From English-Centric to Effective Bilingual: LLMs with Custom Tokenizers for Underrepresented Languages

Artur Kiulian[1], Anton Polishko[1], Mykola Khandoga[1], Yevhen Kostiuk[1,2], Guillermo Gabrielli[1], Łukasz Gągała[1,3], Fadi Zaraket[6], Qusai Abu Obaida[5], Hrishikesh Garud[4], Wendy Wing Yee Mak[7], Dmytro Chaplynskyi[8,9], Selma Belhadj Amor[7], Grigol Peradze[7]

[1]OpenBabylon, [2]ARG-Tech, University of Dundee, UK, [3]Georg-August Universität Göttingen, [4]Google, [5]Arab Center for Research and Policy Studies, [6]Doha Institute for Graduate Studies, [7]PolyAgent, [8]Ukrainian Catholic University, [9]lang-uk initiative

Abstract

In this paper, we propose a model-agnostic cost-effective approach to developing bilingual base large language models (LLMs) to support English and any target language. The method includes vocabulary expansion, initialization of new embeddings, model training and evaluation. We performed our experiments with three languages, each using a non-Latin script—Ukrainian, Arabic, and Georgian.

Our approach demonstrates improved language performance while reducing computational costs. It mitigates the disproportionate penalization of underrepresented languages, promoting fairness and minimizing adverse phenomena such as code-switching and broken grammar. Additionally, we introduce new metrics to evaluate language quality, revealing that vocabulary size significantly impacts the quality of generated text.

1 Introduction

The discovery of the Transformer architecture (Vaswani et al., 2017) has opened doors for creating large language models (LLMs) with billions of parameters, trained on datasets of trillions of tokens. One of the notable features of the LLMs is cross-lingual language understanding (XLU), which allows models to possess multilingual capabilities. However, the XLU ability is restricted by the so-called *curse of multilinguality*, which refers the difficulties and constraints encountered in creating multilingual LLMs. Studies showed that a substantial drop in performance occurs as the number of languages increases, due to the model's limited capacity to adequately capture and represent the nuances of each language (Conneau et al., 2020). The efforts to examine and address the problem have highlighted two key factors: **the composition of the dataset** and **vocabulary composition** (Pfeiffer et al., 2022; Blevins et al., 2024). Some studies (Chang et al., 2023) suggest that the natural limitations on the model capacity, vocabulary and training dataset sizes along with differences in language structures do not allow the creation of the ultimate multilingual model to perform equally in many languages, favoring the creation of custom models targeted at specific languages instead.

The most obvious yet often overlooked consequence of low language representation in a model's vocabulary is a much higher cost of language processing. A sentence in Ukrainian requires about 3 times more tokens for the GPT-4 model (et al., 2024) than the same sentence in English due to higher *tokenization fertility* (see Section 6.1). Three times higher fertility means three times smaller context window, three times higher memory usage, and nine times higher computation cost due to attention's quadratic dependence on the sequence length. On the other hand, high computational costs are not the only ramifications of a poor vocabulary. Recent studies (Rust et al., 2021a) indicate that representation in an LLM vocabulary of a specific language directly relates to the performance of the model in that language (Petrov et al., 2023). In particular, it may be a reason for the generation of non-existing words, code-switching (Winata et al., 2021; Zhang et al., 2023), and broken grammar. Languages that use a non-Latin alphabet are particularly affected by poor vocabulary representation since they cannot rely even on the overlapping tokens with better represented languages.

An insufficient training dataset affects the performance of LLMs as much as it does any other deep learning model. The model might generate a response in the wrong language, probably the one it is most familiar with, such as English (Marchisio et al., 2024). In this work, exposing the model to

Proceedings of the Fourth Ukrainian Natural Language Processing Workshop (UNLP 2025), pages 1–13

additional data in the target language via continual pre-training helped mitigate these effects.

In this paper, we present a model-agnostic resource-effective method to create a base bilingual LLM that supports English and another language. By addressing the above-mentioned issues of dataset and vocabulary composition, we make sure to improve its language capabilities along with boosting its computational efficiency. We illustrate our method in three languages with non-Latin alphabets: Ukrainian, Georgian, and Arabic.

The contributions of our work are as follows:

- We propose a vocabulary extension procedure that preserves the model's accumulated knowledge of English and extends the target language comprehension. The method is verified with Gemma 2 and Mistral models (see Section 3.1).
- We trained two separate bilingual LLMs (English-Ukrainian and English-Arabic) on language-specific datasets using the Mistral (Jiang et al., 2023) 7B model. The models were continually pre-trained for the next token prediction task on the parallel corpora for English and corresponding language. Our experiments showed that the proposed tokenization method reduces **computational complexity** and **inference time** for Ukrainian and Arabic respectively, while also improving model performance for code-switching and grammar correctness tasks. Additionally, we have conducted experiments to test the adoption of extended Georgian vocabulary for the English-Georgian model.
- We introduced new metrics for measuring code-switching and non-existing words ratio for Ukrainian and Arabic. The code-switching metric leverages the unique features of each language to detect instances of code-switching, following the rules of the respective languages.

2 Related Work

The shortcomings of existing multilingual LLMs have motivated numerous scholars and practitioners to address the insufficient performance of underrepresented languages.

Perhaps the most fundamental approach is to design and train a model from scratch, as demonstrated by EuroLLM (Martins et al., 2024). While this method offers maximal flexibility, it is highly demanding in terms of effort and computational resources.

More commonly, available open-source LLMs are used as a starting point, leveraging transfer learning and building on available weights (Tejaswi et al., 2024). This can still involve significant architectural changes compared to other methods, as seen in the SOLAR model (Kim et al., 2024). Despite utilizing transfer learning, such approaches often require pre-training on vast datasets, sometimes reaching trillions of tokens.

A number of publications (Cui et al., 2024; "hemanth kumar"; Nguyen et al., 2023; Vo, 2024) suggest a more lightweight approach, where the model's vocabulary is extended by 10,000–20,000 tokens, entailing the extension of the embedding layer and the language modeling head, while leaving the rest of the architecture unchanged. This method reduces the required training dataset to hundreds, or even tens, of billions of tokens, while still delivering notable improvements in the model's language abilities and computational efficiency.

Finally, instruction fine-tuning (Basile et al., 2023; Azime et al., 2024; Kohli et al., 2023) offers a highly resource-efficient alternative by skipping the base model composition step. While this approach can yield some improvements, it does not enhance the model's factual knowledge or address tokenization issues.

Our approach, in contrast, maintains the overall vocabulary size and keeps the model architecture intact. To create a bilingual model, we extend the vocabulary of the target language at the expense of other languages in the model, except English. This allows us to reduce the pre-training dataset to as little as 2 billion tokens while still improving the model's factual knowledge, enhancing the dataset, and achieving visible improvements in target language generation.

3 Methodology

Our proposed pipeline for training of bilingual LLMs supporting English and a target language $\mathcal{L}$ consists of the following steps:

1. **Vocabulary Extension**. The aim of this step is to create a new bilingual tokenizer T that retains the exact tokenization for English as in the original model, while incorporating an extended vocabulary for the target language $\mathcal{L}$, thus reducing fertility.

2. **Embeddings Initialization**. Initialize new embedding vectors for the newly added $\mathcal{L}$-specific tokens.

3. **Continual model pre-training**. In order to allow the model to adopt the new tokens and use them during the text generation we have continually pre-trained the model with new extended vocabulary.

Each step will be explained in more detail in the following subsections.

3.1 Vocabulary Extension Methodology

In this paper, we experimented with Mistral and Gemma 2 tokenizers, which have vocabulary sizes of 32,768 and 256,000 tokens respectively. Both models use SentencePiece tokenizers (Kudo and Richardson, 2018).

Our vocabulary extension technique can be described as follows. Consider the original tokenizer T_o that includes multilingual tokens. We trained a new tokenizer $T_\mathcal{L}$ for the target language $\mathcal{L}$ using a language-specific dataset. Next, the two tokenizer models are combined in order to obtain a bilingual tokenizer $T_{En-\mathcal{L}}$ that will be used during the training of the bilingual LLM. This is achieved via the following steps:

1. In order to keep the English tokenization intact we copy all the English tokens from the original tokenizer model T_o into bilingual tokenizer $T_{En-\mathcal{L}}$ along with their scores and IDs. We assumed that all tokens that contain only ASCII characters belong to English. We have also kept all the byte fallback tokens, control tokens (e.g. "[SEP]"), and service tokens (e.g. "[UNK]").

2. Tokens that belong in both T_o and $T_\mathcal{L}$ are assigned IDs from T_o and scores from $T_\mathcal{L}$. This procedure ensures tokenization according to the rules of $T_\mathcal{L}$ and at the same time allows the LLM to recognize familiar tokens of the target language $\mathcal{L}$ and to use the existing embeddings.

3. Lastly, the vocabulary of $T_{En-\mathcal{L}}$ is filled with new tokens from $T_\mathcal{L}$ ensuring that the vocabulary size matches the original tokenizer T_o.

The resulting bilingual tokenizer $T_{En-\mathcal{L}}$ is identical to T_o in the tokenization of the English language. On the other hand, in the target language, its fertility is improved thanks to the extended vocabulary (see Table 2).

3.2 Embeddings Initialization

Upon the vocabulary extension, the embedding vectors for the new tokens must be reinitialized. A proper embedding initialization can significantly improve the training convergence speed, while failing to do so might lead to a slower convergence or even non-convergence (Glorot and Bengio, 2010). In our experiments, we have tried a number of embedding initialization techniques, such as random, mean (Hewitt, 2021), FOCUS (Dobler and de Melo, 2023) and technique we called **NA**tural **CH**aracter **O**verlap **S**egmentation (NACHOS). We selected NACHOS because it has shown better convergence during training (see Appendix A). NACHOS works as follows. New tokens in $T_{En-\mathcal{L}}$ are expressed through the tokens that have already existed in the original tokenizer model T_o. Every longer token t_{new} can be split into a n of shorter tokens t: $t_{new} \rightarrow (t_1 ... t_n)$, with shorter tokens belonging to the overlapping vocabulary. We then initialize the embeddings of these new tokens by computing the mean of the shorter tokens embeddings (see Eq. 1):

$$E(t_{new}) = \frac{1}{n} \sum_{1}^{n} E(t_n), \tag{1}$$

where $E(t_{new})$ represents the embedding vector of the new token, $E(t_n)$ denotes the embeddings of the overlapping token t_n into which the new token is segmented.

3.3 Continual pre-training

As a final step, the newly composed model with the extended vocabulary and initialized embeddings is trained on the bilingual parallel corpora. This allows the model to fully adopt the new tokens, which we have verified by checking the token IDs of the model output. This process of new token adoption is put under scrutiny and discussed in detail in Section 7.2.

4 Datasets

Vocabulary Extension Datasets The monolingual language-specific tokenization models $T_\mathcal{L}$ have been trained on monolingual datasets. For the Ukrainian language we've trained on the publicly available UberText 2.0 (Chaplynskyi, 2023),

that contains 3.274B words and consists of 8.59M texts.

To train an Arabic tokenizer we have used a private dataset of non-fiction books of 430 million words based on (ACRPS). For Arabic, we integrated one more additional preprocessing step. As an Arabic word could correspond to several words in another language transmitting the same meaning, it is the best practice to perform light stemming to allow the models to pick the similarity of the semantics of the main parts of words (Larkey et al., 2002). For example, we consider ال (English translation: *the*) as a separate token when it prefixes a word. We processed attached pronouns and gender specifiers in similar way.

For our experiments with Georgian we have used the Georgian section of the public OSCAR dataset (OSCAR), which contains 171.9M words. This dataset has been used for both tokenizer training and continual pre-training of the English-Georgian Mistral model for token adoption experiments.

Continual Pre-training Datasets For continual pre-training we created parallel datasets, consisting of both English and target language.

For Ukrainian and Arabic, we considered Wikipedia parallel dataset dump from June 20th 2024 archive dump[1]. For Ukrainian, the size of the datasets is approximately 2B tokens. The total number of articles was 2.1M (791,336 in Ukrainian and 1,327,709 in English). The total number of Ukrainian tokens was 1.02B and the total number of English tokens was 1.05B. For Arabic, the size of the datasets is approximately 1.8B tokens. The total number of articles was 2.1B (1.2B in Arabic and 882,534 in English). The total number of Arabic tokens was 621.51M and the total number of English tokens was 1.1B. For Georgian token adoption experiments, we trained a model on parallel corpora from the same dump. The dataset was much smaller due to a sparsity of resources in Georgian. It contained 107,123 and 169,602 articles in English and Georgian, respectively. The total number of tokens was approximately 395.2M (219.88M in English and 175.32M in Georgian).

The articles were shuffled to create the training dataset with equal representation of the target language (Arabic or Ukrainian) and English. To determine the amount of tokens, we used the Gemma 2 tokenizer.

To evaluate the results, we used FLORES-200 (Team, 2022) dataset for corresponding languages. The dataset is a collection of parallel translation corpora for 200 distinct languages, including Ukrainian and Arabic. We selected 500 text samples per language from the "devtest" split of the dataset in Arabic and Ukrainian. Each text was separated into tokens by space, and only initial 3 tokens were kept as a model input. Finally, these inputs were provided to the model to generate a completion with a maximum generated sequence length of 128. For Ukrainian inputs, we obtained 1,500 tokens and 1,098 unique tokens. For Arabic inputs, we obtained 1,500 tokens and 1,000 unique tokens.

5 Experimental Setup

We continually pre-trained bilingual models on the next token prediction task on the parallel corpora utilizing HuggingFace (Wolf et al., 2019; Tunstall et al.) instructions for 8x80Gb GPUs. To launch training, we used the SkyPilot framework (Yang et al., 2023). In order to isolate the effects of extended vocabulary and additional pre-training we have conducted the same pre-training for the vanilla models and then compared the performances. For hyper-parameter optimization we used grid search. The selected set of hyper-parameters can be found in our GitHub repository.

6 Evaluation Metrics

Since we work on the base completion model, we focused mainly on the metrics that reflect the text completion performance: tokenizer fertility, code switching score, non-existing words ratio, and manually evaluated grammar correctness score.

6.1 Tokenizer Fertility

Fertility is the most common metric for evaluating tokenizer performance (Scao and et al., 2023; Rust et al., 2021b). This is an intrinsic metric of the tokenization model and is defined as the average number of tokens required to represent a word. For a tokenizer T and a dataset D, fertility is calculated by dividing the total number of tokens in $T(D)$ by the total number of words in D.

6.2 Non-Existing Words Ratio (NEWR)

We used a following heuristic to detect non-existent words generated by LLMs. A word is considered

[1] https://huggingface.co/collections/PolyAgent/parallel-datasets-6707e4197a737319934d2a48

non-existent if it is absent from a large language-specific corpus or vocabulary. For Ukrainian, we used the Ubertext fiction corpus (Chaplynskyi, 2023) to create a set of 2.6M unique words, mostly Ukrainian. Each generated word is checked against this set, and if absent, it is marked as non-existent. The Non-Existing Words Ratio (NEWR) was calculated as the percentage of non-existent words in the output for each language-specific LLM output.

Arabic requires more processing, as it is a language with several dialects associated with it. While each Arabic-speaking region has its own dialect, it significantly intersects with the modern standard Arabic (MSA), which is used in legal, news and other domains. While in this work we focused on MSA, dialectal words are often present in MSA. Therefore, we used the corpora associated with the Doha historical dictionary of Arabic (ACRPS)[2] to cover traditional Arabic (Albared et al., 2023), Aya Dataset (Singh and Vargus, 2024) to cover MSA, and Lisan corpora (Jarrar et al., 2023) to cover accepted dialectal words, 3.9M words in total.

6.3 Code Switching Word Ratio (CSWR)

In linguistics, code switching is a phenomenon, when a speaker uses (or "switches" between) two or more different languages in a conversation. To detect code switching in LLM outputs, we introduced a novel metric: Code Switching Word Ratio (CSWR). Unlike previous token-based methods (Marchisio et al., 2024), our approach uses language-specific rules to better identify code switching. The implementations are available in the GitHub repository[3].

CSWR is a ratio of words in the text that includes at least one foreign symbol (outside of the alphabet of the language, not a number or punctuation) and does not fit the rules of the correct code switching usage. The lower this ratio is - the better performance model showed from a code switching perspective.

The correct instances of code switching are detected depending on the language. A detailed explanation and a list of rules are provided in the Appendix B.

[2] https://dohadictionary.org/

[3] https://github.com/PolyAgent/PNaCoS-NER-Metric

6.4 Grammar Correctness Score (GCS)

To evaluate grammar correctness, the model generated text was evaluated by experts for the particular language on the following criteria: usage of incorrect words (e.g. wrong gender of the word, plural and single word form confusion, non-existing words, word merging, typos etc.), incorrect capitalization and punctuation and instances of incorrect code switching. If any of those flaws were encountered by the annotator the score of 0 was assigned to the text. If the text passes the check, it was assigned the score of 1. Finally, the **Grammar Correctness Score (GCS)** is calculated as an average of all assigned scores for the test completions.

For each language (Ukrainian and Arabic) we employed three native speakers annotators.

7 Results

7.1 Tokenizer Intrinsic Performance

The comparison of the original model tokenizer with the customized bilingual tokenizers developed by us via the procedure described in Section 3.1 can be found in Table 2. Besides Mistral with its 32,768 tokens in the vocabulary we have also experimented with Gemma 2, which has a vocabulary 8 times larger. That has allowed us to substantially extend the target language vocabulary without changing the model architecture. Naturally, in every case the extended vocabulary has improved the tokenization fertility in the target language, allowing the model to process the same amount of text at lower computational cost. The non-linear fertility improvement is expected due to the logarithmic character of its dependence on the vocabulary size (Tao et al., 2024).

Ukrainian In the case of the Ukrainian language, it was challenging to estimate the exact number of the language-specific tokens in the original vocabulary due to possible confusions with other languages that use the Cyrillic alphabet. The number presented in the Table 2 is a lower estimate. Fertility has been measured with 13 million words from the Ukrainian section of the OSCAR dataset. Notably in the case of Gemma 2 we have developed a tokenizer that ensures comparable fertility for the English and Ukrainian languages, thus reaching parity between the two (1.52 for Ukrainian and 1.53 for English). Parallel fertility has been measured using the Macocu parallel English-Ukrainian dataset (Bañón et al., 2023).

Arabic For the Arabic language, fertility was measured using a stemmed dataset (see Section 4). Due to this, the numerical fertility results for Arabic differ from those of the other languages and can't be directly compared to them.

Georgian The original Mistral vocabulary did not cover 6 letters from the Georgian alphabet, which has forced the model to resort to byte fallback (see also Section 7.2), which affected the original model's fertility in Georgian. Extending the vocabulary by 5,500 tokens has allowed to improve token usage by nearly three times. Due to Georgian dataset size limitations we were not able to properly train and evaluate a Gemma-compatible tokenizer for the Georgian language.

7.2 Token Adoption Process

In this subsection, we investigate the token composition of the Mistral model output during the continual pre-training that followed the vocabulary extension for Ukrainian (Mean initialization), Georgian (NACHOS initialization), and Arabic languages respectively. The output tokens have been split into 5 categories:

- Existing: tokens of the target language that exist in the default Mistral vocabulary.
- New: tokens of the target language that were added to the vocabulary.
- English: tokens used to represent English.
- Byte-encoded: 256 byte fallback tokens used to encode characters absent in the vocabulary in UTF-8 format.
- Other: tokens that do not belong to any of the above-mentioned categories (e.g. tokens of other languages, punctuation, etc.).

On Figure 1, Y axis of the plot corresponds to the relative fraction of the tokens in each category (all categories sums up to 1). In general, we observed similar phenomena in all three languages. Being prompted in a target language, the original Mistral model is likely to produce a response in English, most probably due to insufficient pre-training on the target language corpus. Once our pre-training starts, the model learns to produce responses in the target language and after a few hundred training steps it outputs little to no English tokens.

At first, the model favors the usage of familiar tokens that already exist in its vocabulary before the extension. Subsequent pre-training teaches the model to use the new tokens along with the familiar ones. After 2,000 training steps, the process stabilizes and becomes nearly static between 5,000 and 10,000 steps.

The same pattern holds in all three of the considered languages, though with some differences which we would like to discuss in more detail. We experimented with Ukrainian, Georgian, and Arabic.

Ukrainian Ukrainian is much better represented in Mistral model than Arabic and Georgian. The original Mistral vocabulary contains 1,731 Cyrillic tokens, with about 1,600 of them suitable for the Ukrainian language representation. The original model occasionally replies in English if prompted in Ukrainian, producing about 35% of English tokens in the output. Upon the start of the pre-training the model learns to use Ukrainian tokens, though initially the model tends to use the existing Ukrainian tokens. After 200 training steps, this ratio increases to about 65%. With more training, this number drops to 50%, indicating that the model fully adopted new tokens. However, despite the new tokens make about 75% of the extended Ukrainian vocabulary, the fraction of existing tokens remains dominant due to higher frequency of occurrence.

Arabic Qualitatively, the situation with the Arabic language is similar to that of the Ukrainian, but with two important differences. When prompted in Arabic, original Mistral is more likely to respond in English, with the fraction of produced English tokens reaching 60%. In the original Mistral vocabulary there is 70 Arabic tokens, which is enough to avoid byte fallback, but is still a relatively small number. That is why the fraction of the new tokens overtakes as early as 200 training steps and remains dominant afterwards.

Georgian There are 29 Georgian tokens in the original Mistral vocabulary, which does not even cover the Georgian alphabet (35 letters). That forces the model to resort to byte fallback when generating text in Georgian more frequent than in Ukrainian or Arabic. The fraction of the byte encodings grows when the model learns to respond in Georgian and then drops along with the adoption of the new tokens, similarly to previously discussed languages. In case if Georgian, the token adaptation takes longer, as the model resorts to using

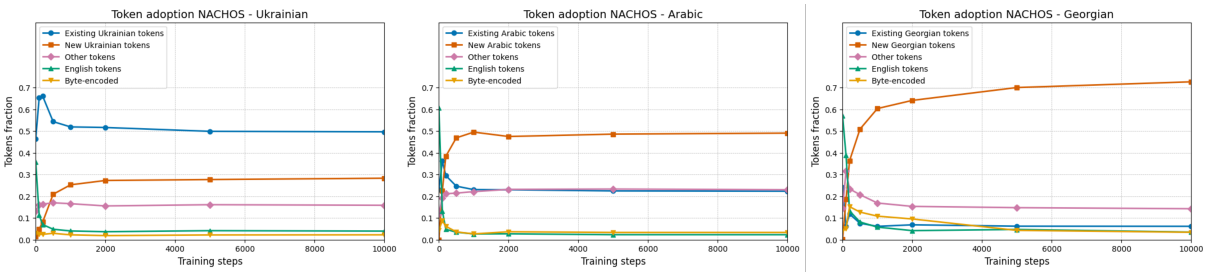

Figure 1: Tokens adoption by Mistral model.

Model	GCS ↑	NEWR ↓	CSWR ↓
Ukrainian			
Vanilla	0.264	0.089	0.515
Tuned	0.388	0.032	0.002
Ours	**0.503**	**0.030**	**0.001**
Arabic			
Vanilla	0.040	0.863	0.450
Tuned	0.238	0.079	0.004
Ours	**0.548**	**0.050**	**0.002**

Table 1: Results for trained model of Grammar Correctness Score (GCS), Non-Existing Words Ratio (NEWR), and Code Switching Word Ratio (CSWR). "Vanilla" refers to the original Mistral 7b model without additional training, "Tuned" refers to the continually pre-trained Mistral model on the same datasets, "Our" refers to Mistral continually pre-trained with extended vocabulary. ↑ indicates that bigger value is better. ↓ indicates that lower value is better.

the byte encodings for the text prediction while learning new tokens. Byte encodings are always encoded with a pair of tokens and that might explain a longer period of adopting the new Georgian tokens.

7.3 Performance Metrics

The results for the trained model of Grammar Correctness Score (GCS), Non-Existing Words Ratio (NEWR), and Code Switching Word Ratio (CSWR) are presented in Table 1.

The results showed that the model trained with our approach outperformed both Vanilla and Tuned models in terms of GCS in Ukrainian and Arabic. Notably, the vanilla model struggled with grammatical accuracy, achieving a score of 0.264 on Ukrainian compared to the our model's score of 0.503. Tuned English-Ukrainian model achieved GCS of 0.388. For Arabic, tuned model achieved 0.238 and 0.04 for the vanilla model, demonstrating lack of grammatical knowledge. Our model achieved GCS score of 0.548.

Our method demonstrated NEWR of 3%, which is not significantly different from the score of the tuned model (3.2%) for Ukrainian. The reason for such similarity could be in a better representation of Ukrainian tokens in Mistral (see Figure 1). Vanilla model showed 8.9% of non-existing words in its generated texts. On the other hand, for Arabic our approach obtained NEWR of 5%, when vanilla and tuned models obtained 86.3% and 7.9% respectively. The vanilla model's performance was really poor when it comes to generating existing modern Arabic words. The tuning improved the performance in more than 10 times, but our model outperformed it.

Finally, we achieved a score of 0.001 for CSWR for Ukrainian, which indicates a very little incorrect usage of foreign languages in the text. The second best score was obtained for tuned model (0.002). The vanilla model performed significantly worse: 0.515, indicating that more than half of generated words are used incorrectly in terms of code switching. For Arabic, the situation is similar. Our model obtained a score of 0.002, outperforming tuned model (0.004) and vanilla model (0.45).

7.4 Preventing catastrophic forgetting in English

After a series of experiments, we found that after just 1 epoch of training on the bilingual corpora, the models showed improvement in the target language but experienced a substantial drop in the English MMLU benchmark (Hendrycks et al., 2021b,a). However, by lowering the learning rate from $1.5e-5$ to $2e-6$, training resulted in a much smaller loss in MMLU benchmark points. These important results demonstrate that, with the right training, the model can retain its English performance and remain bilingual, as shown in Table 3.

Mistral	Vanilla		*Ours*	
	Tokens	**Fertility**	**Tokens**	**Fertility**
Ukrainian	1,077	3.35	5,552	2.55
Arabic*	70	3.3	3,618	1.68
Georgian	29	7.61	5,531	2.68
Gemma	Vanilla		*Ours*	
	Tokens	**Fertility**	**Tokens**	**Fertility**
Ukrainian	6,426	2.55	75,704	1.56
Arabic*	6,075	1.65	32,333	1.52

Table 2: Tokenization metrics. *Stemmed tokenization for Arabic.

Model	**GCS↑**	**NEWR ↓**	**CSWR↓**	**MMLU↑**
Vanilla	0.26	0.09	0.52	0.59
Tuned	0.39	0.03	2e-3	0.34
Ours	0.50	0.03	1e-3	0.25
Tuned†	0.31	0.03	2e-3	0.49
Ours†	0.42	0.03	9e-4	0.507

Table 3: Retention of the MMLU performance in the English-Ukrainian models trained with low learning rate (denoted with †).

8 Discussion

The obtained results highlight a subject that has been largely overlooked, particularly in the context of generative LLMs: the impact of vocabulary size and composition an on the quality of generated text.

Our experiments with the vanilla model pre-training demonstrated that the effects of training on additional data can be mitigated via the vocabulary extension. Additional pre-training on the target language corpus can noticeably increase text quality, particularly in addressing issues like code-switching and the generation of non-existent words. However, handling more complex linguistic features, such as grammar, requires vocabulary extension. Ukrainian and Arabic tokens are represented differently in the original model's vocabulary, resulting in distinct yet complementary outcomes for the two languages. While for Ukrainian a substantial 29.6% improvement was obtained with the extended vocabulary, the severely underrepresented Arabic achieves a much higher 90.5% improvement. This effect was confirmed with another round of training at a lower learning rate for the English-Ukrainian models, which showed a 35% improvement utilizing the model vocabulary extension.

We propose the following explanation for this phenomenon: a poor vocabulary results in tokens that contain only one or a few characters, conveying very little specific semantic meaning. As a result, the model is forced to rely heavily on context during training and inference. This increases the noisiness of the data and prevents the model from learning nuanced meanings or effectively constructing complex grammatical structures.

Unfortunately, a static and limited vocabulary with fixed token-to-embedding mappings is a limitation of the standard transformer architecture. This makes it challenging to create a transformer-based LLM that is equally proficient in multiple distinct languages. Some methods that utilized char-based (CANINE (Clark et al., 2022)), patch-based (MegaByte (Yu et al., 2023)), or byte-based (Pagnoni et al., 2024) transformers were suggested. They often suffer from longer sequences, reduced linguistic abstraction, and increased computational cost, which can hinder downstream performance compared to efficient BPE-based tokenization.

For this reason, we advocate training bilingual models, which are both cost-effective and proficient in their target languages.

9 Conclusions

In this work, we introduced a model-agnostic, cost-effective method for developing bilingual base completion LLMs that support English and a target language, including low-resource or underrepresented languages. Our approach, centered on vocabulary extension and efficient embedding initialization, was validated by creating two bilingual LLMs: English-Ukrainian and English-Arabic. Moreover, we conducted experiments with Georgian tokenization and explored token adoption process during the training of a English-Georgian model. Georgian has a unique underreprsentation in the Mistral tokenizer.

We demonstrated that extending the vocabulary of a pre-trained model enhances its performance in target language while maintaining its English performance. Specifically, the grammar correctness results indicate that pre-training alone provides only limited improvement. The comparison between Ukrainian and Arabic further emphasizes the limitations of poor vocabulary for the underrepresented language. Expanding the tokenizer's vocabulary with target language tokens reduced tokenizer fertility, resulting in lower computational costs and improved processing efficiency. Finally, retaining the original English tokens in the custom tokenizer while adding new language-specific tokens lead to

preservation of the model's English performance on the MMLU benchmark, while also improving its performance in the target language from perspective of grammar, code switching and non-existant word ratios.

Our approach promotes a more equitable and inclusive NLP ecosystem, contributing to the revitalization of underrepresented languages. By lowering the barrier to developing more literate and grammatically capable models, we believe our work also paves the way for enhanced economic viability of using LLMs in non-English languages.

10 Limitations

In this work, we have focused on creating a minimal working example of a base bilingual model with an extended vocabulary in a cost-effective way. While our approach is model-agnostic, it has yet to be tested with models other than Mistral 7B. Gemma 2 is the most likely candidate, as we have already concluded tokenizer experiments. However, applying the method to other open-source models, such as Llama 3 or Qwen, would provide further validation for our approach.

Another important limitation is that the method was eventually tested only for English-Ukrainian and English-Arabic models. Due to the limited availability of Georgian corpora, we were unable to complete the experiment with the English-Georgian model.

The retention of English language capabilities has only been tested with the English-Ukrainian model. We are currently in the process of testing it for the English-Arabic model.

Further experiments with the vocabulary size and composition could help to find the optimal parameters along with their dependence on the available dataset size and individual language properties.

To fully evaluate the model across a variety of downstream tasks, such as machine translation, question answering, summarization, or text completion, instruction tuning will be required. This step, however, goes beyond the scope of our current work.

While we believe that the proposed metrics for assessing the language quality are an important step, they leave enough space for refinement. In particular, the code-switching metric for Ukrainian and Arabic might benefit from implementing additional rules. In our evaluation we did not test on downstream tasks like machine translation, summarization, QA etc.

Acknowledgements

We would like to express our gratitude to the following organizations for their generous support, which made this work possible:

- **Observea** for providing access to a 16xTesla H100 cluster, which significantly enhanced our computational resources.
- **NVIDIA** for providing DGX Workstation with 4xTesla V100 used for inference and evaluations.
- **HotAisle** for granting access to the 8xAMD MI300x node, enabling critical model training experiments.
- **AWS** for offering cloud credits that supported the use of Tesla H200 instances for training.
- **GCP (Google Cloud)** for providing credits used for model training and inference.
- **TPU Research Cloud (Google Cloud)** for providing TPU VMs for our training pipeline experiments.
- **A.I. Hero** for providing access to a 8xA100 instance for our original set of experiments.
- **Doha Graduate Studies University** and the **Arab Center for Research and Policy Studies** for being key strategic collaborators, whose guidance and expertise greatly contributed to the development of this work.

All of the contributions above were instrumental in achieving the results presented in this paper, and we look forward to continued collaborations.

Author Contributions

Anton Polishko, Mykola Khandoga, and Yevhen Kostiuk led the core model development and experimentation. Artur Kiulian supervised the project and coordinated the collaboration. Guillermo Gabrielli, Łukasz Gągała, and Wendy Wing Yee Mak contributed to data preprocessing and evaluation pipeline implementation. Fadi Zaraket, Qusai Abu Obaida, and Selma Belhadj Amor were responsible for Arabic language integration and evaluation. Hrishikesh Garud supported the engineering infrastructure. Dmytro Chaplynskyi contributed

Ukrainian language resources. Grigol Peradze provided Georgian language resources and analysis.

All authors reviewed and approved the final manuscript.

References

ACRPS. Doha historical dictionary of arabic.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Doha Albared, Hadi Hamoud, and Fadi Zaraket. 2023. Arabic topic classification in the generative and AutoML era. In *Proceedings of ArabicNLP 2023*, pages 399–404, Singapore (Hybrid). Association for Computational Linguistics.

Israel Abebe Azime, Mitiku Yohannes Fuge, Atnafu Lambebo Tonja, Tadesse Destaw Belay, Aman Kassahun Wassie, Eyasu Shiferaw Jada, Yonas Chanie, Walelign Tewabe Sewunetie, and Seid Muhie Yimam. 2024. Enhancing amharic-llama: Integrating task specific and generative datasets. *Preprint*, arXiv:2402.08015.

Pierpaolo Basile, Elio Musacchio, Marco Polignano, Lucia Siciliani, Giuseppe Fiameni, and Giovanni Semeraro. 2023. Llamantino: Llama 2 models for effective text generation in italian language. *Preprint*, arXiv:2312.09993.

Marta Bañón, Malina Chichirau, Miquel Esplà-Gomis, Mikel L. Forcada, Aarón Galiano-Jiménez, Cristian García-Romero, Taja Kuzman, Nikola Ljubešić, Rik van Noord, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Peter Rupnik, Vít Suchomel, Antonio Toral, and Jaume Zaragoza-Bernabeu. 2023. Ukrainian-english parallel corpus macocu-uk-en 1.0. `http://hdl.handle.net/11356/1858`. ISSN 2820-4042.

Terra Blevins, Tomasz Limisiewicz, Suchin Gururangan, Margaret Li, Hila Gonen, Noah A. Smith, and Luke Zettlemoyer. 2024. Breaking the curse of multilinguality with cross-lingual expert language models. *Preprint*, arXiv:2401.10440.

Tyler A. Chang, Catherine Arnett, Zhuowen Tu, and Benjamin K. Bergen. 2023. When is multilinguality a curse? language modeling for 250 high- and low-resource languages. *Preprint*, arXiv:2311.09205.

Dmytro Chaplynskyi. 2023. Introducing UberText 2.0: A corpus of modern Ukrainian at scale. In *Proceedings of the Second Ukrainian Natural Language Processing Workshop*, pages 1–10, Dubrovnik, Croatia. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

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.

Yiming Cui, Ziqing Yang, and Xin Yao. 2024. Efficient and effective text encoding for chinese llama and alpaca. *Preprint*, arXiv:2304.08177.

Konstantin Dobler and Gerard de Melo. 2023. Focus: Effective embedding initialization for monolingual specialization of multilingual models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

OpenAI et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

"hemanth kumar". Tamil-mistral-7b-v0.1.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

John Hewitt. 2021. Initializing new word embeddings for pretrained language models.

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

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Kyiv, Ukraine (Online). Association for Computational Linguistics.

Mustafa Jarrar, Fadi A. Zaraket, Tymaa Hammouda, Daanish Masood Alavi, and Martin Wählisch. 2023. Lisan: Yemeni, iraqi, libyan, and sudanese arabic dialect corpora with morphological annotations. In *20th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2023, Giza, Egypt, December 4-7, 2023*, pages 1–7. IEEE.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2024. Solar 10.7b: Scaling large language models with simple yet effective depth up-scaling. *Preprint*, arXiv:2312.15166.

Guneet Singh Kohli, Shantipriya Parida, Sambit Sekhar, Samirit Saha, Nipun B Nair, Parul Agarwal, Sonal Khosla, Kusumlata Patiyal, and Debasish Dhal. 2023. Building a llama2-finetuned llm for odia language utilizing domain knowledge instruction set. *Preprint*, arXiv:2312.12624.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *Preprint*, arXiv:1808.06226.

Wuwei Lan, Yang Chen, Wei Xu, and Alan Ritter. 2020. GigaBERT: Zero-shot transfer learning from English to Arabic. In *Proceedings of The 2020 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Leah S. Larkey, Lisa Ballesteros, and Margaret E. Connell. 2002. Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, page 275–282, New York, NY, USA. Association for Computing Machinery.

Kelly Marchisio, Wei-Yin Ko, Alexandre Bérard, Théo Dehaze, and Sebastian Ruder. 2024. Understanding and mitigating language confusion in llms.

Pedro Henrique Martins, Patrick Fernandes, João Alves, Nuno M. Guerreiro, Ricardo Rei, Duarte M. Alves, José Pombal, Amin Farajian, Manuel Faysse, Mateusz Klimaszewski, Pierre Colombo, Barry Haddow, José G. C. de Souza, Alexandra Birch, and André F. T. Martins. 2024. Eurollm: Multilingual language models for europe. *Preprint*, arXiv:2409.16235.

Mohamed Megahed. 2021. Sequence labeling architectures in diglossia - a case study of arabic and its dialects.

Quan Nguyen, Huy Pham, and Dung Dao. 2023. Vinallama: Llama-based vietnamese foundation model. *Preprint*, arXiv:2312.11011.

OSCAR. Oscar.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. 2024. Byte latent transformer: Patches scale better than tokens. *Preprint*, arXiv:2412.09871.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Preprint*, arXiv:2305.15425.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021a. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021b. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Teven Le Scao and Angela Fan et al. 2023. Bloom: A 176b-parameter open-access multilingual language model. *Preprint*, arXiv:2211.05100.

Shivalika Singh and Freddie et al Vargus. 2024. Aya dataset: An open-access collection for multilingual

instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11521–11567, Bangkok, Thailand. Association for Computational Linguistics.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. 2024. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *Preprint*, arXiv:2407.13623.

NLLB Team. 2022. No language left behind: Scaling human-centered machine translation.

Atula Tejaswi, Nilesh Gupta, and Eunsol Choi. 2024. Exploring design choices for building language-specific llms.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Shengyi Huang, Kashif Rasul, Alvaro Bartolome, Alexander M. Rush, and Thomas Wolf. The Alignment Handbook.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

James Vo. 2024. Vi-mistral-x: Building a vietnamese language model with advanced continual pre-training. *Preprint*, arXiv:2403.15470.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? *CoRR*, abs/2103.13309.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. arxiv. *arXiv preprint arXiv:1910.03771*.

Zongheng Yang, Zhanghao Wu, Michael Luo, Wei-Lin Chiang, Romil Bhardwaj, Woosuk Kwon, Siyuan Zhuang, Frank Sifei Luan, Gautam Mittal, Scott Shenker, and Ion Stoica. 2023. SkyPilot: An intercloud broker for sky computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 437–455, Boston, MA. USENIX Association.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. *Preprint*, arXiv:2305.07185.

Ruochen Zhang, Samuel Cahyawijaya, Jan Christian Blaise Cruz, Genta Indra Winata, and Alham Fikri Aji. 2023. Multilingual large language models are not (yet) code-switchers. *Preprint*, arXiv:2305.14235.

A Embedding Initialization Comparison

In the Table 4 the metrics for the different languages and embedding initializations are presented. The graph of training and evaluation losses are presented on the Figure 3 and 2.

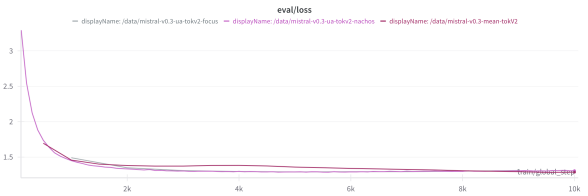

Figure 2: Ukrainian evaluation graph per training step. The name includes the embedding initialization technique: mean, residual, and NACHOS.

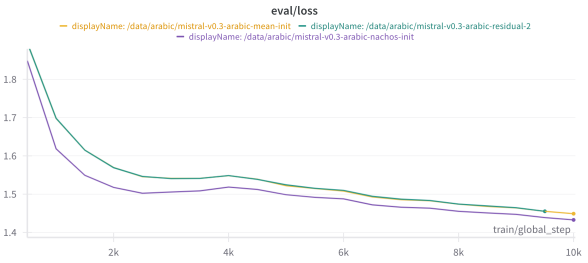

Figure 3: Arabic evaluation graph per training step. The name includes the embedding initialization technique: FOCUS, NACHOS, and mean.

Model	NEWR↑	CSWR↓
Vanilla	0.9118	0.5156
Tuned	0.9667	0.0006
Mean	0.9667	0.0009
NACHOS	0.9665	0.0009
FOCUS	0.9634	0.0011

Table 4: Comparison of Model Performance on NEWR and CSWR Metrics

In our experiments, NACHOS demonstrated a better convergence compared to other methods, however the performance results for the final models were similar. As complete evaluation is computationally expensive and requires manual annotation, we decided to continue only with NACHOS approach.

B Code Switching scoring rules

To calculate the score for each language, the same initial preprocessing for the generated text was applied: the accents were replaced with regular corre-

sponding letters and HTML formatting tags were removed.

2.0.1 Ukrainian CSWR Rules

In Ukrainian, the usage of code switching is allowed if it respects the following rules. All the mentions of the following entities are allowed in a foreign language:

- Proper names: names of the music bands, locations, restaurants, libraries, cities, titles, identification numbers etc. For example, **Pythagoras**, **California**, **MIT**, **Metallica**, **F-16** and so on.
- Medical terms, additives and vitamins. For example, (vitamin) **B12**, (food additive) **E110** etc.
- Roman integers and math symbols. For example, II, X, $\sum_{i=1}^{N}$, etc.
- Quotes. If text is a direct quote, it can be used in Ukrainian without translation, marked with the special symbols.
- URL links, hashtags, encoding names, mentions of the most common file formats and filenames. For example, **PDF**, **my_cv.pdf**, **mydog.png**, **https://www.wikipedia.org**, **UTF-8**, **#Euro2012** and so on.
- Common Latin phrases. Some of the well-known Latin sayings and quotes can be used as is if they are widely known. For example, **Veni, vidi, vici**, **A priori** etc.

To accommodate these rules, our metric utilizes an ensemble of named entity recognition (NER) models as well as a rule-based approach to pick up the correct usage of foreign words or symbols. In particular, we have used XML-based Ukrainian NER model[4], SpaCy (Honnibal et al., 2020) uk_core_news_lg[5] model, and Stanza (Qi et al., 2020) Ukrainian model. All the URL links, Roman integers, math symbols, and text in quotation marks were extracted as separate named entities with the regular expressions. Finally, each sentence were checked if it contained any char in Ukrainian. If it did not and the whole sentence was not considered to be a named entity, the whole sentence and words in it were considered as incorrect.

To accommodate medical terms, additives and vitamins usage rule, we manually extracted a list of them from the US Food and Drug Administration[6], as they can be used in Ukrainian language as well without translation. The total number of terms is 2,729.

To extract encoding names, file formats and filenames, and widely recognised Latin phrases, we manually retrieved them from Wikipedia. We obtained a list of 79 encoding names, 1,995 file formats, and 2,373 Latin phrases.

All of the resources are available on our GitHub repository[7].

2.0.2 Arabic CSWR Rules

Arabic follows the following rules.

- Arabic does not have capital letters which renders named entity detection especially for proper names a specialized task.
- In Arabic, both Indian or Arabic numerals can be used.
- Some Arabic characters are non-connecting characters and are written separately from the next word, even if there is no space between them. Arabic is written right to left, but Arabic words followed by non-Arabic words written in the other direction (sometimes with no white space separation).

To address these issues, we utilized a different ensemble of NER models, specifically Flair (Akbik et al., 2019) pre-trained Arabic NER model (Megahed, 2021)[8], transformer-based Arabic NER models (Lan et al., 2020; Inoue et al., 2021)[9], and Stanza (Qi et al., 2020) Arabic model. Resources and algorithms to identify medical terms, additives, vitamins, hashtags, encoding names, URL links, file formats, roman integers and quotes are the same as we introduced in the Ukrainian Code Switching Metric.

[4] https://huggingface.co/EvanD/xlm-roberta-base-ukrainian-ner-ukrner

[5] https://spacy.io/models/uk#uk_core_news_lg

[6] https://www.fda.gov/food/food-additives-petitions/food-additive-status-list

[7] https://github.com/PolyAgent/PNaCoS-NER-Metric

[8] https://huggingface.co/megantosh/flair-arabic-multi-ner

[9] https://huggingface.co/ychenNLP/arabic-ner-ace, https://huggingface.co/CAMeL-Lab/bert-base-arabic-camelbert-mix-ner