Specialized Monolingual BPE Tokenizers for Uralic Languages Representation in Large Language Models

Iaroslav Chelombitko
Neapolis University Pafos
Paphos, Cyprus
i.chelombitko@nup.ac.cy

Aleksey Komissarov
Neapolis University Pafos
Paphos, Cyprus
ad3002@gmail.com

Abstract

Large language models show significant inequality in language representation, particularly for Uralic languages. Our analysis found that existing tokenizers allocate minimal tokens to Uralic languages, highlighting this imbalance. To address this, we developed a pipeline to create clean monolingual datasets from Wikipedia articles for four Uralic languages. We trained Byte Pair Encoding (BPE) tokenizers with a vocabulary size of 256,000 tokens, though Northern Sami had only 93,187 due to limited data.

Our findings revealed most tokens are unique to each language, with 8,102 shared across all four, and 25,876 shared among Estonian, Finnish, and Hungarian. Using the Compression Ratio metric, our tokenizers outperformed popular ones like LLaMA-2 and Gemma 2, reducing Finnish's compression ratio from 3.41 to 1.18.

These results demonstrate the importance of specialized tokenizers for underrepresented languages, improving model performance and lowering costs. By sharing our tokenizers and datasets, we provide crucial resources for further research, emphasizing the need for equitable language representation.

1 Introduction

Large language models have rapidly integrated into our daily lives, yet they exhibit significant inequality in language representation due to issues of digital vitality. This imbalance is particularly pronounced in the Uralic language family, which includes well-represented languages like Finnish, Estonian, and Hungarian, as well as underrepresented ones like Northern Sami. Our initial assessment revealed that existing tokenizers inadequately represent Uralic languages, with a minimal number of tokens allocated to them.

However, there is significant inequality in language representation at all stages due to the issue of digital vitality (Acs et al., 2017; Zaugg et al., 2022), where there is an insufficient amount of digitized text available (Arkhangelskiy, 2019). As a result, training datasets are dominated by texts in popular languages, leading to an imbalance (Choi et al., 2023). It's not just the training datasets; the same problem exists with tokenizers (Petrov et al., 2023), as was shown earlier. The efficiency of a tokenizer directly impacts performance in that language and, consequently, the cost of tasks, since pricing is tightly linked to the number of tokens.

On the other hand, new methods are emerging that allow adjustments to the tokenizer's vocabulary during fine-tuning, and even better, future models could pay closer attention to token representation in tokenizers during training (Downey et al., 2024). However, to achieve this, a resource is needed that enables such adjustments (Alnajjar et al., 2023; Paul et al., 2024). The approach we propose for creating monolingual tokenizers serves as exactly this kind of resource. Uralic languages provide a useful model, as the group is relatively small and includes three languages that are comparatively well represented on Wikipedia, one that is poorly represented, and a significant number of languages that are not represented at all.

First, we decided to assess how many Uralic language tokens are present in existing tokenizers. After evaluating the token representation, we realized that these languages are almost entirely absent. This led us to the question: can we train a BPE tokenizer both for individual Uralic languages and for all Uralic languages combined?

We encountered the challenge that detecting Uralic languages, beyond Finnish, Estonian, and Hungarian, is a complex task. For the tokenizer, we used Wikipedia articles from four languages: in addition to the three mentioned, we also included Northern Sami. We developed a pipeline to extract the cleanest monolingual texts for these four languages, and based on these texts, we trained four

Proceedings of the 9th International Workshop on Computational Linguistics for Uralic Languages, pages 89–95
November 28-29, 2024 ©2024 Association for Computational Linguistics

monolingual tokenizers with a size of 256K, as well as one tokenizer for the combined languages.

Although the initial goal was to create monolingual tokenizers, the tokenizers themselves turned out to be an interesting new tool for comparative analysis. They provide an additional perspective for approaching many classic tasks (Toraman et al., 2023).

We have made both the tokenizers and the clean monolingual Wikipedia datasets publicly available. These datasets can also serve as a foundation for other research projects.

Finally, we evaluated how much more effectively our proposed tokenizers perform compared to existing ones. We also assessed the impact of tokenizer size on tokenization efficiency and proposed methods for determining how many additional tokens need to be added to a tokenizer for it to effectively handle Uralic languages.

2 Results and Methods

2.1 Assessing the Representation of Uralic Languages in Existing Tokenizers

Our first task was to assess how well Uralic languages are represented in existing tokenizers. To study the representation of Uralic languages within the tokenizers of popular LLMs, we first classified each token according to the language it belongs to. Table 1 presents the Uralic language encodings from the key standards.

The ISO 639-1 standard encompasses three languages: Estonian, Finnish, and Hungarian. Expanding to broader classifications, the ISO 639-2 standard includes nine languages, while ISO 639-3 covers twenty-seven languages. Additionally, the Common Locale Data Repository (CLDR) incorporates sixteen Uralic languages. This hierarchical structuring across multiple standards highlights the varying levels of granularity and coverage, underscoring the importance of comprehensive language encoding for effective NLP applications in the Uralic language family. While our findings are promising, we are currently unable to accurately assess the representation of these languages in the Common Crawl dataset, which constitutes a substantial portion of the training data for foundation models. The seemingly large number of Uralic languages represented in standards like ISO 639-3 is offset by the fact that most language classifiers do not use this standard. To compare the classifications, we use two popular libraries for language identification

Tokenizers	llama-2	llama-3	gemma
fi	415	875	3518
et	429	1082	3335
hu	711	1359	4456

Table 1: Langid numbers of tokens for llama-2, llama-3, and gemma tokenizers across three languages.

Tokenizers	llama-2	llama-3	gemma
fi	202	779	1905
et	251	1145	2066
hu	403	1115	2570

Table 2: Cld3 numbers of tokens for llama-2, llama-3, and gemma tokenizers across three languages.

— langid.py (Lui and Baldwin, 2012) and pycld3 (Ooms, 2024) The langid.py classifier is based on the ISO 639-1 standard, which significantly limits the classification of Uralic languages, narrowing it down to Finnish, Estonian, and Hungarian. The pycld3 classifier uses the CLDR standard, which offers an apparent advantage over langid.py and ISO 639-1 when it comes to Uralic language classification. However, none of the existing classification tools are comprehensive, as they have considerable biases when classifying short tokens and underrepresented languages (Chelombitko et al., 2024). To highlight this issue, we provide a comparison of the number of Uralic language tokens in the tokenizers of top models with publicly available tokenizers in Table 1 and 2.

2.2 Creating Monolingual Datasets Based on Wikipedia Articles

We downloaded Wikipedia dumps in ZIM format for four languages—Estonian, Finnish, Hungarian, and Northern Sami. The links are organized by language and further classified by size, such as "nopic" or "maxi," and date. Specifically, links related to the "all" topic are added to the data structure. For each language, we prioritize the "nopic" size if it is available. If the "nopic" size is not available, the "maxi" size is used instead. Within each size category, the links are sorted by date to identify the most recent links. If a link from May 2024 is available, it is given priority. For Finnish, no link from May 2024 was available. Therefore, we used the most recent link from May 2023.

Downloaded zim files were processed with a custom C++ program available in github. It iterates through each entry in the archive by path. If

an entry is identified as a redirect, it is skipped. The script parses the HTML content using an HtmlParser library. The script looks for elements with the class name "mw-parser-output" within the parsed HTML document. If such elements are found, the script further extracts all HTML paragraph elements within the "mw-parser-output" element. For each paragraph element found, the script outputs the HTML content of the paragraph. Finally we used a custom python script to clean HTML paragraphs of the text. The script reads input from the standard input, which is usually provided via a pipeline or redirection. The input text is passed through the 'clean_text' function, and the cleaned text is printed to the standard output.

To eliminate contamination from other languages, we performed two iterations. In the first iteration, we removed all paragraphs containing non-Latin characters, which we identified based on their Unicode values. In addition, we removed all paragraphs containing fewer than ten words, which is particularly relevant for Wikipedia. Wikipedia articles often include not just text but also various links and other irrelevant content that we wanted to eliminate. The ten-word filter effectively removed such unnecessary paragraphs. In the second iteration, we used CLD2, a tool commonly used for language detection in the Common Crawl dataset.

We removed all paragraphs that CLD2 did not identify as at least 90 percent belonging to the target language in order to ensure the dataset remained monolingual. The exception was Northern Sami, as CLD2 does not support this language. For Northern Sami, we retained all paragraphs without applying language detection.

Within Hungarian language content, there exists both modern Hungarian and historical Hungarian (Old Hungarian) texts. In our research, we only include modern Hungarian content, as Wikipedia dumps are categorized using ISO 639-1 language codes, which only includes the code 'hu' for Hungarian. While Old Hungarian exists as a distinct language variety in the more comprehensive ISO 639-3 standard (with code 'ohu'), the current Wikipedia's infrastructure does not distinguish between historical and modern variants of Hungarian.

The significant reduction in the number of paragraphs—almost 35 percent—is due to the fact that a Wikipedia paragraph is far from being clean text. It contains a large amount of metadata, and many paragraphs include links to other languages, fragments of other languages, and quotes in different languages. We removed all of this to make our datasets as monolingual as possible. Comprehensive statistics for the dataset are provided in Table 3.

2.3 Training and Characterization of Tokenizer

Typically, a tokenizer training dataset uses only a portion of the available data. However, we decided to use the entire monolingual Wikipedia dataset we created to minimize any bias from dataset sampling on the tokenizer. By using all available data, we aim to produce a tokenizer that is as close as possible to an ideal tokenizer for the given language.

For training the tokenizer, we used a custom program based on the Tokenizers library from HuggingFace. The program is available on our GitHub. The only change we made to the reference code was setting the vocabulary size to 256,000, as we aimed to capture the maximum number of tokens. It's clear that such a large tokenizer is not practical for use, and the actual tokenizer will be a subset of these 256,000 tokens.

Interestingly, for Northern Sami, the dataset was insufficient to train a tokenizer with 256,000 tokens. We were only able to generate 93,187 tokens for this language. It is important to emphasize that even with some amount of text available, a minimum dataset size is required to effectively train a tokenizer. Nevertheless, the tokenizer for Northern Sami remains a valuable outcome.

2.4 Comparative Tokenology of Uralic Languages

With the trained tokenizers in hand, we asked how many common tokens exist between them. Since we created a monolingual dataset for training and ensured that the tokenizers were as monolingual as possible, we were curious to see how many tokens are shared among the Uralic languages, as well as how many tokens are specific to each language. When we used our tokenizer and examined the total number of unique tokens across all tokenizers, we found that there were 785,115 unique tokens. Not surprisingly, the majority of tokens are unique to a single tokenizer. We were particularly interested in finding how many tokens are shared across tokenizers, but we found no fully common tokens. The number of tokens that appear in only one tokenizer is 692,081. Only 8,102 tokens are shared across all four languages. After excluding Northern Sami,

Language Code	Estonian et	Finnish fi	Hungarian hu	Northern Sami se
Content pages	248,159	581,930	548,859	7,892

Table 3: Language and number of content pages in Wikipedia.

Language Code	Estonian et	Finnish fi	Hungarian hu	Northern Sami se
Raw Paragraphs	1,342,090	3,141,975	3,932,082	18,652
HQ paragraphs	926,628 (69%)	2,081,601 (66%)	2,385,635 (61%)	6,720 (36%)
HQ Words	33,140,238	86,033,550	122,512,745	249,444
HQ Chars	265,243,383	765,211,790	967,144,007	1,938,899

Table 4: Language and number of paragraphs, words, and characters.

we examined the common tokens between Estonian, Finnish, and Hungarian, and found 25,876 shared "core" tokens. It's important to note that, since the data comes from Wikipedia, the number of shared tokens is slightly overestimated. This is due to the presence of similar names, toponyms, and terms. While we tried to exclude non-Latin scripts, some countries and terms appear in English on Wikipedia, leading to an inflated count. In reality, the number of truly shared tokens is likely lower.

2.5 Evaluation of Tokenizer Efficiency for Tokenizing Texts in Uralic Languages

Our next task was to assess how effectively our tokenizers, trained on individual languages, tokenize their corresponding datasets. We compared their performance with widely used open tokenizers, including Mistral, LLaMA-2, LLaMA-3, and Gemma 2. We aimed to determine whether our language-specific tokenizers outperform these general-purpose tokenizers, which inevitably include tokens from multiple languages. To evaluate tokenizer efficiency, we used the Compression Rate metric. This metric assesses how effectively the tokenizer performs during training. The smaller the resulting text after tokenization, the better the tokenizer's performance. Overall, this metric can be interpreted as 'how many tokens are needed to represent a single word.' The closer this value is to one, the more efficient the tokenizer. The results of the analysis are presented in Table 5.

The figure 1 shows tokenization of parallel text samples in English and Finnish using different tokenizers. The Finnish text is processed by GPT4o (149 tokens, compression ratio 2.76), fi BPE (77 tokens, compression ratio 1.43), and uralic BPE (86 tokens, compression ratio 1.59). The English tokenization ratio (1.96) can be considered as a reference point for expected tokenizer performance, showing that current general-purpose tokenizers handle Finnish significantly worse than English (2.76 vs 1.96). However, specialized Finnish and Uralic tokenizers achieve even better compression ratios than English, demonstrating that proper language-specific tokenization can be highly efficient.

	urBPE	llama2	llama3	Gemma2
et	1.13	3.09	2.88	2.45
fi	1.18	3.41	3.12	2.49
hu	1.11	2.69	2.83	2.18
se	1.32	3.53	3.27	3.04
uralic	1.24	3.00	2.94	2.33

Table 5: Comparison of tokenizer efficiency between our tokenizer and three popular open-source tokenizers for four individual languages and the combined Uralic dataset. The metric used is the Compression Ratio, which is the ratio of the number of tokens after tokenization to the number of words in the dataset. The lower the ratio, the better the performance.

This visualization effectively demonstrates the efficiency gains achieved by language-specific tokenizers over general-purpose ones, with fi BPE showing nearly 50% reduction in token count compared to GPT4o for the same content. This reduction in token count directly translates to a proportional decrease in processing costs - effectively halving the cost of working with Finnish text. While our current work focuses on building separate specialized tokenizers, future research might explore whether similar efficiency gains could be achieved by selectively adding essential language-specific tokens to existing tokenizers, potentially offering a more practical path to improving multilingual performance while maintaining compatibility with existing models.

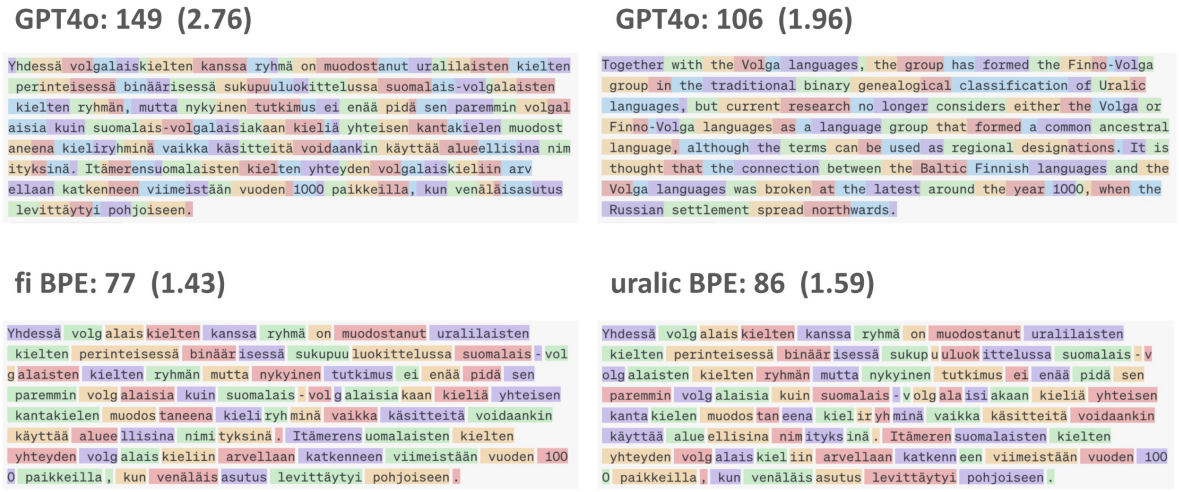

Figure 1: Visual comparison of tokenization patterns across different tokenizers. Each tokenizer's output is color-coded to show individual token boundaries, illustrating how the same text is segmented differently. Parallel text is shown in English (top-right panel, GPT4o) and Finnish (other panels) using different tokenizers.

3 Discussion

3.1 The Importance of Creating Well-Designed Multilingual Tokenizers

As shown in Table 5, if a tokenizer compresses texts efficiently, this not only holds fundamental importance but also has practical implications for the use of large language models, whose costs increase with each new iteration and generation. Perhaps we should rethink how we approach training tokenizers—not just by processing large amounts of text, but by assembling tokenizers from more specialized ones. This would ensure that tokens and languages are represented more equally in each tokenizer, allowing them to tokenize a wider variety of texts rather than primarily texts in the most popular languages, such as English, Russian, Chinese, German, and Japanese.

3.2 Significant Challenges with Low Digital Vitality Languages

For languages with low representation in textual data, none of the available tools for language analysis work reliably (Zaugg et al., 2022). We encountered difficulties even with relatively larger languages, such as Estonian. This issue is critical not only for endangered languages but also for any underrepresented languages in online texts (Hjortnaes et al., 2021). The most accurate count of existing Uralic languages is around 39-40 living and extinct languages (including recently extinct ones like Kamassian). Currently, 14 Uralic languages have their own Wikipedia editions, and only four have automatic Wikipedia dumps for computer-friendly processing.

The complete absence of Wikipedia editions for the most Uralic languages actually reinforces our paper's main point about the severe digital divide within the Uralic family. While other text sources might exist, using them would require entirely different methods for data cleaning and quality control.

The issue is not just the availability of text; it also lies in the tools themselves, which are trained on specific datasets. The performance of these tools varies greatly across different languages. This needs to be considered, and we believe that providing a high-quality monolingual dataset is a crucial resource for both evaluating existing tools and, in cases where they perform poorly, improving them or developing new tools tailored specifically to languages for which standard tools are ineffective.

In addition, we observed that language detection proved to be quite challenging, especially for Estonian. Estonian Wikipedia articles often contain more English words than other languages, which led to many false negatives and false positives in the detection process.

One potential approach for low-resource languages could be the artificial transfer of tokenizers from one language to another. This might be a solution for such languages, as word fragments can still be derived from relatively small amounts of text,

as we have demonstrated here with Northern Sami. Despite the limited amount of text, we were able to extract the basic tokens quite effectively; however, the challenge lies with full words. Perhaps enriching tokenizers with additional dictionaries could be a viable solution for languages that lack sufficient textual resources.

3.3 Monolingual BPE tokenizer as promising new tool for comparative linguistic

One of the surprising properties of BPE is its ability to find all repeating substrings in a language, making it a potentially valuable tool for comparative linguistics, especially when working with large text corpora. It can serve as an additional resource alongside existing linguistic tools (Hämäläinen, 2019; Silfverberg and Tyers, 2019).

This tool can be used, for example, for language detection, potentially more effectively than probability-based models. In essence, BPE is also a type of probabilistic model, but it has the advantage of identifying tokens that are more characteristic of a language, even if they are relatively short. By analyzing these short tokens, we can estimate the likelihood that a given text belongs to a specific language. Currently, no such tools exist Language Modeling and Perplexity Reduction

3.4 Texts Authorship Quality in Low-Resource Uralic Languages

For low-resource languages, particularly within the Uralic language family, large-scale datasets remain scarce. A critical consideration when working with extremely low-resource languages is the assessment of dataset quality and representativeness. This is particularly relevant for languages with a small number of native speakers yet maintain some presence in digital corpora. A fundamental methodological challenge arises regarding the linguistic authenticity of the collected data, specifically concerning the authorship of corpus texts. The distinction between native speakers and proficient L2 (second language) users becomes crucial, as the latter may possess sufficient competency to produce texts while potentially introducing subtle non-native patterns. In low-resource scenarios, where the corpus size is inherently limited, such authorship variations can significantly impact the linguistic quality of the dataset, subsequently affecting downstream tasks such as tokenization and model training.

Future research will explore methodologies for assessing dataset authenticity through perplexity-based analysis. We hypothesize that texts authored by native and non-native speakers will demonstrate measurably different perplexity patterns, though the exact nature of these differences requires empirical investigation. We plan to validate this hypothesis by developing a systematic approach to perplexity-based authorship analysis, which could provide a quantitative tool for dataset quality assessment in low-resource scenarios, particularly valuable for Uralic languages where high-quality digital data is crucial yet scarce.

4 Conclusion

In this work, we addressed the inequality in language representation within large language models, focusing on the Uralic language family. Using high-quality monolingual datasets from Wikipedia for Estonian, Finnish, Hungarian, and Northern Sami, we trained specialized BPE tokenizers. Our analysis showed that existing tokenizers poorly represent Uralic languages, leading to inefficient tokenization and increased computational costs. Our monolingual tokenizers outperformed widely used open-source tokenizers with lower compression ratios, improving both performance and cost-efficiency for underrepresented languages. Additionally, these tokenizers offer valuable tools for comparative linguistic analysis, highlighting shared and unique features of Uralic languages. By sharing our tokenizers and datasets, we provide key resources for further research in natural language processing for low-resource languages. This work underscores the need for equitable tokenization, especially for languages with low digital vitality.

5 Availability

All resources from this research are publicly available:

- Source Code: Complete analysis tools and scripts are available in our GitHub repository https://github.com/nup-csai/uralicBPE
- Reproducibility: A Jupyter notebook containing step-by-step analysis reproduction, including all tables and figures
- Data: Processed Wikipedia datasets and pretrained tokenizers are hosted on Hugging Face https://huggingface.co/datasets/nup-csai/uralicBPE

References

Judit Acs, Katalin Pajkossy, and András Kornai. 2017. Digital vitality of Uralic languages. *Acta Linguistica Academica*, 64(3):327–345.

Khalid Alnajjar, Mika Hämäläinen, and Jack Rueter. 2023. Sentiment analysis using aligned word embeddings for Uralic languages. In *Proceedings of the Second Workshop on Resources and Representations for Under-Resourced Languages and Domains (RESOURCEFUL-2023)*, pages 19–24, Tórshavn, the Faroe Islands. Association for Computational Linguistics.

Timofey Arkhangelskiy. 2019. Corpora of social media in minority Uralic languages. In *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages*, pages 125–140, Tartu, Estonia. Association for Computational Linguistics.

Iaroslav Chelombitko, Egor Safronov, and Aleksey Komissarov. 2024. Qtok: A Comprehensive Framework for Evaluating Multilingual Tokenizer Quality in Large Language Models. *arXiv preprint arXiv:2410.12989*.

Dami Choi, Derrick Xin, Hamid Dadkhahi, Justin Gilmer, Ankush Garg, Orhan Firat, Chih-Kuan Yeh, Andrew M. Dai, and Behrooz Ghorbani. 2023. Order Matters in the Presence of Dataset Imbalance for Multilingual Learning. *arXiv preprint arXiv:2312.06134*.

C. M. Downey, Terra Blevins, Dhwani Serai, Dwija Parikh, and Shane Steinert-Threlkeld. 2024. Targeted multilingual adaptation for low-resource language families. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15647–15663, Miami, Florida, USA. Association for Computational Linguistics.

Nils Hjortnaes, Niko Partanen, and Francis M. Tyers. 2021. Keyword spotting for audiovisual archival search in Uralic languages. In *Proceedings of the Seventh International Workshop on Computational Linguistics of Uralic Languages*, pages 1–7, Syktyvkar, Russia (Online). Association for Computational Linguistics.

Mika Hämäläinen. 2019. UralicNLP: An NLP Library for Uralic Languages. *Journal of Open Source Software*, 4(37):1345.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea. Association for Computational Linguistics.

Jeroen Ooms. 2024. *cld3: Google's Compact Language Detector 3*.

Ravi Paul, Himanshu Buckchash, Shantipriya Parida, and Dilip K. Prasad. 2024. Towards a More Inclusive AI: Progress and Perspectives in Large Language Model Training for the Sámi Language. *arXiv preprint arXiv:2405.05777*.

Alexander Petrov, Enrico La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language Model Tokenizers Introduce Unfairness Between Languages. *arXiv preprint arXiv:2305.15425*.

Miikka Silfverberg and Francis Tyers. 2019. Data-driven morphological analysis for Uralic languages. In *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages*, pages 1–14, Tartu, Estonia. Association for Computational Linguistics.

Cansu Toraman, Emine Hande Yilmaz, Fırat Şahinuç, and Osman Ozcelik. 2023. Impact of Tokenization on Language Models: An Analysis for Turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21.

Isabelle A. Zaugg, Anushah Hossain, and Brendan Molloy. 2022. Digitally-disadvantaged languages. *Internet Policy Review*, 11(2).