Natural Language Processing - Mini-Challenge 2

Sentiment Analysis

Daniel Perruchoud & Joe Weibel, Institut für Data Science

1 Introduction

Text classification is a common task in NLP. One particular case, the sentiment analysis identifies the mood or tone of written texts. Sentences may express positive, negative, and neutral sentiment, in practice we also find applications with positive vs negative sentiment, and the task may also be formulated as sentiment regression.

While data is typically available in abundance, creating labels manually for supervised learning is typically a costly, time-consuming task motivating the interest for alternatives to automatically generate annotations via semi-supervised learning (weak supervision), i.e., leveraging a smaller set of manual annotations to generate a much larger set of synthetic labels.

However, semi-supervised learning introduces uncertainties, moreover, when multiple individuals assess texts, there may be discrepancies in judgments (annotator disagreement).

2 Content & solution approach

2.1 Content and learning objectives

You and your team are tasked to develop and evaluate models to classify texts by sentiment, i.e., the mood or tone of written texts. Here we will be using data with positive, negative, and neutral categories. You will inspect

- the number of labels which need to be manually annotated,
- the level of annotator agreement required and
- the configuration appropriate for a semi-supervised learning approach.

All five NPR learning objectives are included in this mini-challenge which offers many opportunities to understand central concepts in text-based NLP.

- LO1 - Preparation and representation of text data: ingesting and filtering textual data, constructing a hierarchically nested data split, creating weak labels for unlabeled texts
- LO2 - Statistical and neural language models: understanding, applying and evaluating pre-trained masked LMs to generate sentence-level text embeddings
- LO3 Transformer-based model algorithms: understanding transformed-based models used for sentiment classification

- LO4 - Learning methods for NLP: understanding and applying semi-supervised learning, transfer learning and fine-tuning for sentiment classification; developing and evaluating impact of different weak labelling strategies
- LO5 - NLP Tools & Frameworks: building a classification-evaluation pipeline, setting up and running systematic experiments

The goal of this mini-challenge is to go beyond mere engineering and implementation and understand the inner workings.

2.1.1 Model setup

Get to know dataset used in this mini-challenge and provide basic statistical characteristics. Then, choose two pretrained **general purpose** Transformer-based language models from Hugging Face (e.g., BERT-base and ModernBERT-base) for the sentiment classification, and suitable performance metrics.

Prepare your data to train and evaluate the classification models using varying amounts of training data, i.e. construct a hierarchically nested data split to simulate different amounts of available labels. This means that differently sized training datasets have to be grown so that larger datasets contain all observations of preceeding smaller datasets, the largest data set holding around 1000 records. Also think about how to split your data, so that you can analyze a scenario with low vs high interannotator agreement.

Note, that you will need either a separate validation dataset or use cross-validation for evaluation.

2.1.2 Baseline classification model

Use your two pre-trained classification models as a baseline and then fine-tune your models on all hard labelled data first. Select the best model based on performance and repeat fine-tuning for each sample size of the hierachically nested data partitions. Find suitable model hyperparameters and evaluate the model's performance considering training data size (see Figure 1).

You may apply post-training quantization to reduce memory consumption. Building a pipeline for training the classification model is recommended, as you might want to outsource it to a computation cluster such as CSCS.

Bonus task: in addition you may also consider inspecting the classification performance of an LLM with In-Context Learning and appropriate prompting techniques.

2.1.3 Text embeddings

Weak labelling (see below) is a semi-supervised learning technique to overcome the limitation of costly manual annotation by leveraging text similarities. But how? First, texts need to be represented in numeric form as embedding vectors generated by means of a language model. Then, similarities between embedding vectors can be quantified and a strategy can be developed to retrieve similar labelled texts for a given unlabelled text.

Specific model architectures such as sentence-transformers have been trained especially for semantic representation of entire sentences. Apply at least two models for generating text embeddings and explore the embeddings quantitatively and qualitatively. To inspect the embeddings' quality, you calculate and visualize similarity-based statistics without dimensionality reduction techniques.

Bonus task: visualize and analyze text embeddings with a dimension reduction technique with appropriate hyperparameters, you might identify interesting patterns.

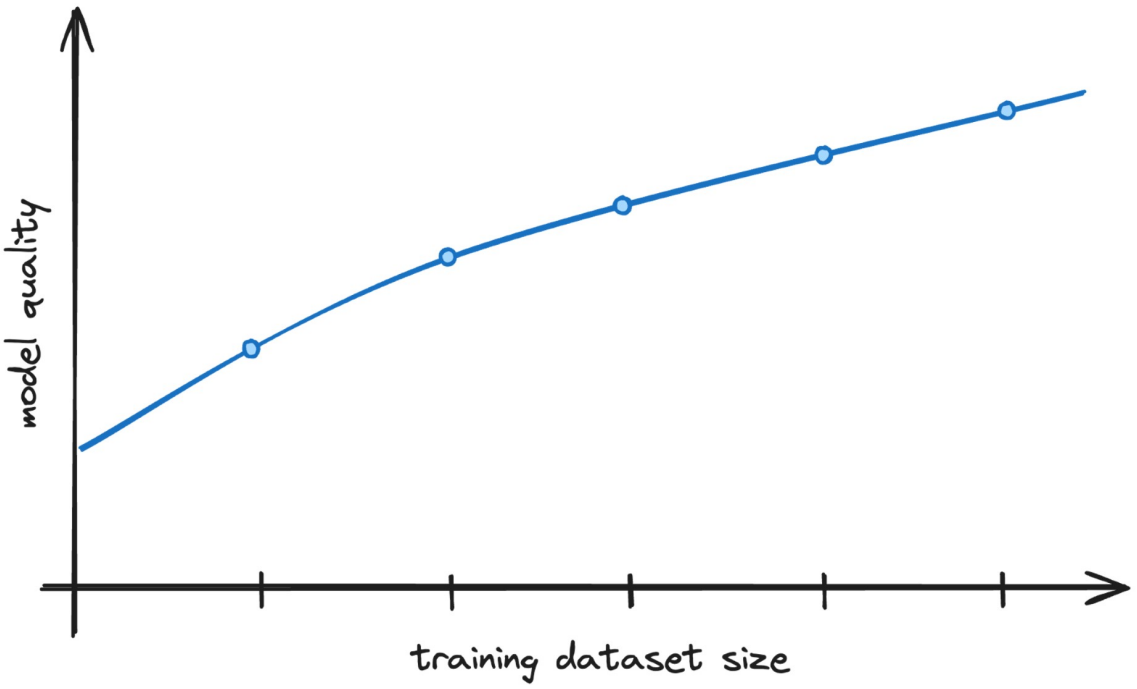

Figure 1: Model performance vs training data size - Learning curve for sentiment classification based of hierarchically nested training data

2.1.4 Weak labelling techniques

Using sentence embeddings, weak labels are to be generated for unlabeled texts by retrieving similar labelled texts and learning from their labels. Develop and compare weak labelling techniques by appropriate selection of algorithms, number of labeled texts and label weighting.

Also analyze the influence of your dataset's annotator agreement, i.e., how much weak labelling improves if hard labels with high or low annotator agreement are injected into the process.

Once you have generated weak labels, evaluate their quality before using them for your classification task. After the evaluation, you should be able to select the 2-5 most promising techniques and parameterizations.

2.1.5 Model training with additional weak labels

After generation of weak labels for the hierarchically nested training datasets, train a classification model now with hard and weak labels combined for each training sample size. Use the same training datasets as in the baseline training to allow for a fair comparison of the model performance.

Remember that in a real-world company context weak labels may be generated with virtually no cost. Hence, always utilize all available unlabelled data for weak labelling.

2.1.6 Model comparison

Compare the results with your baseline models and analyze how the model performance changes with additional weak labels under different weak label strategies. Also, compare the results of sentiment classification with the weak labels directly, i.e. the metrics for the weak labels obtained in section "Weak labelling techniques" above.

Eventually, decide on the best approach for your dataset and conclude. Is training a classification model with weak labels worthwhile or is the weak label generation process sufficient to generate the labels directly? If your weak labelling approach boosts the performance of the classification model, provide a time savings factor. This should indicate how many fewer hard labels are required when using your weak labelling approach to reach the same model performance.

2.2 Solution approach

The mini-challenge may be implemented individually or as a team in groups of maximally three persons.

Collaboration between groups is limited to conceptual aspects. In particular no code may be copied from other groups or from the Internet.

2.3 Exchange meetings and Deadline

Every team is required to set up in advance one **exchange meeting** during NPR contacts hours **at least four weeks before submission** of mini-challenge 2.

The deadline for this mini-challenge is December 19, 2025.

3 Foundation

3.1 Software

Python will be used in this mini-challenge; a consistent usage of existing frameworks is to be combined with implementation of own functions.

Hugging Face provides a wide selection of models which we utilize for this mini-challenge together with the Hugging Facce transformers library.

3.2 Data

For this mini-challenge we will using the financial phrasebank data set available at https://huggingface.co/datasets/takala/financial_phrasebank.

The dataset consists of 4840 sentences from English language financial news categorised by sentiment. The dataset is labelled as ‘positive’, ‘negative’ or ‘neutral’. The data is further divided by agreement rate of 5-8 annotators, i.e., whether 50%, 66%, 75% or all annotators agree on the sentiment. This information is to be considered and analysed carefully in weak labelling and when analyzing the classification error.

3.3 Infrastructure

It is advisable to outsource longer-running jobs to scripts that you can then execute on a computer or server with a GPU.

If training the classification model is not possible due to memory limitations or excessive runtime, you can apply for 50 CSCS hours for this purpose. Please note, that set up and usage of the CSCS infrastructure requires additional time.

3.4 Tools

The use of ChatGPT or comparable AI tools is permitted. Their use must be noted in the deliverable for corresponding pieces of code and briefly assessed and discussed in a separate section at the end of the analysis (length 250-500 words).

The task for which the AI tool was used and which prompting strategy was used must be specified In addition, it should be described which prompting strategy was most successful, i.e. which contributed most a) to solving the task and b) to the acquisition of skills.

4 Deliverables

4.1 Notebooks

Analyses must be submitted in the form of notebooks, whereby in addition to the .ipynb file, a version rendered as .html or .pdf must also be submitted. All analyses must be carried out in one piece before submission, the idea and execution of the analyses must be described precisely, and the results must be documented and interpreted comprehensibly.

4.2 Code repositories

In addition, a well-structured and documented repository of the final and executable codes must be made accessible including details on dependencies on additional libraries (i.e. requirements.txt or environment.yml file). Further recurring functionalities should be outsourced in script files, libraries or packages.

The title of the mini-challenge and authorship must be noted in the name. The analyses must be sent on time by e-mail to “daniel.perruchoud@fhnw.ch”.

5 Mini-Challenge assessment criteria

Grades are awarded based on the four assessment criteria listed below with **priority #1 on Traceability** and **priority #2 on Completeness**.

5.1 Traceability

The analyses must be designed in such a way that both the

- underlying considerations,
- their implementation and
- the derived results

are comprehensible. This presupposes that the

- notebooks and codes are well structured and commented according to best practice standards,
- the data pipeline is visually displayed and well explained,
- analytical results are presented with tables and graphics and are fully discussed,
- intermediary and final results are analyzed on a random sample of observations and critically inspected.

5.2 Completeness

The content of the analyses must be complete in accordance with the description of the mini-challenge.

5.3 Correctness

The submitted analyses are checked for correctness of content. Running your codes before form scratch before submission is mandatory.

5.4 Best practice standards

Code repetition should be avoided by copying code and outsourced to functions that are tested and logged before use.