Allen Institute for AI @ SIGTYP 2024 Shared Task

Lester James V. Miranda
ljm@allenai.org

EACL 2024

Problem Statement

Constrained Task (Goal)

Build a system for three linguistic tasks—parts-of-speech (POS) tagging, morphological annotation, and lemmatization.

Dataset

Contains Universal Dependencies v2.12 data and Hungarian codices. Dated before 1700 CE, with four language families and six scripts. Has 2.6M tokens for training and ~300k tokens for validation and testing.

Code	Language
CHU	Old Church Slavonic
COP	Coptic
FRO	Old French
GOT	Gothic
GRC	Ancient Greek
HBO	Ancient Hebrew
ISL	Medieval Icelandic
LAT	Classical and Late Latin
LATM	Medieval Latin
LZH	Classical Chinese
OHU	Old Hungarian
ORV	Old East Slavic
SAN	Vedic Sanskrit

Methodology - General approach

Pretraining a transformer-based multilingual LM on the shared task tokens, and then **fine-tuning** it on the UD annotations of each language.

Input: Shared task tokens
Output: Multilingual base model
Learning Method: Unsupervised

Input: UD annotations
Output: Fine-tuned model (per language)
Learning Method: Supervised

Methodology - Related work

Multilingual pretraining

- XLM-RoBERTa (Conneau et al., 2020) and mBERT (Devlin et al. 2019) are one of the few demonstrations of a single LM pretrained on a mixed-language corpora.
- Pires et al. (2019) showed that multilingual BERT is "surprisingly good" at zero-shot cross-lingual model transfer.
- However, majority of these models were trained primarily on modern text.

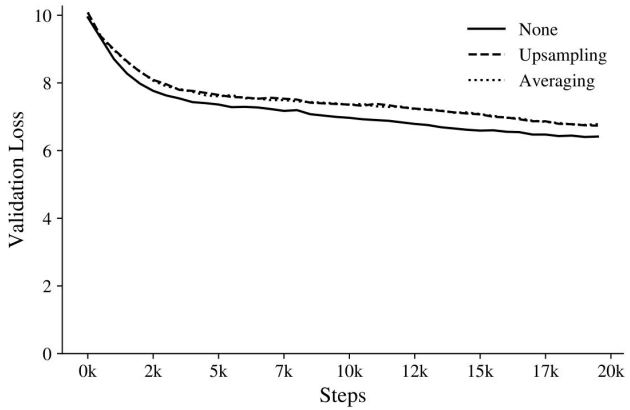
Multilingual LMs for ancient and historical languages

- Few multilingual LMs for our domain of interest. Notable examples include hmBERT (Schweter et al., 2022), Persian spaCy models (Kapan et al., 2022), etc.
- Most works are language-specific and focus on named entity recognition.

Methodology - Model pretraining (1/4)

Building the pretraining corpus → Pretraining architecture → Pretraining setup and hyperparam.

- We used the tokens from the UD corpora to construct the pretraining corpus.
- Initially, we experimented on sampling techniques to balance our dataset.
  - Sampling unique tokens based on the language with the largest representation (i.e., Medieval Latin).
  - However, this process made the pretraining unstable with little pretraining val. accuracy gain.



Validation loss after 20k steps for each sampling strategy.

Methodology - Model pretraining (2/4)

Building the pretraining corpus → **Pretraining architecture** → Pretraining setup and hyperparam.

Our configuration resembles that of XLM-RoBERTa's setup (with variants)

Architecture Details (Base)	
Hidden Size	768
Intermediate Size	3072
Max position embeddings	512
Num. attention heads	12
Hidden Layers	12
Dropout	0.1

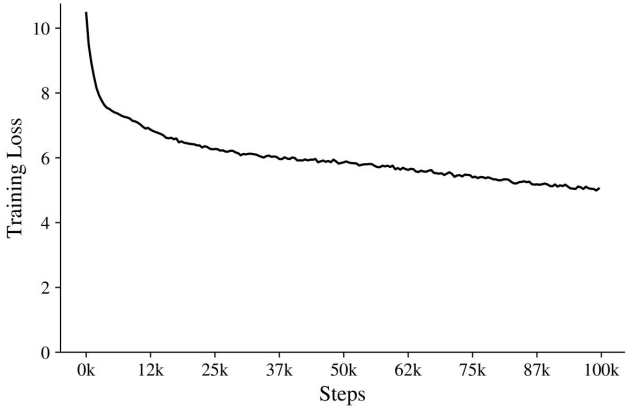
Variants	Layers	# Parameters
Base	12	126-M
Medium	8	43-M
Small	4	32-M
Mini	4	15-M
Tiny	2	7-M

Methodology - Model pretraining (3/4)

Building the pretraining corpus → Pretraining architecture → **Pretraining setup and hyperparam.**

Then we tuned our training hyperparameters

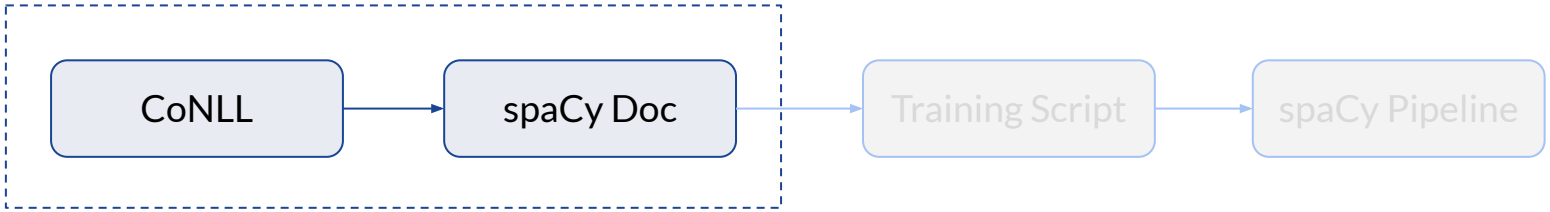
Training Parameters	
Tokenizer	Byte-pair Encoding (Sennrich et al., 2016)
Optimizer	AdamW (β_2=0.98 and decay=0.1)
Learning Rate	2e-4
Training steps	100k (linear warm up for the first 12k)



(Pre)training loss after 100k steps.

Methodology - Model fine-tuning (1/7)

We used the UD annotations to train language-specific pipelines.

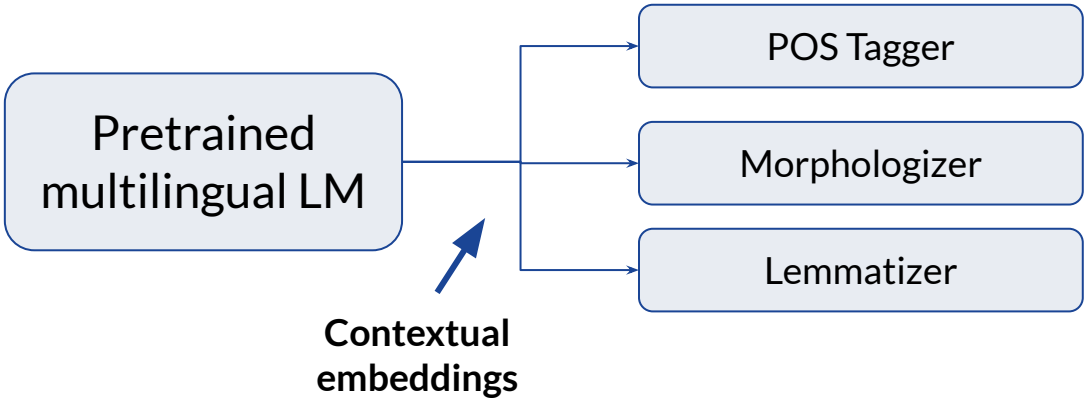


Preprocessing step while preserving the tokenization in the original annotations.

Methodology - Model fine-tuning (2/7)

Preparing the dataset → **Fine-tuning Architecture** → Fine-tuning Setup

We used spaCy (Honnibal et al., 2020) to train multi-task pipelines per language.



On a high-level:

- We access the contextual embeddings from our pretrained LM's last hidden state.
- We pass these embeddings as add'l input to our downstream task-specific network.

Methodology - Model fine-tuning (3/7)

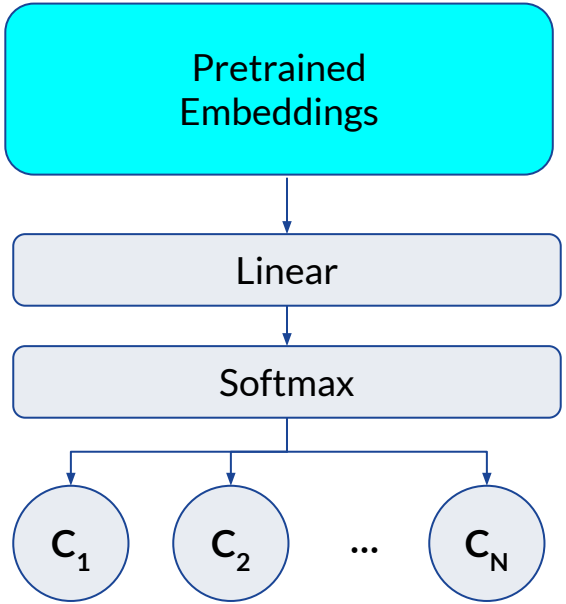
Preparing the dataset → **Fine-tuning Architecture** → Fine-tuning Setup

Parts-of-speech tagger

- We employed a standard multiclass classifier that predicts a vector of tag probabilities for each token.
- {C} = set of all POS tags.

Morphologizer

- This time, we treat every unique combination of morphological features as a single class.
- {C} = set of unique morph combinations



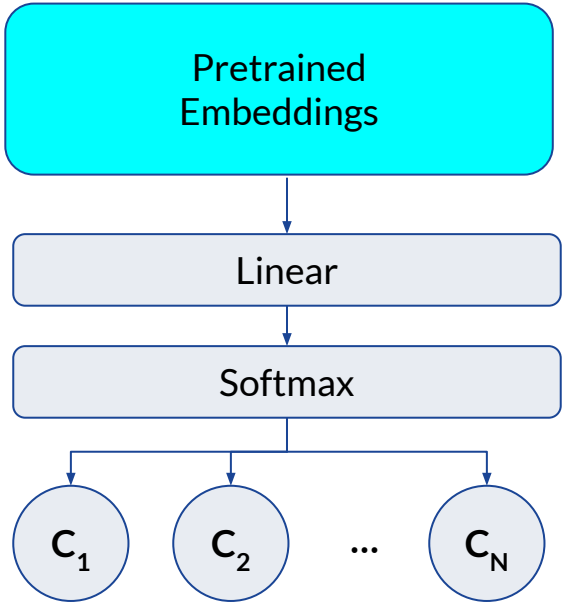
Loss function is NLL

Methodology - Model fine-tuning (4/7)

Preparing the dataset → **Fine-tuning Architecture** → Fine-tuning Setup

Lemmatizer

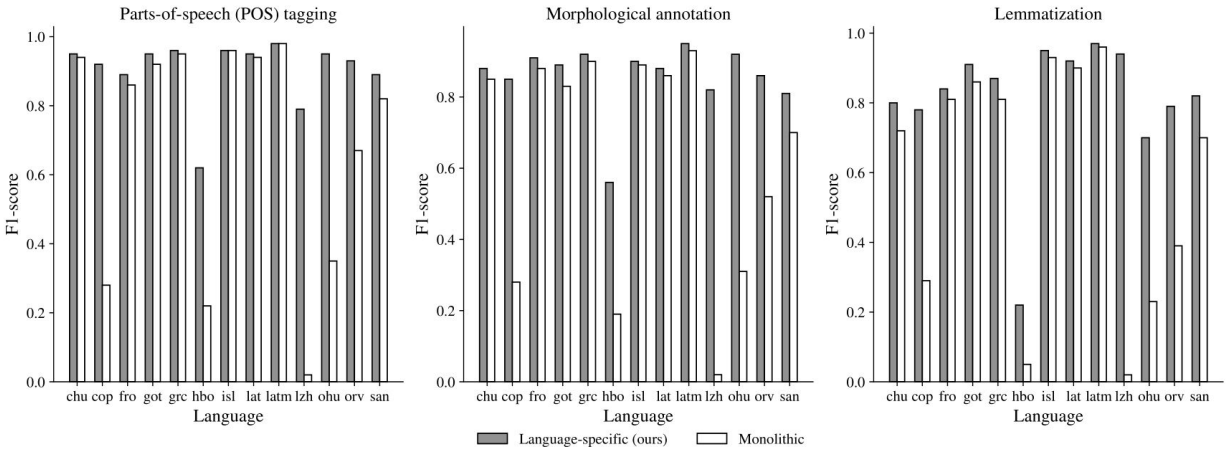
- We used a neural-based edit tree lemmatizer (Müller et al., 2015) by extracting an edit tree for each token-lemma pair.
- {C} = set of all possible edit trees
- If the most probably edit tree cannot be applied to a token, then we back off to the most probable tree (until we reach the surface form).



Loss function is NLL

Methodology - Model fine-tuning (5/7)

Preparing the dataset → Fine-tuning Architecture → **Fine-tuning Setup**

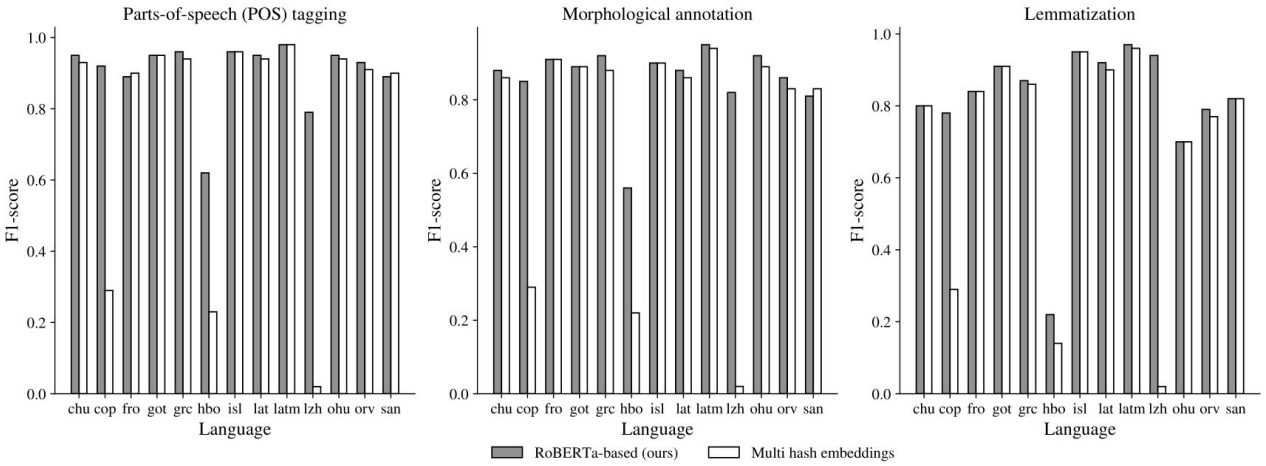


We also tried fine-tuning on the combined dataset (**monolithic**) as opposed to creating **language-specific** models.

Findings: Our fine-tuned language-specific models still perform better on the development set.

Methodology - Model fine-tuning (6/7)

Preparing the dataset → Fine-tuning Architecture → **Fine-tuning Setup**



We also tried using spaCy's default embedding method, multi hash embed (Miranda et al., 2022), as opposed to our RoBERTa LM.

Findings:

Transformer model performs better, but the hash embed is more memory-efficient.

Methodology - Model fine-tuning (7/7)

Preparing the dataset → Fine-tuning Architecture → **Fine-tuning Setup**

In the end, we fine-tuned RoBERTa for each language

Training Parameters	
Tokenizer	Byte-pair Encoding (Sennrich et al., 2016)
Optimizer	Adam (β_1=0.99, β_2=0.999 and decay=0.1)
Learning Rate	0.001 (linear warm up for the first 25% steps)

- We fine-tuned a multi-task pipeline for each language.
- You can find the rest of the training configuration in our GitHub repository

Language	spaCy Model Name	Model Size
Old Church Slavonic	`xx_chu_sigtyp_trf`	491 MB
Coptic	`el_cop_sigtyp_trf`	477 MB
Old French	`xx_fro_sigtyp_trf`	471 MB
Gothic	`xx_got_sigtyp_trf`	474 MB
Ancient Greek	`xx_grc_sigtyp_trf`	501 MB
Ancient Hebrew	`he_hbo_sigtyp_trf`	475 MB
Medieval Icelandic	`xx_isl_sigtyp_trf`	496 MB
Classical and Late Latin	`xx_lat_sigtyp_trf`	486 MB
Medieval Latin	`xx_latm_sigtyp_trf`	510 MB
Classical Chinese	`zh_lzh_sigtyp_trf`	469 MB
Old Hungarian	`xx_ohu_sigtyp_trf`	488 MB
Old East Slavic	`xx_orv_sigtyp_trf`	503 MB
Vedic Sanskrit	`xx_san_sigtyp_trf`	473 MB

spaCy pipelines that we trained for each language.

Results & Discussion - Test set performance

Lang.	POS tag.	Morph. annot.	Lemma.
CHU	0.946	0.941	0.796
COP	0.426	0.805	0.463
FRO	0.851	0.941	0.833
GOT	0.934	0.940	0.908
GRC	0.935	0.965	0.883
HBO	0.273	0.712	0.618
ISL	0.939	0.948	0.946
LAT	0.924	0.933	0.923
LATM	0.944	0.980	0.972
LZH	0.818	0.860	1.000
OHU	0.944	0.946	0.700
ORV	0.912	0.922	0.784
SAN	0.873	0.900	0.832

Metrics

- POS-tagging: Accuracy@1, F1-score
- Morph. annot: Macro-avg of Acc @1 per tag
- Lemma.: Accuracy@1, Accuracy@3

Results

- Results greater than baseline are highlighted in green.
- Decent results on POS-tagging and Morphological annotation, not so much in lemmatization.

Results & Discussion - Dev't set performance

Lang.	POS tag.	Morph. annot.	Lemma.
CHU	0.947	0.876	0.803
COP	0.924	0.846	0.776
FRO	0.890	0.912	0.844
GOT	0.951	0.886	0.914
GRC	0.956	0.915	0.873
HBO	0.624	0.561	0.219
ISL	0.963	0.901	0.949
LAT	0.949	0.882	0.922
LATM	0.984	0.951	0.968
LZH	0.795	0.824	0.942
OHU	0.953	0.919	0.697
ORV	0.933	0.859	0.787
SAN	0.888	0.811	0.817

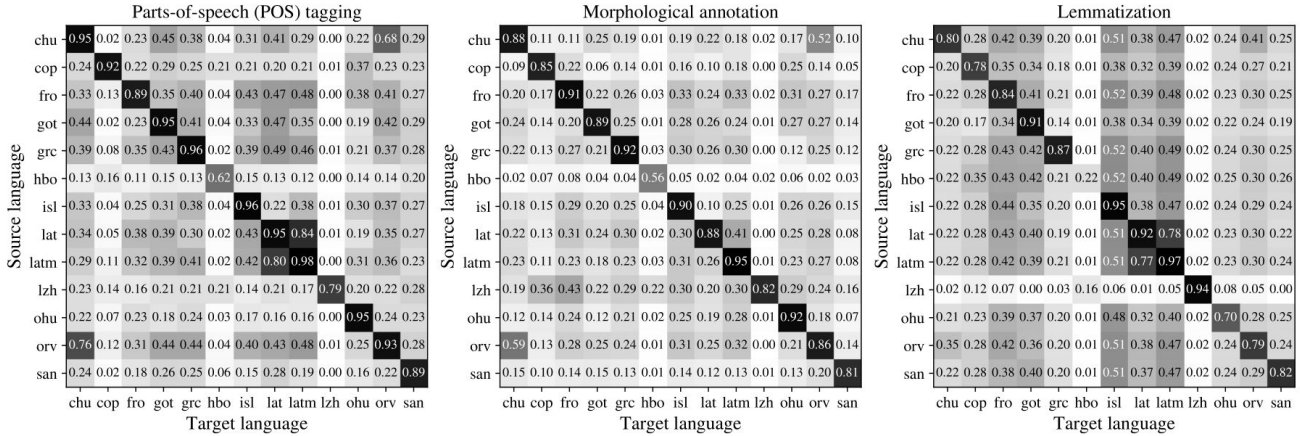
Metrics

- F1-score for all tasks

Results

- Decent results overall.
- Large difference on Coptic (COP) and Ancient Hebrew (HBO) because our pipeline cannot handle *multi-word expressions* (will discuss later in Limitations).

Results & Discussion - Cross-lingual capabilities



Does a model from one language work on another?

- F1-score on the development set.

- Classical and Late Latin (LAT) and Medieval Latin (LATM) have decent cross-lingual perf. because the latter is a direct continuation of the former.
- Some languages in the Indo-European family such as Old Church Slavonic (CHU) and Old East Slavic (ORV) have decent cross-lingual performance.

Results & Discussion - Limitations & Challenges

Pretrained LM size

Because of computing constraints, we were only able to pretrain an LM w/ the same size as RoBERTa-base.

Label combination as indiv. classes

Each feature combination is treated as an indiv. class. Therefore, our classifier can only predict combinations it has seen during training.

Subtoken performance and MWEs

Our pipeline can only predict on the full token / MWE, not on its sub-parts. In our submission, we have to substitute each subtoken on the text, resulting to sentences that may not be *correct* and impact our context-sensitive embeddings.

Conclusion

- Our **general approach** involves pretraining a multilingual LM on the UD tokens, and fine-tuning a multitask pipeline on the annotations.
  - We have produced 1 multilingual LM and 13 fine-tuned models (one for each language).
- Good results on the test set: majority of language-task pairs outperform the baseline. Development set performance is also good.
- All models and artifacts are available on GitHub and Hugging Face.

Models

Code & Experiments

Thank you!