Natural Language Processing - Mini-Challenge 2

Sentiment Analysis

Daniel Perruchoud & Joe Weibel, Institut für Data Science

This mini-challenge is slightly more structured and closely related to ongoing research at FHNW.

1 Mini-Challenge content & learning objectives

1.1 Background

In this mini-challenge, we focus on sentiment analysis, a typical NLP task. Sentiment in the NLP context refers to the mood or tone of a text. Consequently, it often involves classification into positive, negative, and neutral categories, or optionally binary classification and sentiment regression. However, the criteria for assigning a text to one of these classes may vary depending on the application.

Typically, this task is approached in a supervised learning setting, requiring sentiment labels for the texts. Creating labels manually is typically a costly, time-consuming task. Moreover, when multiple individuals assess texts, there may be discrepancies in judgments (annotator disagreement). An alternative to manual labelling is semi-supervised learning (weak supervision), where labels are automatically generated. With this method, more or even all sentences can be labelled. However, this method introduces uncertainties as well. The question arises as to how this uncertainty in the weak labels differs from the uncertainty in the ground truth resulting from annotator disagreement.

The practical value-add of this is mini-challenge is to help a company decide on the number of labels they need to manually annotate, and how much this annotation process can be reduced when only few manually annotated labels are used for semi-supervised learning.

1.2 Description

In this mini-challenge, you are tasked with examining how the amount of training data affects the quality of models for sentiment analysis. To improve the model quality, especially if only little manually annotated data is available, you will add weak labels to your training dataset.

1.2.1 Model setup

Choose your dataset, a pretrained language model from Hugging Face for the sentiment classification (e.g. BERT or an equivalent model), and suitable performance metrics. Get to know your dataset and provide basic statistical characteristics.

Prepare your data to train and evaluate the classification model using varying amounts of training data, i.e. construct a hierarchically nested data split to simulate different amounts of available labels. You will have multiple differently sized training datasets grown so that larger datasets contain all observations of preceeding smaller datasets. Note, that you will need either a separate validation dataset or use cross-validation for evaluation.

1.2.2 Baseline classification model

Decide whether transfer learning or fine-tuning is the suitable approach for the pre-trained classification model. Apply the selected technique for each sample size of the hierachically nested data partitions, find suitable model hyperparameters and evaluate the model's performance considering training data size.

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.

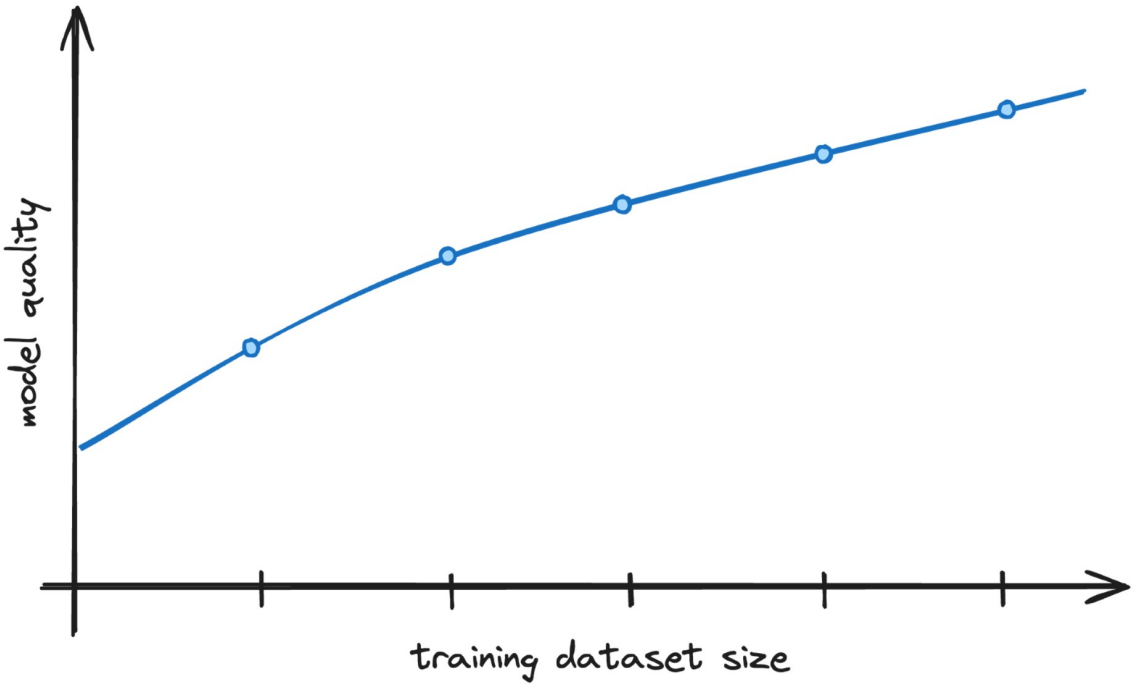

Figure 1: Model performance vs training data size - Learning curve for sentiment classification based of hierarchically nested training data

1.2.3 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 such models for generating text embeddings and explore the results quantitatively and qualitatively. To inspect the embeddings' quality, you may calculate and visualize similarities.

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

1.2.4 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.

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.

Bonus task: Calculating similarities between a text and all other texts can be time-consuming, especially when done for multiple texts. Consider optimizing this process by calculating centroids, apply local sensitive hashing or another clever technique.

Alternatively, you could verify the influence of your dataset's annotator agreement if different levels are available.

1.2.5 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. Remember that in a real-world company context weak labels may be generated with virtually no cost. Hence, always utilize all available weak labels in your work. Use the same training datasets as in the baseline training to allow for a fair comparison of the model performance.

1.2.6 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.

1.3 Learning objectives

- 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

2 NPR achievement of competences

In the “Natural Language Processing (NPR)” module, competence is acquired by demonstrating practical and theoretical skills.

Practical competence is assessed by means of two mini-challenges, theoretical competence by means of an oral MSP, which is completed after submission of the mini-challenges.

The overall assessment is made up of the graded mini-challenges and the oral MSP.

3 Mini-Challenge Basics

3.1 Software

Python will be used in this mini-challenge and 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.

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

3.2 Data

The decision which dataset to use is up to you. Please ensure that the number and distribution of sentiment labels is sufficient (hint: at least 100 records with labels and 500 without labels, not too large to allow model training). You can also choose a dataset with many labels and remove some of them to build your unlabelled dataset.

Please note: If you choose a dataset with long texts (longer than the context length of the models), you must consider how to handle it. You can apply undersampling to your data to avoid struggles with an imbalanced dataset. You should be able to find suitable datasets on huggingface.com/datasets or kaggle.com/datasets. Alternatively, you can use one of the following datasets. If you wish to use a different dataset, please discuss your choice briefly with me.

Recommended datasets:

- financial_phrasebank (https://huggingface.co/datasets/financial_phrasebank)
- amazon_polarity (https://huggingface.co/datasets/amazon_polarity)
- sst2 (https://huggingface.co/datasets/sst2)

3.3 Infrastructure

Ideally, you run and train the models on your computer 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.

4 Mini-Challenge submission conditions

4.1 Deliverables

4.1.1 Notebooks

Analyses must be submitted in the form of notebooks, in addition to the .ipynb file, a version rendered as .html or .pdf must 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 in a comprehensible manner.

4.1.2 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”.

4.2 Teamwork

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.

4.3 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 assessed. 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.

4.4 Exchange meetings and Deadline

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

The deadline for this mini-challenge is December 20, 2024.

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 Completenss.

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 splitting strategy is visually displayed and well explained,
- analytical results are presented with tables and graphics and are fully discussed in text form,
- 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 before use.