Multilingual Natural Language Processing

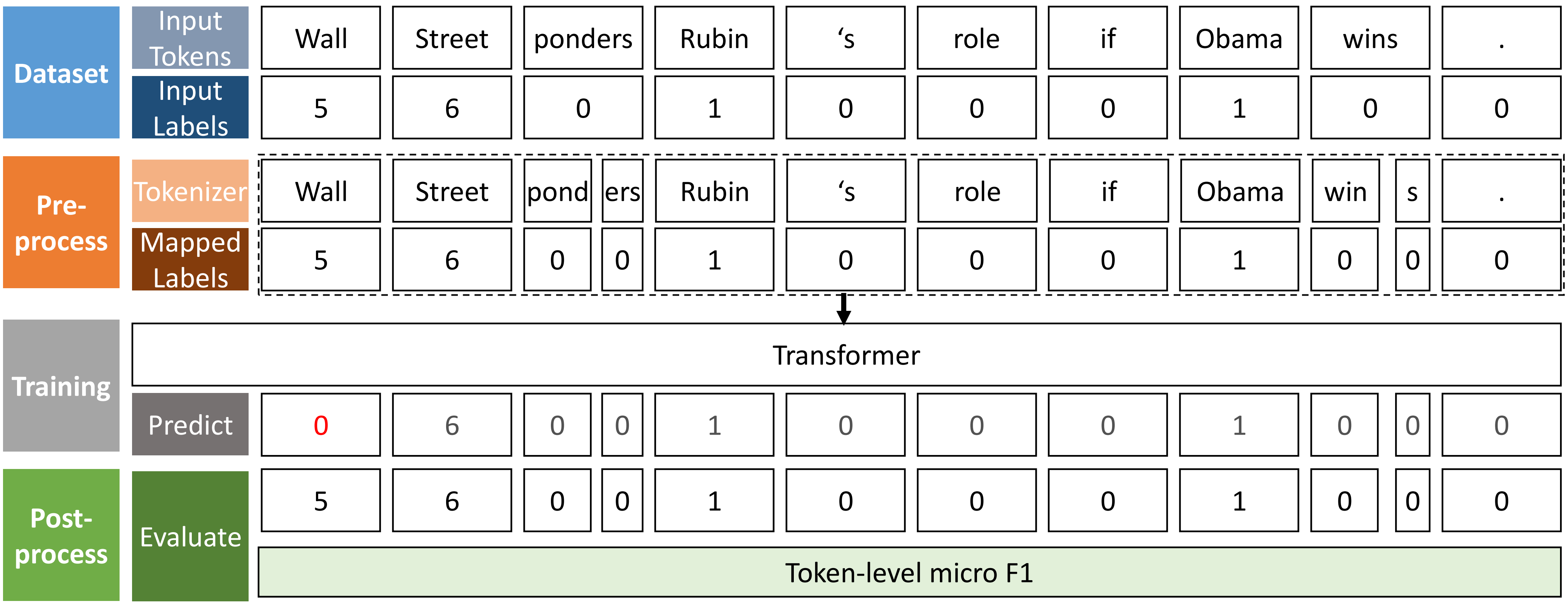
Team Projects

Goran Glavaš, Benedikt Ebing, Fabian David Schmidt

Project Overview

- All groups (3 students) will tackle the same base task named entity recognition
- Groups can select approach on how to tackle the task
- Short project presentations (~10 minutes) will be held 14th July
- Coaching sessions with TAs on demand (at most 2)
- Grading on 4-point scale from 0 to 3 points that count toward exam bonus
  - Take this nevertheless as a learning experience!

Token Classification for Named Entity Recognition with Transformer Models: Task at a Glance



Token Classification for Named Entity Recognition with Transformers: Task Details

- **Base model:** smaller pre-trained multilingual transformers
- **Goal:** implement entire token classification pipeline & architectural/model tweak by yourselves
- **Datasets:**
  - **Source language:** CoNLL 2003 English / (WikiANN English)
  - **Target language(s):** MasakhaNER
- **Infrastructure:** Google Colab / Kaggle
- **Key:** split tasks wisely!

Intermittent Language Modelling for Better Cross-Lingual Transfer

Rationale

- While multilingual language models span 100+ languages, vast majority of 7K languages are un(der)represented in today's models
- Post-hoc language modelling greatly improves transfer capabilities to unseen languages (provided tokenizer *can* tokenize unseen language meaningfully)

Language Modelling

- Bilingual language modelling simultaneously on source & target language improves and stabilizes cross-lingual transfer
- Representations re-fined from multilingual representation space
- Suitable for post-hoc addition of language unseen in initial multilingual pre-training

Project

- Bilingual Language Modelling of Source & Target Language (English + Yoruba)
- Perform zero-shot transfer from CoNLL (news-domain) to languages part of MasakhaNER
- Comparative Evaluation and Analysis between bilingually specialized and original multilingual model

BAD-X: Bilingual Adapters Improve Zero-Shot Cross-Lingual Transfer, NAACL 2022, https://aclanthology.org/2022.naacl-main.130/
Yoruba data: https://github.com/ajesujoba/YorubaTwi-Embedding
Zero-shot Cross-lingual Transfer is Under-specified Optimization, Repl4NLP 2022, https://arxiv.org/pdf/2207.05666.pdf

Parameter-Efficient Fine-Tuning (PEFT)

Rationale

- Storage & training requirements are proportional to model size
- Model size keeps on increasing (albeit maybe starting to hit limits)
- Practical issue: hardly feasible to fine-tune large models since they do not fit on GPU VRAM

PEFT strategies need less VRAM

- PEFT strategies fine-tune only a small fraction (0.1-3%) of the parameter count of the original model
- PEFT keeps (most often close to) performance of `full fine-tuning'
- **Strategies:**
  - **BitFit:** only fine-tune bias terms of layers
  - **Prefix-Tuning:** add new input embeddings
  - **Adapters, LoRA, …**

Project

- Implement BitFit **or** Prefix-Tuning from scratch (w/o dedicated frameworks) and compare against full fine-tuning
- Perform zero-shot transfer evaluation from both WikiANN (wiki-domain) and CoNLL (news-domain) to languages part of MasakhaNER (African languages in news domain)

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models, ACL 2022, https://aclanthology.org/2022.acl-short.1/
Prefix-Tuning: Optimizing Continuous Prompts for Generation, ACL 2021, https://aclanthology.org/2021.acl-long.353.pdf

SLICER: Sliced Fine-Tuning for Low-Resource Cross-Lingual Transfer for NER

Rationale

- **Premise:** fine-tuning named entity recognition decontextualizes word representations
- **Implication:** implicit `overfitting‘ on monolingual token properties (casing, prefixes, suffixes)
- **Effect:** quality of cross-lingual transfer to distant languages suffers, as no subwords overlap and syntax often is very different

SLICER

- SLICER is an approach to force token representations to retain more contextualization in monolingual fine-tuning, leading to more robust transfer in challenging scenarios
- **Intuition:** train classification on slices (sub-segments, cf. multi-head attention) of token representations, disabling the transformer to co-adapt on redundancies; inference ‘ensemble’ over slices

Project

- Implement SLICER training step from scratch and compare against full fine-tuning
- Perform zero-shot cross-lingual transfer evaluation from both WikiANN (wiki-domain) and CoNLL (news-domain) to languages part of MasakhaNER (African languages in news domain)

SLICER: Sliced Fine-Tuning for Low-Resource Cross-Lingual Transfer for Named Entity Recognition, EMNLP 2022, https://aclanthology.org/2022.emnlp-main.740/

Roadmap for the project

- Write the LightningModule
  - Use „xlm-roberta-base“ as encoder
  - Write your own model head for token classification
  - Train your model minimizing cross entropy loss
  - Evaluate your models on micro F1
  - Use the AdamW optimizer with:
    - Learning rate: 2e-5
    - Weight decay: 0.05
  - Add you projection specific modifications

Roadmap for the project

- Write the LightningDataModule
  - Datasets to use:
    - Train, Validation:
      - https://huggingface.co/datasets/conll2003
      - https://huggingface.co/datasets/wikiann
    - Test: https://huggingface.co/datasets/masakhaner
    - Take care when preprocessing the data (token classification task!)
    - Additional resource: https://huggingface.co/learn/nlp-course/chapter7/2?fw=pt
  - Take care of multiple test datasets (one for each target language)
    - https://lightning.ai/docs/pytorch/LTS/guides/data.html

Roadmap for the project

- Write the final training script
  - Train for 10 epochs on ConLL / 5 epochs on WikiAnn
  - Test the model performance on the last checkpoint

Do’s & Don’ts

Do’s

- Use `AutoModel.from_pretrained`
- Write your own classification head tailored to the token classification task
- Use available frameworks to simplify boilerplate code (pre-processing, post-processing, CLI, etc.) and transformer implementation
- Refer to existing code with code comment citations

Don’ts

- Blindly copy available open-source code
- Turn a group project into a single person effort