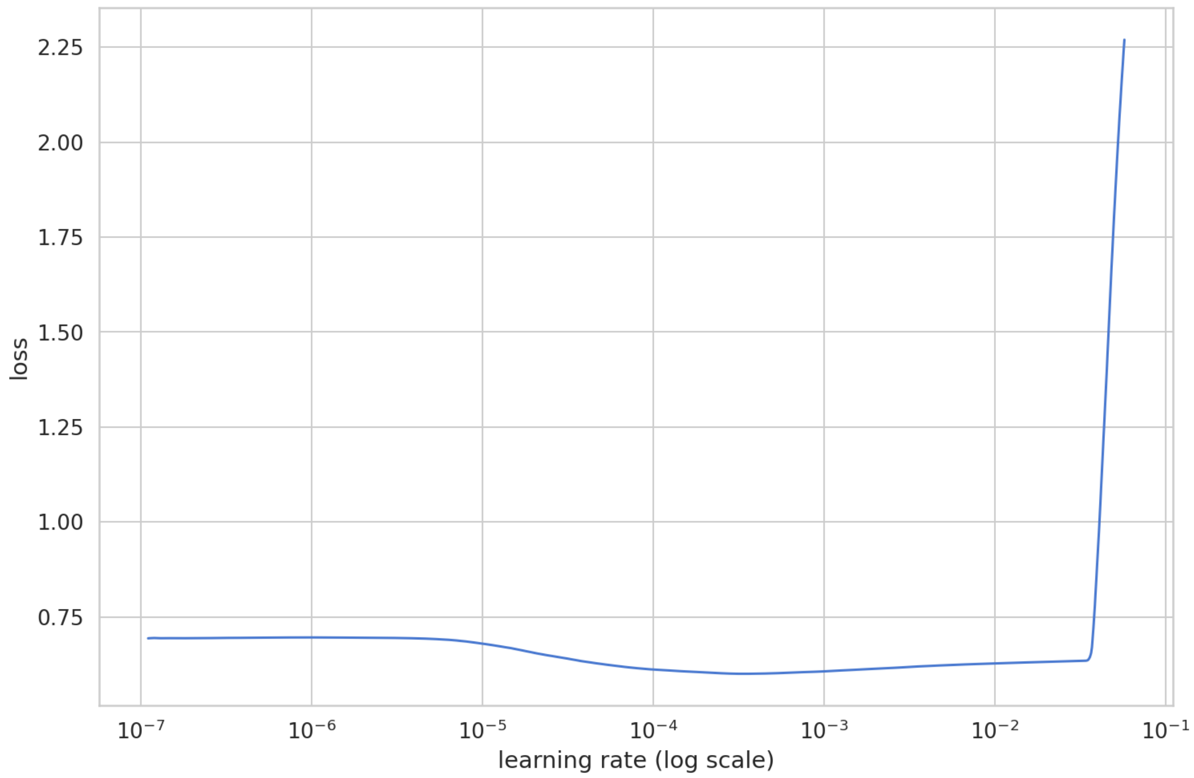


Figure 33: Ktrain learner plot

fit_onecycle The fit_onecycle method is better in speed and accuracy as just the fit method. The fit_onecycle is an implementation of Leslie Smith's 1cycle policy [69]. By using the function:

```
learner.fit_onecycle(lr = 3e-5, epochs=4)
```

```
begin training using onecycle policy with max lr of 3e-05...
Epoch 1/4
23270/23270 [==============================] - 2593s 110ms/step - loss: 0.3926 -
accuracy: 0.8214 - val_loss: 0.3282 - val_accuracy: 0.8553
Epoch 2/4
23270/23270 [==============================] - 2587s 110ms/step - loss: 0.3142 -
accuracy: 0.8632 - val_loss: 0.3245 - val_accuracy: 0.8584
Epoch 3/4
23270/23270 [==============================] - 2587s 110ms/step - loss: 0.2734 -
accuracy: 0.8844 - val_loss: 0.3210 - val_accuracy: 0.8600
Epoch 4/4
23270/23270 [==============================] - 2588s 110ms/step - loss: 0.1708 -
accuracy: 0.9318 - val_loss: 0.4025 - val_accuracy: 0.8537
```

At the 4th epoch I have about 85% accuracy.

validate Using the:

```
learner.validate()
```

method I can create a Confusion matrix on the newly trained data in order to get a more detailed picture:

```
              precision    recall  f1-score   support

           0       0.85      0.86      0.85     17380
           1       0.86      0.85      0.85     17525

    accuracy                           0.85     34905
   macro avg       0.85      0.85      0.85     34905
weighted avg       0.85      0.85      0.85     34905
```

```
array([[14883,  2497],
       [ 2610, 14915]])
```

autofit Not satisfied with this result I tried whit autofit [54]. Next, I called the function as below:

```
learner.autofit(3e-5, reduce_on_plateau=3, checkpoint_folder='./checkpoint/')
```

```
early_stopping automatically enabled at patience=5

begin training using triangular learning rate policy with max lr of 3e-05...
Epoch 1/1024
23270/23270 [==============================] - 2593s 110ms/step - loss: 0.1924 -
accuracy: 0.9222 - val_loss: 0.4034 - val_accuracy: 0.8520
...
Epoch 6/1024
23270/23270 [==============================] - 2588s 110ms/step - loss: 0.0412 -
accuracy: 0.9842 - val_loss: 0.7817 - val_accuracy: 0.8460
Restoring model weights from the end of the best epoch.
Epoch 00006: early stopping
Weights from best epoch have been loaded into model.
```

A.2.4 Discuss the limitations

As can be seen from the autofit code, also in this case I do not exceed 85% accuracy on validation data, having also here a problem of overfitting.

I then wanted to see what the data was where I had the most loss, this is possible due to:learner.view_top_losses(n=1),
with the result: id:11764 | loss:8.19 | true:1 | pred:0).

At this point I saved the model and weights so that I would have a starting point for next time.
I made a list of improvements I want to make in the next test.
In text_from_df function:

- label_columns = must be a list, so I have to modify this argument,
- label_columns = I can try wiht two different target columns positive and negative (so edit the dataframe),
- val_df = none, so the 10% of documents in training dataframe will be used for testing/validation,
- maxlen = from 400 to 500,

- random_state = none, to have train/test split random.

I want also to use the Interactive Training [70].

A.2.5 Lesson learned

The most important of these improvements is missing, which is the work on the dataset. In fact, the 85% accuracy is due to the fact of how I made the split between positive and negative reviews.

A.3 Third model: Ktrain - Hotel dataset optimized

The idea for this phase is to reduce the review window, focusing only on the actual negative reviews and the very positive reviews. The continuation for this phase is as follows:

- resume the original dataset,
- take only the best and worst reviews, based on the ”Review_Score” column,
- make a resample, so that you have the two categories in equal measure,
- and the other improvements described in the previous chapter.

A.3.1 Improvements

After making the improvements described above, I was able to create a model with an accuracy for the English language of 96%.

A.4 Fourth model: Automatic Ktrain - Filmstarts dataset

In Switzerland the greater part of the newspaper articles are in German language, and for this reason the models that I have trained up to this moment are not suitable to the task that they must execute, therefore I have created a new model for the German language on the base of the previous models. Fortunately, the basis I have for creating the model is correct, what I had to do is look for a suitable dataset. For this assignment I chose a dataset on German movies, although it was difficult to find. For training I did exactly the same as for the previous models, so I have:

- clean the dataset,
- I have taken in consideration for the model only the more opportune data,
- trained model thanks to Ktrain.

A.4.1 Major changes

For this model I used as in the previous Ktrain, with these modifications:

- BERT instead of distilBERT for greater accuracy with the German language,
- automatic model,

arriving at an accuracy of 90%.

A.4.2 Discuss the limitations

Now that I have a trained model I tried testing with a file. The file consists of 30 news articles in German.

To do this test I manually labelled each item positive/negative creating a ground true and then compared it to what the model predicted.

What I saw was that the model correctly predicted most articles, with a few issues where the article was a fact or neutral, but this is normal since I only trained for positive and negative. Unfortunately, despite the model arrives at a good percentage (90%) I noticed that some articles where it is clearly negative were predicted positive.

At this point I thought the problem was with the dataset and the model so here are the improvements I want to make in the next one:

- take in consideration only 0 and 5 from the dataset,
- modify Ktrain.

B Attachments

In addition to the documentation, the following documents are available:

- **"Task.pdf"**, the schedule of the project tasks in pdf format,
- **"BachelorTime.pdf"**, the schedule of the project in pdf format.

The code for the entire project, as well as the documentation and scripts, can be reached at the GitHub repository of the bachelor thesis:
`https://github.com/bak89/Sentiment-Analysis-on-Swiss-Newspapers`

Declaration of primary authorship

I hereby confirm that I have written this report independently and without using other sources and resources than those specified in the bibliography. All text passages which were not written by me are marked as quotations and provided with the exact indication of its origin.

Place, Date: Kehrsatz, June 15, 2021

Last Name, First Name: Bakhiet Derias Giorgio

Signature: ....

Index

References

[1] Antony Samuels and John Mcgonical. "News Sentiment Analysis". In: *arXiv:2007.02238 [cs]* (5th July 2020). arXiv: 2007.02238. URL: http://arxiv.org/abs/2007.02238 (visited on 10/04/2021).

[2] *TensorFlow*. Available on: https://www.tensorflow.org/. Accessed: 12.10.2020.

[3] *Keras: the Python deep learning API*. URL: https://keras.io/ (visited on 10/04/2021).

[4] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv:1810.04805 [cs]* (24th May 2019). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (visited on 10/04/2021).

[5] *Keras Tutorial: The Ultimate Beginner's Guide to Deep Learning in Python*. URL: https://elitedatascience.com/keras-tutorial-deep-learning-in-python.

[6] Venelin Valkov. *curiousily/Deep-Learning-For-Hackers*. Available on: https://github.com/curiousily/Deep-Learning-For-Hackers. Accessed: 25.11.2020.

[7] *Sentiment Analysis with TensorFlow 2 and Keras using Python*. Available on: https://curiousily.com/posts/sentiment-analysis-with-tensorflow-2-and-keras-using-python/. Accessed: 12.10.2020.

[8] Paul Karikari. *Deep Learning LSTM for Sentiment Analysis in Tensorflow with Keras API*. Available on: https://medium.com/datadriveninvestor/deep-learning-lstm-for-sentiment-analysis-in-tensorflow-with-keras-api-92e62cde7626. Accessed: 25.12.2020.

[9] Hasan Faraz Khan. *Natural Language Processing and Sentiment Analysis using Tensorflow*. Available on: https://levelup.gitconnected.com/natural-language-processing-and-sentiment-analysis-using-tensorflow-c2948f2623f. Accessed: 05.10.2020.

[10] Real Python. *Sentiment Analysis: First Steps With Python's NLTK Library – Real Python*. en. URL: https://realpython.com/python-nltk-sentiment-analysis/ (visited on 14/03/2021).

[11] Harshita Pandey. *Sentiment Analysis using Deep Learning with Tensorflow*. en. May 2020. URL: https://medium.com/analytics-vidhya/sentiment-analysis-using-deep-learning-with-tensorflow-2bb176c40257 (visited on 14/03/2021).

[12] Real Python. *Practical Text Classification With Python and Keras – Real Python*. en. URL: https://realpython.com/python-keras-text-classification/ (visited on 14/03/2021).

[13] *Classify text with BERT | TensorFlow Core*. en. URL: https://www.tensorflow.org/tutorials/text/classify_text_with_bert (visited on 14/03/2021).

[14] Dipansh Girdhar. *Text Classification with BERT using Transformers for long text inputs*. en. June 2020. URL: https://medium.com/analytics-vidhya/text-classification-with-bert-using-transformers-for-long-text-inputs-f54833994dfd (visited on 14/03/2021).

[15] Thilina Rajapakse. *Simple Transformers — Multi-Class Text Classification with BERT, RoBERTa, XLNet, XLM, and...* en. Nov. 2019. URL: https://medium.com/swlh/simple-transformers-multi-class-text-classification-with-bert-roberta-xlnet-xlm-and-8b585000ce3a (visited on 14/03/2021).

[16] Denis Antyukhov. *Fine-tuning BERT with Keras and tf.Module*. en. Mar. 2020. URL: https://towardsdatascience.com/fine-tuning-bert-with-keras-and-tf-module-ed24ea91cff2 (visited on 14/03/2021).

[17] Sylvain Gugger. *The 1cycle policy*. Another data science student's blog. Section: Experiments. URL: /the-1cycle-policy.html (visited on 20/05/2021).

[18] Arun S. Maiya. *amaiya/ktrain*. original-date: 2019-02-06T17:01:39Z. 9th Apr. 2021. URL: https://github.com/amaiya/ktrain (visited on 10/04/2021).

[19] *Project Jupyter*. URL: https://www.jupyter.org (visited on 10/04/2021).

[20] Alphabet Inc. *Google Colaboratory*. Available on: https://colab.research.google.com/notebooks/intro.ipynb. Accessed: 12.10.2020.

[21] *Kaggle: Your Home for Data Science*. URL: https://www.kaggle.com/ (visited on 10/04/2021).

[22] Usmani Zeeshan-ul-hassan. *What is Kaggle, Why I Participate, What is the Impact? | Data Science and Machine Learning*. Getting Started. 2018. URL: https://www.kaggle.com/getting-started/44916 (visited on 10/04/2021).

[23] Anaconda Inc. *Anaconda | The World's Most Popular Data Science Platform*. Anaconda. URL: https://www.anaconda.com/ (visited on 10/04/2021).

[24] JetBrains s.r.o. *PyCharm: the Python IDE for Professional Developers by JetBrains*. JetBrains. URL: https://www.jetbrains.com/pycharm/ (visited on 10/04/2021).

[25] GitHub Inc. *GitHub: Where the world builds software*. GitHub. URL: https://github.com/ (visited on 10/04/2021).

[26] *Overleaf, Online LaTeX Editor*. URL: https://www.overleaf.com (visited on 10/04/2021).

[27] Berner Fachhochschule. *MLMP*. URL: https://mlmp.ti.bfh.ch/vpn (visited on 10/04/2021).

[28] GIPHY. *Blown Away GIF - Find & Share on GIPHY*. URL: https://media0.giphy.com/media/Y1YNVL8SUm5v5WeN15/giphy.gif?cid=790b7611b589b8521fc262e478a94f529e46a11e6de4dd03&rid=giphy.gif&ct=g (visited on 10/05/2021).

[29] *scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation*. URL: https://scikit-learn.org/stable/ (visited on 10/04/2021).

[30] *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/ (visited on 10/04/2021).

[31] *TensorFlow*. TensorFlow. URL: https://www.tensorflow.org/ (visited on 10/04/2021).

[32] Keras Team. *Keras documentation: Introduction to Keras for Engineers*. URL: https://keras.io/getting_started/intro_to_keras_for_engineers/ (visited on 10/04/2021).

[33] Zhenzhong Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *arXiv:1909.11942 [cs]* (8th Feb. 2020). arXiv: 1909.11942. URL: http://arxiv.org/abs/1909.11942 (visited on 10/04/2021).

[34] Akshay Smit et al. "CheXbert: Combining Automatic Labelers and Expert Annotations for Accurate Radiology Report Labeling Using BERT". In: *arXiv:2004.09167 [cs]* (18th Oct. 2020). arXiv: 2004.09167. URL: http://arxiv.org/abs/2004.09167 (visited on 10/04/2021).

[35] Kevin Clark et al. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *arXiv:2003.10555 [cs]* (23rd Mar. 2020). arXiv: 2003.10555. URL: http://arxiv.org/abs/2003.10555 (visited on 10/04/2021).

[36] Noam Shazeer et al. "Talking-Heads Attention". In: *arXiv:2003.02436 [cs, eess, stat]* (5th Mar. 2020). arXiv: 2003.02436. URL: http://arxiv.org/abs/2003.02436 (visited on 10/04/2021).

[37] Noam Shazeer. "GLU Variants Improve Transformer". In: *arXiv:2002.05202 [cs, stat]* (12th Feb. 2020). arXiv: 2002.05202. URL: http://arxiv.org/abs/2002.05202 (visited on 10/04/2021).

[38] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv:1706.03762 [cs]* (5th Dec. 2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 10/04/2021).

[39] *PyTorch*. URL: https://www.pytorch.org (visited on 10/04/2021).

[40] *Hugging Face – The AI community building the future*. URL: https://huggingface.co/models (visited on 10/05/2021).

[41] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[42] *huggingface/transformers*. original-date: 2018-10-29T13:56:00Z. 10th Apr. 2021. URL: https://github.com/huggingface/transformers (visited on 10/04/2021).

[43] *Plotly: The front end for ML and data science models*. URL: / (visited on 20/05/2021).

[44] *Plotly Python Graphing Library*. URL: https://plotly.com/python/ (visited on 20/05/2021).

[45] *Dash Documentation & User Guide | Plotly*. URL: https://dash.plotly.com/ (visited on 20/05/2021).

[46] *Sentiment Analysis*. URL: https://ai.stanford.edu/~amaas/data/sentiment/ (visited on 10/04/2021).

[47] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[48] Jiashen Liu. *515K Hotel Reviews Data in Europe*. Available on: https://kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe. Accessed: 25.10.2020.

[49] Oliver Guhr. *oliverguhr/german-sentiment*. original-date: 2019-12-02T13:32:51Z. 9th Apr. 2021. URL: https://github.com/oliverguhr/german-sentiment (visited on 10/05/2021).

[50] *What is Tokenization | Methods to Perform Tokenization*. Analytics Vidhya. 18th July 2019. URL: https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization/ (visited on 23/05/2021).

[51] *dbmdz/bert-base-german-uncased · Hugging Face*. URL: https://huggingface.co/dbmdz/bert-base-german-uncased (visited on 10/05/2021).

[52] *Open Source at the Bayerische Staatsbibliothek*. GitHub. URL: https://github.com/dbmdz (visited on 10/05/2021).

[53] *Munich Digitization Center (MDZ) - Homepage*. URL: https://www.digitale-sammlungen.de/en/ (visited on 10/05/2021).

[54] *amaiya/ktrain/autofit*. GitHub. URL: https://github.com/amaiya/ktrain/blob/master/ktrain/core.py (visited on 12/04/2021).

[55] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *arXiv:1506.01186 [cs]* (4th Apr. 2017). arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186 (visited on 12/04/2021).

[56] Jason Brownlee. *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification*. Machine Learning Mastery. 2nd Jan. 2020. URL: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/ (visited on 23/05/2021).

[57] *News API – Search News and Blog Articles on the Web*. News API. URL: https://newsapi.org (visited on 10/05/2021).

[58] *GNews API - An alternative to the Google News API*. URL: https://gnews.io/ (visited on 10/05/2021).

[59] *TextExplainer: debugging black-box text classifiers — ELI5 0.11.0 documentation*. URL: https://eli5.readthedocs.io/en/latest/tutorials/black-box-text-classifiers.html (visited on 23/05/2021).

[60] Marco Tulio Ribeiro, Sameer Singh and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *arXiv:1602.04938 [cs, stat]* (9th Aug. 2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938 (visited on 10/05/2021).

[61] Marco Tulio Correia Ribeiro. *marcotcr/lime*. original-date: 2016-03-15T22:18:10Z. 10th May 2021. URL: https://github.com/marcotcr/lime (visited on 10/05/2021).

[62] *TensorFlow - Optimizers - Tutorialspoint*. URL: https://www.tutorialspoint.com/tensorflow/tensorflow_optimizers.htm (visited on 12/04/2021).

[63] *tf.keras.optimizers.Adam | TensorFlow Core v2.4.1*. TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam (visited on 11/04/2021).

[64] Amaresh Patnaik. *Loss Function in TensorFlow*. Medium. 30th Dec. 2018. URL: https://medium.datadriveninvestor.com/loss-function-in-tensorflow-b7eb1215ef78 (visited on 12/04/2021).

[65] *tf.keras.metrics.SparseCategoricalAccuracy | TensorFlow Core v2.4.1*. TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy (visited on 12/04/2021).

[66] *amaiya/ktrain text*. GitHub. URL: https://github.com/amaiya/ktrain/blob/master/ktrain/text/data.py (visited on 12/04/2021).

[67] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv:1910.01108 [cs]* (29th Feb. 2020). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108 (visited on 12/04/2021).

[68] *DistilBERT*. URL: https://huggingface.co/transformers/model_doc/distilbert.html (visited on 12/04/2021).

[69] Leslie N. Smith. "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay". In: *arXiv:1803.09820 [cs, stat]* (24th Apr. 2018). arXiv: 1803.09820. URL: http://arxiv.org/abs/1803.09820 (visited on 12/04/2021).

[70] *Jupyter Notebook Viewer*. URL: https://nbviewer.jupyter.org/github/amaiya/ktrain/blob/master/tutorials/tutorial-02-tuning-learning-rates.ipynb (visited on 12/04/2021).

List of Figures

List of Tables

Attachments

Summary of tasks Sentimental Analysis Bachelor-Thesis

Author: Giorgio Bakhiet Derias
Tutor: Prof. Dr. Mascha Kurpicz-Briki
Expert: Andreas Dürsteler

List of all the task

1 Project research

This task covers the initial part of the project:

- initial information search
- meeting
- time planning

Next, there will be a constant search for information, as the project is not only programming, but also research.

2 Research dataset

The dataset is one of the key parts of the project, so there will be a search for one or more datasets, appropriate for the job.
The datasets will need to meet these requirements:

- large size
- binary labelled

3 Set up virtual machine

This task covers all the work that needs to be done to set up the virtual machine:

- order a vm
- initialize the machine
- install the necessary software
- check that there are no errors

4 Jupyter Notebook / Repository Setup

For my machine learning project I have chosen a Jupyter notebook as my workspace, for this reason in this task I will be engaged in:

- managing the notebook
- import libraries
- manage the libraries
- correct any errors in the imports

Later will be created a git to have a versioning of my work.

5 BERT model research

State of the art regarding sentiment analysis is the BERT model, in order to use this model, I will need:

- initial research on technology
- search of the adapted model
- various evaluations

6 Work on dataset

Once I have the information I need I can work on the dataset:

- Import dataset
- Extract dataset
- Transforming dataset
- Categorize reviews from dataset
- Present findings from data set in diagrams
- Resample dataset
- Preprocessing Dataset

7 Work on model

Once I am done with the dataset I can deal with the model:

- Create model
- Training model
- Test model
- Tuning model

8 Tests

This task is to do more testing on the goodness of the newly trained model.

9 Documentation

Documentation will be created at the same time so that at the end of the project we will have:

- a documentation of the work with deadline 17.6.2021, 18h00
- file in which will be marked the planned and worked hours with deadline 17.6.2021, 18h00
- a book with deadline 17.06.2021, 18h00
- a short film with deadline 17.06.2021, 18h00
- a poster with deadline 07.06.2021, 12h00

Bachelor work February 2021 until June 2021	Effort in hours		Work weeks																			
			Week 1					Week 2					Week 3					Week 4				
	SOLL	IST	22/02/2021					01/03/2021					08/03/2021					15/03/2021				
Project Tasks					◆ 1																	
Project initialization	1.50	1.75		1.5 1.8					◆ 2			◆ 3										
Research dataset	7.00	7.00			1.0 1.0	2.0 2.0			1.0 1.0	1.0 0.0	1.0 2.0	1.0 1.0										
Set up virtual machine	4.00	7.50			1.0 1.0		1.0 1.0	1.0 1.0				1.0 3.0	0.0 0.0	0.0 1.5			◆ 4					
Jupyter Notebook / Repository Setup	8.50	7.00			0.5 0.5							2.0 2.0	2.0 0.0	2.0 4.5	2.0 0.0							
BERT model research	12.00	11.50				2.0 2.0									1.0 0.0	2.0 2.0	2.0 3.0	1.0 1.0	1.0 1.0	1.0 1.0	1.0 1.5	1.0 0.0
Categorize reviews from dataset	6.50	2.50																2.5 0.0	1.0 0.5	1.0 1.0	1.0 1.0	1.0 0.0
Present findings from data set in diagrams	4.00	2.25																	1.0 0.0	1.0 1.0	1.0 1.0	1.0 0.0
Resample dataset	10.00	5.00																				2.5 2.5
Preprocessing Dataset	8.00	11.25																				
Create model	7.00	16.75																				2.0
Training model	29.00	26.00			2.0 2.0																	
Test Model	24.50	29.00			2.0 2.0																	
Work on Newspaper	40.00	71.50																				
Book, Film, Poster	25.00	13.00																				
Meeting	9.70	9.70	0.7 0.7	0.5 0.5							0.5 0.5								1.0 1.0		0.5 0.5	
Project research	92.75	60.25		0.5 0.5		0.5 3.8	3.5 4.0	2.0 2.0	2.8 2.8		2.0 2.0	2.0 0.5	2.0 2.0	2.0 1.5	0.5 0.5	0.5 0.0	0.5 0.0	0.0 0.0	0.0 1.0	0.0 0.0	0.0 0.0	0.0 3.0
Writing Documentation	71.50	105.80	1.5 2.0	1.5 1.8	0.5 0.5	0.5 0.5		0.5 0.5	0.5 0.0	0.5 0.0	0.5 1.7	0.5 0.0	0.5 0.0	0.5 0.5	0.5 7.5	0.5 0.6	0.5 1.0	0.5 0.5	0.5 0.0	0.5 0.5	0.5 0.5	0.5 0.0
Hours/day soll			2.2	4.0	7.0	5.0	4.5	3.5	4.3	1.5	4.0	6.5	4.5	4.5	4.0	3.0	3.0	4.0	4.5	3.5	4.0	6.0
Hours/day ist			2.7	4.5	7.0	8.3	5.0	3.5	3.8	0.0	6.2	6.5	2.0	8.0	8.0	2.6	4.0	1.5	3.5	3.5	4.5	7.5
Hours week			27.5					20.0					24.6					20.5				

Bachelor work February 2021 until June 2021	Work weeks																			
	Week 5					Week 6					Week 7					Week 8				
	22/03/2021					29/03/2021					05/04/2021					12/04/2021				
Project Tasks																				
Project initialization																				
Research dataset																				
Set up virtual machine																				
Jupyter Notebook / Repository Setup																				
BERT model research																				
Categorize reviews from dataset																0.3				
Present findings from data set in diagrams			◆ 5													0.3				
Resample dataset	2.5 0.0	2.5 0.0	2.5 2.0				◆ 6									0.5				
Preprocessing Dataset				2.5 3.0	1.5 2.0	2.5 4.0	1.5 2.0						◆ 7			0.3				
Create model							1.5 4.0	2.5 2.0	2.5 0.0	2.5 4.0	2.0 2.0	2.5 2.0	2.5 0.0			0.3				
Training model											2.0 6.0	2.5 2.0	2.5 0.0	2.5 0.0	2.5 0.0	2.5 3.0	2.5 0.0	2.5 0.0	2.5 0.0	2.5 0.0
Test Model																				
Work on Newspaper																				
Book, Film, Poster																				
Meeting		0.5 0.5										0.5 0.5					0.5 0.5			
Project research	1.5 0.0	1.5 1.5	1.5 0.0	1.5 0.0	1.5 2.0	1.5 0.0	1.5 2.0	1.5 1.0	1.5 0.0	1.5 1.0	0.5 0.0	1.5 0.5	1.5 4.0	1.5 1.5	1.5 2.0	1.5 1.5	1.5 2.0	1.5 3.0	1.5 3.8	1.5 2.5
Writing Documentation	0.5 0.0	0.5 0.0	0.5 0.0	0.5 0.0	0.5 2.0	0.5 2.0	0.5 1.5	0.5 1.5	0.5 5.0	0.5 3.0	0.5 1.0	0.5 0.5	0.5 0.0	0.5 3.5	0.5 4.0	0.5 2.0	0.5 2.8	0.5 0.0	0.5 1.0	0.5 1.0
Hours/day soll	4.5	5.0	4.5	4.5	3.5	4.5	5.0	4.5	4.5	4.5	5.0	7.5	7.0	4.5	4.5	4.5	5.0	4.5	4.5	4.5
Hours/day ist	0.0	2.0	2.0	3.0	6.0	6.0	9.5	4.5	5.0	8.0	9.0	5.5	4.0	5.0	6.0	8.0	5.3	3.0	4.8	3.5
Hours week	13.0					33.0					29.5					24.5				

Project Tasks Balance	
Total SOLL in hours	**360.95**
Total IST in hours	**387.75**
Difference in hours	**26.80**

Project goals (milestones)

1. Modify the previous dataset and model, than training and testing
2. Inital project research
3. Find dataset
4. Set up development environment
5. Dataset Exploration
6. Dataset Preprocessing
7. Create model
8. Model training
9. Test model
10. Work on Newspaper
11. Book, Film, Poster
12. Export and Deploy

Legend	
Time estimate in hours (SOLL)	🟩
Effective time in hours (IST)	🟦
Project objective (milestone)	◆

Bachelor work February 2021 until June 2021	Effort in hours		Work weeks														
			Week 9					Week 10					Week 11				
	SOLL	IST	19/04/2021					26/04/2021					03/05/2021				
Project Tasks																	
Project initialization	1.50	1.75															
Research dataset	7.00	7.00															
Set up virtual machine	4.00	7.50															
Jupyter Notebook / Repository Setup	8.50	7.00															
BERT model research	12.00	11.50															
Categorize reviews from dataset	6.50	2.50															
Present findings from data set in diagrams	4.00	2.25															
Resample dataset	10.00	5.00															
Preprocessing Dataset	8.00	11.25															
Create model	7.00	16.75	0.5				◆ 8										
Training model	29.00	26.00	2.5 6.5	1.5	2.0		3.0										
Test Model	24.50	29.00	2.5 0.0	2.5 0.0	2.5 7.0	2.5 1.5	2.5 4.0	2.5 1.5	2.5 1.0	2.5 8.0	2.5 4.0		◆ 9				
Work on Newspaper	40.00	71.50										2.5 8.5	2.5 6.0	2.5 8.5	2.5 9.0	2.5 0.0	2.5 0.0
Book, Film, Poster	25.00	13.00															
Meeting	9.70	9.70		0.5 0.5					0.5 0.8								
Project research	92.75	60.25	1.5 1.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 1.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0	1.5 0.0
Writing Documentation	71.50	105.80	0.5 1.5	0.5 0.0	0.5 0.0	0.5 0.0	0.5 0.0	0.5 3.0	0.5 4.5	0.5 0.0	0.5 0.0	0.5 2.5	0.5 0.0	0.5 0.0	0.5 0.0	0.5 4.5	0.5 4.0
Hours/day soll			7.0	5.0	4.5	4.5	4.5	4.5	5.0	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5
Hours/day ist			9.5	2.0	9.0	1.5	7.0	4.5	7.3	8.0	4.0	11.0	6.0	8.5	9.0	4.5	4.0
Hours week			29.0					34.8					32.0				

Bachelor work	Work weeks														
	Week 12					Week 13					Week 14				
	10/05/2021					17/05/2021					24/05/2021				
Project Tasks															
Project initialization															
Research dataset															
Set up virtual machine															
Jupyter Notebook / Repository Setup															
BERT model research															
Categorize reviews from dataset															
Present findings from data set in diagrams															
Resample dataset															
Preprocessing Dataset															
Create model															
Training model															
Test Model										◆ 10					
Work on Newspaper	2.5 0.0	2.5 7.5	2.5 11.0	2.5 5.5	2.5 8.0	2.5 6.5	2.5 1.0	2.5 0.0	2.5 0.0	2.5 0.0					
Book, Film, Poster											1.0 0.0	1.0 1.5	1.0 0.0	1.0 0.0	1.0 1.5
Meeting		0.5 0.5					0.5 0.5					0.5 0.5		1.0 1.0	
Project research	2.0 2.0	2.0 0.0	2.0 0.0	2.0 1.5	2.0 0.0	2.0 0.0	2.0 0.0	2.0 0.0	2.0 0.0	2.0 1.5	0.5 1.5	0.5 0.0	0.5 0.0	0.5 0.0	0.5 0.0
Writing Documentation	0.5 4.0	0.5 0.0	0.5 0.0	0.5 0.0	0.5 0.0	2.0 2.0	2.0 3.0	2.0 3.0	2.0 2.0	2.0 6.5	2.0 0.0	2.0 4.5	2.0 5.0	2.0 3.0	2.0 0.0
Hours/day soll	5.0	5.5	5.0	5.0	5.0	6.5	7.0	6.5	6.5	6.5	3.5	4.0	3.5	4.5	3.5
Hours/day ist	6.0	8.0	11.0	7.0	8.0	8.5	4.5	3.0	2.0	8.0	1.5	6.5	5.0	4.0	1.5
Hours week	40.0					26.0					18.5				

Bachelor work	Work weeks														
	Week 15					Week 16					Week 17				
	31/05/2021					07/06/2021					14/06/2021				
Project Tasks															
Project initialization															
Research dataset															
Set up virtual machine															
Jupyter Notebook / Repository Setup															
BERT model research															
Categorize reviews from dataset															
Present findings from data set in diagrams															
Resample dataset															
Preprocessing Dataset															
Create model															
Training model															
Test Model															
Work on Newspaper										◆ 11					
Book, Film, Poster	2.0 0.0	2.0 1.0	2.0 2.0	2.0 0.0	2.0 0.0	2.0 0.0	2.0 0.0	2.0 1.0	2.0 3.0	2.0 3.0		◆ 12			
Meeting		0.5 0.3					0.5 0.5					0.5 0.5			
Project research															
Writing Documentation	2.0 0.0	2.0 0.0	2.0 1.5	2.0 2.0	2.0 0.0	2.0 0.0	2.0 0.5	2.0 0.0	2.0 0.0	2.0 0.0					
Hours/day soll	4.0	4.5	4.0	4.0	4.0	4.0	4.5	4.0	4.0	4.0	0.0	0.5	0.0	0.0	0.0
Hours/day ist	0.0	1.3	3.5	2.0	0.0	0.0	1.0	1.0	3.0	3.0	0.0	0.5	0.0	0.0	0.0
Hours week	6.8					8.0					0.5				

Project Tasks Balance

Total SOLL in hours	**360.95**
Total IST in hours	**387.75**
Difference in hours	**26.80**

Project goals (milestones)

1. Modify the previous dataset and model, than training and testing
2. Inital project research
3. Find dataset
4. Set up development environment
5. Dataset Exploration
6. Dataset Preprocessing
7. Create model
8. Model training
9. Test model
10. Work on Newspaper
11. Book, Film, Poster
12. Export and Deploy

Legend

Time estimate in hours (SOLL)	🟩
Effective time in hours (IST)	🟦
Project objective (milestone)	◆