Team 21a's Submission to the SIGTYP 2024 Shared Task on Word Embedding Evaluation for Ancient and Historical Languages

Anonymous ACL submission

Abstract

In this paper, we describe Team 21a's submission to the constrained track of the SIGTYP 2024 Shared Task. Using only the data provided by the organizers, we pretrained a transformer-based multilingual model, then finetuned it on the Universal Dependencies (UD) annotations of a given language for a downstream task. We also explored the cross-lingual capability of our trained models. [Our systems achieved]$_{LJ}$

1 Introduction

This paper describes Team 21a's submission to the *constrained* track of the SIGTYP 2024 Shared Task on Word Embedding Evaluation for Ancient and Historical Languages. Our general approach involves pretraining a transformer-based multilingual model on the shared task dataset, and then finetuning the pretrained model using the Universal Dependencies (UD) annotations of each language. Throughout this paper, we will refer to the pretrained model as LIBERTUS. We also explored data sampling and augmentation techniques during the pretraining step to ensure better generalization performance.

Our systems achieved...[stuff]$_{LJ}$. Table 1 shows our systems' performance on the shared task test set.

We detail our resource creation, model pretraining, and finetuning methodologies. The source code for all experiments can be found on GitHub: [insert github link or OSF link for anonymity]$_{LJ}$.

2 Methodology

2.1 Resource creation

We constructed the pretraining corpora using the annotated tokens of the shared task dataset. Then, we explored several data augmentation techniques to ensure that each language is properly represented based on the number of unique tokens.

	POS tag.	Morph. annot.	Lemma.
CHU			
COP			
FRO			
GOT			
GRC			
HBO			
ISL			
LAT			
LATM			
LZH			
OHU			
ORV			
SAN			

Table 1: SIGTYP 2024 Shared Task final results as evaluated on the test set. Cells where we obtained the best competition score are highlighted in green.

From our experiments, upsampling underrepresented languages helped reduce our pretraining validation loss. Figure 1 shows that LATM has the most number of unique tokens in the corpora. We upsampled each language by randomly duplicating a document in the training pool until the number of unique tokens is greater than or equal to that of LATM. The same figure also shows the token counts after the augmentation process.

2.2 Model Pretraining

Using the pretraining corpora, we trained a model with 126M parameters, that will later on serve as bases for finetuning downstream tasks. LIBERTUS follows RoBERTa's pretraining architecture (Liu et al., 2019) and takes inspiration from Conneau et al. (2020)'s work on scaling BERT models to multiple languages.

Our hyperparameter choices closely resemble that of the original RoBERTa implementation as seen in Table 2. We also trained the same BPE tok-

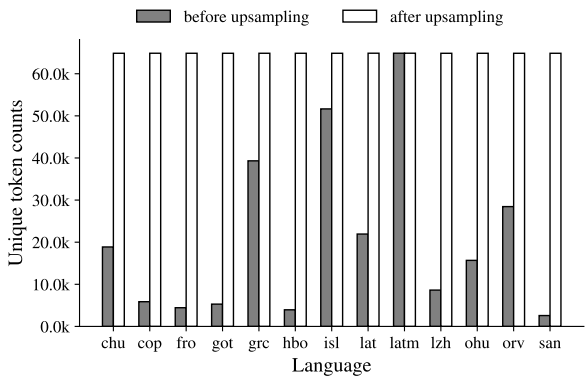

Figure 1: Unique token counts for each language before and after upsampling.

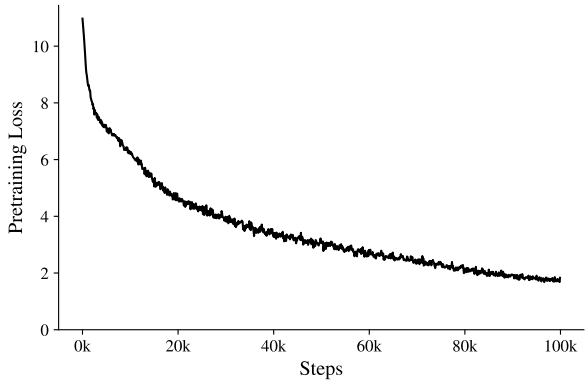

Figure 2: Training loss curve for the 126M-parameter model after 100k steps.

enizer (Sennrich et al., 2016) using the constructed corpora. During model pretraining, we used the AdamW optimizer with β_2=0.98 and a weight decay of 0.01. The base model underwent training for 100k steps with a learning rate of 2e-4. We used a learning rate scheduler that linearly warms up during the first quarter of the training process, then linearly decays for the rest. Figure 2 shows the training curve.

2.3 Model Finetuning

For each language, we finetuned a multitask model using spaCy (Honnibal et al., 2020). The final system consists of a parts-of-speech (POS) tagger, morphological analyzer, and lemmatizer.

Parts-of-speech (POS) tagger: We employed a standard classifier that predicts a vector of tag probabilities for each token. Each POS tag is a unique class that we assign exclusively to a token. We trained a model by taking the context-sensitive vectors from our pretrained embeddings, and passing it to a linear layer with a softmax activation. The network is then optimized using a categorical cross-entropy loss.

Hyperparameters	Value
Hidden size	768
Intermediate size	3072
Max position embed.	512
Num. attention heads	12
Hidden layers	12
Dropout	0.1

Table 2: Hyperparameter configuration for the LIBERTUS pretrained model.

Morphological analyzer: Similar to the POS tagger, we treat morphological annotation as a token classification task. Instead of directly modeling each feature, we made every unique combination of morphological features as a class. The limitation of this approach is that it can only predict combinations that were present in the training corpora.

Lemmatizer: We trained a neural-based edit tree lemmatizer (Müller et al., 2015) by first extracting an edit tree for each token-lemma pair. Because this process can result to hundreds of edit trees, we treat the problem of picking the correct tree as a classification task. Here, each unique tree serves as a class and we compute a probability distribution over all trees for a given token. To obtain the most probable tree, we passed the context-sensitive embeddings from our pretrained model to a softmax layer and trained the network with a cross-entropy loss objective.

We set the minimum frequency of an edit tree to 3, and used the surface form of the token as a backoff when no applicable edit tree is found. Finally, we ensured that the lemmatizer checks at least a single tree before resorting to backoff.

We trained each component of the system in parallel, although the final "pipeline" assembles them together using the spaCy framework. For all components, the pretrained embeddings are passed on to linear layer with softmax activation. Sometimes, the tokenization from the multilingual model does not align one-to-one with spaCy's tokenization. In such case, we use a pooling layer that computes the average of each feature to obtain a single vector per spaCy token.

During finetuning, we used the Adam optimizer

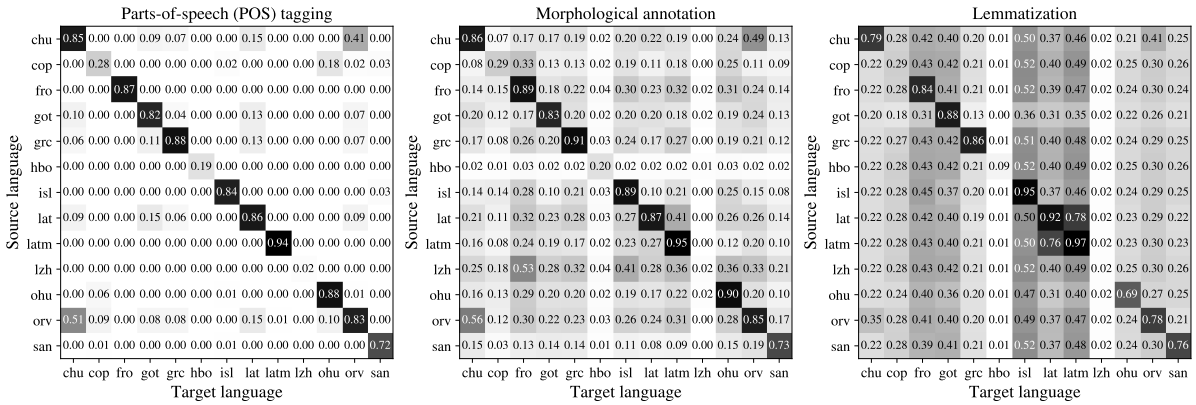

Figure 3: Cross-lingual evaluation given a monolingual model from one language and a validation set in another.

	POS tag.	Morph. annot.	Lemma.
CHU	0.851	0.861	0.794
COP	0.281	0.286	0.294
FRO	0.866	0.890	0.836
GOT	0.820	0.828	0.877
GRC	0.883	0.906	0.861
HBO	0.194	0.202	0.094
ISL	0.836	0.891	0.945
LAT	0.859	0.869	0.918
LATM	0.938	0.946	0.967
LZH	0.021	0.021	0.021
OHU	0.883	0.903	0.692
ORV	0.833	0.853	0.782
SAN	0.715	0.733	0.758

Table 3: F1-score results on the validation set. Cells with scores below 0.5 (random chance) are highlighted in red.

with β_1=0.9, β_2=0.999 and a learning rate of 0.001. The learning rate also warms up linearly for the first 250 steps, and then decays afterwards.

3 Results

Table 1 shows the test scores for the shared task. [Talk about how you ranked against other teams?]$_{\text{LJ}}$ In this section, we will outline our internal evaluations and benchmarking experiments.

3.1 Performance on the validation set

First, we evaluated our finetuned models on the validation set. Table 3 shows the results. Although we obtained strong performance on the majority of languages, our results on COP, HBO, and LZH are poor. These languages have non-latin scripts, and our BPE tokenizer cannot handle these characters. In the end, we decided to prioritize having a single multilingual model for the task so we maintained our tokenizer choice.

3.2 Evaluating cross-lingual capabilities

To test the cross-lingual capability of a language, we evaluated its finetuned model to the validation set of another. Figure 3 shows the results.

We found that it is possible to adapt a language onto another for morphological annotation and lemmatization. However, this does not extend to POS tagging, as the validation set performs best only in the language it was trained on.

Some target languages tend to be cross-lingually receptive on lemmatization, i.e., many source languages can perform decently when applied to them. This observation is true for FRO, GOT, ISL, LAT, and LATM. Finally, there is also good cross-lingual compatibility between LAT and LATM—which is expected because they came from similar roots.

4 Conclusion

This paper describes Team 21a's system: a pretrained multilingual model (LIBERTUS) finetuned on different languages for each downstream task. Our system obtained [describe your rank, how you stack, etc. etc.]$_{\text{LJ}}$

Our system's main strength is its downstream performance. The validation scores are high for the majority of languages. However, it is limited by its tokenizer, resulting to poor performance on languages with non-latin scripts. We also evaluated each language's cross-lingual capability and showed that transfer learning is possible especially on lemmatization. This approach can be a viable alternative on limited corpora.

The source code for training and experimentation can be found on GitHub: [insert GitHub or OSF url]$_{LJ}$. We will also release the pretrained multilingual model and finetuned pipelines in public shortly after the review period.

Limitations

Use of BPE tokenizer:

Pretrained LM size:

References

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lester James Miranda, Ákos Kádár, Adriane Boyd, Sofie Van Landeghem, Anders Søgaard, and Matthew Honnibal. 2022. Multi hash embeddings in spaCy.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

A Appendix

A.1 Effect of different sampling strategies on pretraining performance

We also explored different sampling strategies and their effect on the pretraining loss curve. In our final system, we used the upsampling strategy as it provides clear benefit compared to others.

We examined the following configurations:

- **None:** we used the original dataset without any data sampling or augmentation.
- **Upsampling:** we upsampled each language to ensure that the number of their unique tokens is greater than or equal to the most dominant language.
- **Averaging:** we took the average number of unique tokens in the whole set and up-/downsampled each language based on this value.

Figure **??** shows that upsampling provides the clearest benefit for our pretrained model.

A.2 Finetuning a model per language vs. monolithic system

We also investigated if finetuning a model per language is more effective against a monolithic system, i.e., training on the full multilingual annotated corpora. Here, we combined the training corpora for all languages, then shuffled them before batching. This means that the downstream model sees a language mix per training epoch.

We found out that although more expensive, finetuning a model per language still yields the best results. Table **??** illustrates these results.

A.3 Alternative approach

We also considered using multi-hash embeddings (Miranda et al., 2022) as an alternative approach. Instead of pretraining, these embeddings use orthographic features (e.g., prefix, suffix, norm, shape) to create a word vector table. This approach also applies the hashing trick, inspired by Bloom filters (Bloom, 1970), to decrease the vector table's memory footprint.

Figure **??** shows the results in comparison to our final system. It is notable that simple orthographic features are competitive with our transformer-based model. However, we chose to submit the transformer-based pipeline as our final system because it still outperforms the multi-hash embed

method in the majority of our language-task pairs. We still recommend investigating this approach further because the hash-embed method has noticeable efficiency gains in terms of model size.