On the implementation of baselines and *Lightweight Conditional Model Extrapolation (LIMES)* for class-prior shift

Paulina Tomaszewska, Christoph H. Lampert
Outline

1. Overview of LIMES method
2. Implementation standpoint
3. Reproducibility of the results
LIMES - adaptation using analytical formula

LIMES achieves superior performance to baselines especially in the most difficult scenarios

Inspiration - MAML

Empirical experiments - setting
Reproducibility of the results

1. dataset
2. code - available at https://github.com/ptomaszewska/LIMES:
   • LIMES
   • baselines
   • analysis of the results (plots)
Dataset

• collected using free Twitter's Streaming API*
• cannot be shared publicly in a raw version (prepared for direct use for training) due to Terms of Service

Solution:

• we put the dehydrated dataset on the website
How to restore the original data?

How to use the Twarc?
1. **Configuration:**
   - provide Twitter application API keys
   - grant access to Twitter account
2. **Rehydration:**
   - use command `twarc hydrate`

After rehydration, the dataset size increases 200x.
Implementation
Tools

python

Keras

TensorFlow

backend

python libraries specified in requirements.txt

cluster with SLURM queuing system

(to ensure increased efficiency in case of large size of dataset)
Pipeline - scripts

- Preprocessing of json files with Twitter data
- Generating embeddings
- Creating subsets of dataset
- Training of a model
Generating embeddings

• pretrained `distiluse-base-multilingual-cased-v1` multilingual sentence embedding network\(^1\) (suitable for social media texts)

• results saved to `.npy` files

\(^1\)https://github.com/UKPLab/sentence-transformers
Model training

• custom training loop and layer definition (incorporating non-trainable bias correction term)
• streaming data is simulated

Each sample is processed only once in a chronological order.

• the training step is wrapped using `tensorflow.function` decorator
  → model is treated as static graph
  → global performance optimization enabled

Note: the default mode in Tensorflow 2 is eager execution which can slow down training but can be useful for the debugging.
Model training

- script for model training can be directly used to reproduce results of experiments on Twitter data
  - for other use (replication), the code can be easily adjusted (thanks to modularity)

- baseline methods can be specified as an argument – they are implemented using if conditions
Large scale training

• it is recommended to run experiments on cluster
• it can be done efficiently using two bash scripts from the repository
  1. scheduling training within loops iterating over different parameters of experiments (subset, realization, data source, model)
  2. the experiments are sent to the SLURM queue and run
Towards high level reproducibility\textsuperscript{1}

\begin{itemize}
  \item source code available at github repository
  \item permissive MIT license (very limited restriction on reuse)
  \item requirements.txt
  \item configuration scripts to run the experiments on SLURM queuing system
  \item detailed instructions on how to get the data and run the code
\end{itemize}

Towards high level reproducibility\(^1\) – regarding code

- good code writing practices:
  - modularity
  - informative variable names
  - comments
- fixed random seed (for train/validation split, weight initialization)
- code for results analysis and figure generation provided
Facilitating reproducibility

Good practice is to provide Google Colaboratory notebooks

**BUT** we didn’t

Why?

- data cannot be publicly shared in an original form and rehydration requires personal keys to configure the Twarc tool
- dataset is of significant size
- computations take significant time so cluster would be helpful
Credibility of results

Common problems:
• selective reporting of results
• drawing conclusions from insufficient number of experiments

How did we mitigate these risks?
1. We generated many dataset subsets:
   • 2 different time subsets
     • from 1\textsuperscript{st} to 5\textsuperscript{th} day of each month – “early”
     • from 11\textsuperscript{th} to 15\textsuperscript{th} day – “lately”
   • 10 different realizations of data
   • 3 different data sources (“tweet”, “location” and concatenation of both)

   to analyze whether there is correlation between the task
difficulty and the performance boost
of the proposed method
Credibility of results – measures taken:

2. We compute 2 metrics: *avg-of-avg* and *avg-of-min accuracy* to reduce bias that could be introduced by the choice of the metric and show the broader spectrum of advantages and limitations of **LIMES**

3. We report mean and std of metric values across runs

4. We provide in code the exact way in which the metrics are computed

5. We applied the Wilcoxon statistical test to validate the performance of the proposed method over the baseline

6. We combined statistical analysis with the results visualization
Supplement
Reproducibility vs. students

• Reproducibility is a great topic for student projects:
  1. Taking the github repository and checking whether the code works and comparing results
  2. Implementing the solution from the paper (when code is not provided) and verifying if the results match

• Both raise awareness and can be adjusted to students’ skills

• The students work from my teaching group:
  https://mi2-education.github.io/2021L-WB-Book/ (chapter 4)

• analogy: ReproducedPapers.org
Thank you very much!

Paulina Tomaszewska, Christoph H. Lampert