Deep Language Geometry: Constructing a Metric Space from LLM Weights

Maksym Shamrai
Institute of Mathematics of NASU
MacPaw
Kyiv, Ukraine
mshamrai@macpaw.com

Vladyslav Hamolia
MacPaw
Kyiv, Ukraine
vhamolya@macpaw.com

Abstract

We introduce a novel framework that utilizes the internal weight activations of modern Large Language Models (LLMs) to construct a metric space of languages. Unlike traditional approaches based on hand-crafted linguistic features, our method automatically derives high-dimensional vector representations by computing weight importance scores via an adapted pruning algorithm. Our approach captures intrinsic language characteristics that reflect linguistic phenomena. We validate our approach across diverse datasets and multilingual LLMs, covering 106 languages. The results align well with established linguistic families while also revealing unexpected inter-language connections that may indicate historical contact or language evolution. The source code, computed language latent vectors, and visualization tool are made publicly available at https://github.com/mshamrai/deep-language-geometry.

1 Introduction

Languages are complex systems with rich internal structures and dynamic evolution. Traditional linguistic classifications based on typological features, historical migration patterns, or lexical similarity have long served to group languages into families such as Indo-European, Uralic, and Turkic. However, these approaches typically capture only historical or static aspects of language similarity, potentially overlooking modern linguistic influences driven by technology and globalization. In an era where language use and structure are continuously reshaped, it is timely to develop methods that automatically capture both historical and current linguistic characteristics.

Recent advances in Natural Language Processing (NLP) have been largely driven by Large Language Models (LLMs), which have demonstrated remarkable capabilities in language modeling and a wide range of linguistic tasks (Devlin et al., 2018; Radford et al., 2019). These models, trained on vast multilingual corpora, learn representations that implicitly encode a wide variety of lexical, syntactic, and even phonological properties (Conneau et al., 2019).

Building on prior work (Shamrai, 2024), which empirically shows that the internal activations of LLM weights vary with the language of the input data, we hypothesize that the internal weights of LLMs encode valuable information about inter-language similarity and can serve as a foundation for quantifying relationships between languages.

Therefore, in this work, we propose a novel approach for constructing a metric space of languages by leveraging the weights of modern LLMs. Our method extracts high-dimensional vector representations from LLM weights activations, where the distance between any two vectors reflects the similarity between the underlying linguistic structures. Activations encode patterns of co-occurrence and contextual relationships specific to each language's grammar and lexical properties.

We construct a metric space (X, d_h), where X is the set of high-dimensional language vectors and d_h is the Hamming distance between them. We then design a distance-preserving mapping that projects these high-dimensional vectors into a low-dimensional space (Y, d_e), where distances are induced from the Euclidean (L2) norm. This transformation provides deeper insight into the latent structures encoded by LLMs.

Furthermore, we calculate this latent representation for 106 languages. This revealed the opportunity to visualize, cluster and analyze the relationships between the languages.

Our code, computed language latent vectors, and analysis tool are made publicly available, designed to assist researchers and practitioners in linguistic analysis and offering valuable resource for further linguistic investigation.

The contributions of this work are as follows:

- We introduce a novel approach that constructs a metric space of languages using LLM weights and apply it to 106 languages, enabling automatic and data-driven measurement of linguistic distances.
- We demonstrate that the derived metric space supports meaningful clustering of languages, reflecting both historical relationships and modern linguistic features.
- We fully open-source our work along with a tool for preliminary analysis.

While not claiming linguistic expertise, this study introduces a novel toolset intended to support linguistic research. It offers a fresh view of language similarity by exploiting the latent knowledge embedded in LLMs.

2 Related Work

The quantification of language similarity has a rich history, beginning with early lexical approaches. Pioneering work (Swadesh, 1952) established methods for comparing languages using shared cognates, a practice later refined by Holman et al. (2011), which employs normalized Levenshtein distances over fixed word lists. Although these lexical methods have been successfully used to construct language family trees, they are handcrafted and require manual effort to select and curate appropriate word lists and features.

Also, resources such as the World Atlas of Language Structures (Dryer and Haspelmath, 2005) offer comprehensive typological data that allow languages to be represented as feature vectors. Distance measures computed over these vectors have been shown to reveal groupings consistent with established genetic relationships (O'Horan et al., 2016; De Gregorio et al., 2024). However, these methods are limited by the quality and coverage of available databases, their reliance on expert-curated features, and their inability to fully capture language-specific variations or recent evolutionary trends.

Phonological properties offer another valuable dimension for language comparison. Studies utilizing phoneme inventory data from resources like PHOIBLE (Moran et al., 2014) demonstrate that phonological distances – often measured by overlap indices such as the Jaccard similarity – can capture both genetic relationships and areal phenomena. But phonological methods need reliable phoneme lists, are affected by how sounds are written, and often miss language structure beyond sounds.

Recent deep-learning work has popularised embedding-based measures of language distance. Multilingual encoders such as mBERT (Devlin et al., 2018), XLM-R (Conneau et al., 2020) and LASER (Artetxe and Schwenk, 2019; Heffernan et al., 2022) produce contextual token embeddings that implicitly encode lexical, syntactic and semantic features. LASER is trained to output a single sentence vector directly, whereas mBERT and XLM-R require a pooling step (e.g., mean pooling or the `[CLS]` token) to obtain a sentence-level embedding. When sentence embeddings are averaged over large, balanced corpora, the resulting language-level representations have proved useful for quantifying cross-lingual similarity (Rama et al., 2020). However, because the underlying encoders operate at the token – and therefore sentence – level, their effectiveness still depends on corpus size and domain balance.

Overall, the literature on language distance metrics has evolved from classical lexicostatistical methods and handcrafted feature extraction to sophisticated neural representations. Each approach offers valuable insights into the relationships between languages, but they often suffer from labor-intensive preprocessing, limited database coverage, or sensitivity to input variations. This motivates our approach: rather than relying on manually curated features or sentence-based embeddings, we propose an automatic, data-driven method that leverages the internal weights of modern LLMs to construct a metric space of languages.

Moreover, to best of our knowledge, no study has attempted to derive a language metric space from decoder-only LLMs. The method introduced here is therefore the first to use weight-level signals in causal transformers for measuring cross-language similarity.

3 Methodology

The main hypothesis in this work is that Large Language Models are a good choice to measure internal language structure since they are trained to model languages. Formally, this is typically framed as maximizing the log-likelihood of the observed sequence of tokens. Let $x_1, x_2, \ldots, x_T$ represent a sequence of tokens, where $x_t \in \mathcal{V}$ and $\mathcal{V}$ is the vocabulary. The objective is to maximize the likelihood of the sequence under the model's

parameters θ:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \log p(x_t | x_1, x_2, \ldots, x_{t-1}; \theta),$$

where $p(x_t | x_1, x_2, \ldots, x_{t-1}; \theta)$ is the conditional probability of the token x_t given the previous tokens, modeled by a neural network or another probabilistic model.

3.1 Weight Importance Metric

We begin by revisiting classical pruning approaches such as Optimal Brain Damage (LeCun et al., 1989), which motivate the rationale behind our approach.

The typical pruning objective is to minimize the error introduced by approximating the original weight matrix. Consider the following objective function:

$$E = \|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_2^2 \to \min, \tag{1}$$

where $\mathbf{W}$ is the original weight matrix of a layer, $\hat{\mathbf{W}}$ is the pruned (sparse) weight matrix, and $\mathbf{X}$ is the input to that layer.

The variation of the error E for a weight row $\mathbf{w}$ can be expressed as:

$$\delta E = \left(\frac{\partial E}{\partial \mathbf{w}}\right)^T \delta\mathbf{w} + \frac{1}{2}\,\delta\mathbf{w}^T \mathbf{H}\,\delta\mathbf{w} + \mathcal{O}(\|\delta\mathbf{w}\|^3),$$

where $\mathbf{H} \equiv \frac{\partial^2 E}{\partial \mathbf{w}^2}$ is the Hessian matrix.

At a local minimum of the training error, we have

$$\frac{\partial E}{\partial \mathbf{w}} \approx 0,$$

and higher order terms are neglected.

Our goal is to set one of the weights, say w_q, to zero while minimizing the increase in error. This introduces the constraint:

$$\mathbf{e}_q^T \delta\mathbf{w} + w_q = 0,$$

where $\mathbf{e}_q$ is the qth standard basis vector. Thus, the optimization problem in Equation (1) can be reformulated as:

$$\begin{aligned} \min_{\delta\mathbf{w}} \quad & \frac{1}{2}\,\delta\mathbf{w}^T \mathbf{H}\,\delta\mathbf{w}, \\ \text{s.t.} \quad & \mathbf{e}_q^T \delta\mathbf{w} + w_q = 0. \end{aligned} \tag{2}$$

This constrained problem can be solved using Lagrange multipliers. For the detailed derivation see Appendix A.

The resulting increase in error is given by:

$$E_q = \frac{1}{2} \cdot \frac{w_q^2}{\mathbf{e}_q^\top \mathbf{H}^{-1} \mathbf{e}_q}. \tag{3}$$

By computing E_q for every weight w_q, one can prune the weight that causes the smallest increase in error, thereby minimally affecting the layer's output. Intuitively, this means we identify which weights are most critical for the model's performance on a specific language. Weights with high importance scores are those whose removal would substantially degrade the model's ability to predict tokens in that language.

SparseGPT (Frantar and Alistarh, 2023) adopts this idea within an LLM pruning algorithm. They compute the importance metric $\mathbf{S}_{ij}$ for a layer as follows (Sun et al., 2023):

$$\mathbf{S}_{ij} = \left[\frac{|\mathbf{W}|^2}{\operatorname{diag}\left((\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\right)}\right]_{ij}. \tag{4}$$

As in SparseGPT, we build $\mathbf{X}$ *per linear sub-layer* by stacking the pre-activation hidden states of a small calibration set into an $N \times d_{\text{in}}$ matrix. For a weight matrix $\mathbf{W}$ the local Hessian is $\mathbf{H} = \mathbf{X}^\top\mathbf{X}$, and we invert $\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)$ *once per layer*. Thus, Equation (4) is simply a matrix-valued, regularised version of the scalar error-increase criterion in Equation (3).

Shamrai (2024) suggests that the SparseGPT algorithm provides statistically stable results for different LLMs and subsets of a data in language-specific setting. Therefore, in our work, we adopt the algorithm to compute weight importance vectors.

3.2 Rationale Behind the Approach

By definition, $\mathbf{S}_{ij}$ quantifies the importance of weight $\mathbf{W}_{ij}$ for a given input. In our approach, we estimate the importance of the weights for a specific language by using datasets in that language. Consequently, $\mathbf{S}_{ij}$ reflects the contribution of each weight to language modeling.

Assuming that the network is well-trained on language modeling, higher $\mathbf{S}$ scores indicate greater contribution. If two languages yield similar patterns of important weights, it suggests that they are similar in terms of language modeling characteristics.

3.3 Constructing a Metric Space

To derive a vector representation from the importance metric, we treat the importance scores as coordinates in a high-dimensional space. Specifically, we define the vector

$$\mathbf{v} = \big(\mathbf{S}^0_{00}, \mathbf{S}^0_{01}, \ldots, \mathbf{S}^k_{ij}, \ldots, \mathbf{S}^l_{nm}\big) \in \mathbb{R}^N,$$

where the set $\{\mathbf{W}^k\}_{k=0}^l$ consists of weight matrices $\mathbf{W}^k \in \mathbb{R}^{n_k \times m_k}$ for each layer k, and N is the total number of parameters in the chosen LLM. In other words, the vector $\mathbf{v}$ is obtained by flattening and concatenating all the importance matrices $\mathbf{S}^k$ corresponding to each layer.

There are two challenges with using the raw importance matrix $\mathbf{S}$ to form this vector representation:

1. The importance scores are not normalized across layers, meaning that they are only meaningful within the context of a single layer.

2. The resulting vector is high-dimensional, with each dimension represented by a floating-point number (typically 16 bits), leading to large memory requirements.

To mitigate this, we propose a thresholding approach analogous to binary quantization. Specifically, we assign a value of 1 only to the most important weights by thresholding $\mathbf{S}_{ij}$ at its median:

$$\hat{\mathbf{S}}_{ij} = \mathbb{1}\big(\mathbf{S}_{ij} > \text{median}(\mathbf{S})\big).$$

This binary representation requires only 1 bit per value, reducing the storage requirement substantially compared to 16-bit floating-point representations.

Let X denote the set of language vectors (one per language) of length N. We then define a metric space on X using the Hamming distance (i.e., the XOR operation) as the metric.

For $\mathbf{x}, \mathbf{y} \in X$ the Hamming distance is

$$d_h(\mathbf{x}, \mathbf{y}) \;=\; \sum_{i=1}^{N} \mathbb{1}\big[x_i \neq y_i\big],$$

where $\mathbb{1}[\cdot]$ is the indicator function.

The function d_h is non-negative, symmetric, equals 0 iff $\mathbf{x} = \mathbf{y}$, and satisfies the triangle inequality, therefore, (X, d_h) is a metric space.

Algorithm 1 Torgerson Scaling (Classical MDS)

Require: Distance matrix $D \in \mathbb{R}^{n \times n}$, $n = |X|$
Ensure: Coordinates $Y \in \mathbb{R}^{n \times d}$ representing points in d dimensions

$J \leftarrow I_n - \frac{1}{n}\mathbf{1}_n$ ▷ Compute centering matrix
$D^2 \leftarrow D \odot D$ ▷ Element-wise square of D
$B \leftarrow -\frac{1}{2}\, J\, D^2\, J$ ▷ Compute Gram matrix
$(\lambda, V) \leftarrow \texttt{eigh}(B)$ ▷ Compute the eigen-decomposition of B
$(\lambda, V) \leftarrow \texttt{sort}((\lambda, V))$ ▷ Sort eigenvalues in descending order and reorder eigenvectors accordingly
$d \leftarrow \#\{\lambda_i \mid \lambda_i > \epsilon\}$ ▷ Select dimensions with significant eigenvalues ($\epsilon \approx 10^{-10}$)
$L \leftarrow \text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_d})$
$V_d \leftarrow [v_1, v_2, \ldots, v_d]$
return $Y \leftarrow V_d\, L$

3.4 Isometry via Dimensionality Reduction

Even after quantization, the binary vectors remain high-dimensional due to the large number of model parameters, making distance computations and other latent space applications computationally expensive. To address this, we construct an isometry – a transformation that preserves distances between points when mapping from one metric space to another.

In our experiments, we employ different LLMs and multiple datasets. We compute the language-by-language distance matrix for each model and dataset, and then average them to obtain a robust distance measure:

$$\begin{aligned}
&\mathbf{D}_{lk} \in \mathbb{R}^{|X| \times |X|},\\
&\mathbf{D}_{lk} = \{d_h(\mathbf{v}_i, \mathbf{v}_j) : \mathbf{v}_i, \mathbf{v}_j \in X\},\\
&\hat{\mathbf{D}} = \mathbb{E}_{l \sim p_{\text{LLM}}} \mathbb{E}_{k \sim p_{\text{data}}}[\mathbf{D}_{lk}]\\
&\quad \approx \frac{1}{nm} \sum_{l=0}^{n} \sum_{k=0}^{m} \mathbf{D}_{lk},
\end{aligned}$$

where $\mathbf{D}_{lk}$ is the distance matrix computed for the lth LLM and the kth dataset, n is the number of LLMs, m is the number of datasets, and $|X|$ is the number of languages.

This averaging process reduces noise and ensures that the final distances are not overly dependent on any particular dataset or model.

Dataset	# Languages in Dataset	# Languages Used in Work
Wikipedia	323	106
CulturaX	167	102
fineweb-2	2051	103

Table 1: Comparison of datasets: Wikipedia, CulturaX, and fineweb-2. The table reports the total number of languages in each dataset and the number of languages used in this work.

We then construct an isometry

$$f : X \to Y,$$

where Y is a metric space endowed with the Euclidean metric $d_e(x, y) = \|x - y\|_2$.

To build f, we apply Torgerson scaling (classical multidimensional scaling) (Borg and Groenen, 2007). The result is a set of points $Y \in \mathbb{R}^{|X| \times d}$, where d is the minimum number of dimensions required to preserve the distances in $\hat{\mathbf{D}}$ (see Algorithm 1). Notably, d is much smaller than the original dimensionality N of the language vectors and satisfies $d \leq |X|$.

Therefore, our method leverages LLMs weights to construct a language vector representation and embed it in a metric space which could be used for analysis of languages similarities.

4 Results

To analyze the metric space of languages, we vary clustering algorithms along with dimensionality reduction ones. In particular, for clustering HDBSCAN (Campello et al., 2013), k-means (Lloyd, 1982), and predefined linguistic families with its subfamilies are used to highlight the correspondence between the derived metric space and established linguistic classifications. Throughout this paper we adhere to the language classification provided in Hammarström et al. (2024).

For two-dimensional visualizations, we reduce the dimensionality of the language vectors using t-SNE (van der Maaten and Hinton, 2008), UMAP (McInnes et al., 2018), and minimum spanning trees (Pettie and Ramachandran, 2002). Although all methods yield valuable insights, we include in the main text only the minimum spanning trees (MST) visualizations colored by language families and subfamilies, as they most clearly represent the inter-language relationships. Additional figures are provided in Appendix D and also available via our open-source tool[1].

4.1 Datasets and Models

In our experiments, we employ three LLMs and three datasets. The models used are Mistral 7B (Jiang et al., 2023), Gemma 3 4B (Team et al., 2025), and Llama 3.2 1B (Grattafiori et al., 2024). All models are multilingual and have been trained on more than 100 languages. Notably, although Llama officially supports only 8 languages, our results indicate that it still produces useful representations for our purposes. As datasets, we selected those with a high number of languages: Wikipedia (Foundation), CulturaX (Nguyen et al., 2024), and fineweb-2 (Penedo et al., 2024).

We start with a target inventory of 106 languages and attempted to apply the same list across all corpora. Wikipedia contains material for every language in this set, but CulturaX omits Chinese (Traditional), Min Nan Chinese, Scots, and Crimean Tatar, whereas fineweb-2 lacks Chinese (Traditional), English[2], Serbo-Croatian, and Tagalog. Table 1 lists the total number of languages present in each dataset alongside the subset that could be retained from our 106-language list. For the full list of languages see Appendix B.

To compute the language vectors, we proceed as follows:

1. **Calibration data.** For every language in each corpus (Wikipedia, CulturaX, fineweb) we sample $2^{19} = 524{,}288$ tokens.

2. **Weight-importance vectors.** For each language–corpus pair and for each LLM (Mistral 7B, Gemma 3 4B, Llama 3.2 1B) we compute a binary weight importance vector whose length matches the model's parameter count, yielding $3(106 + 102 + 103) = 933$ vectors.

[1] https://huggingface.co/spaces/mshamrai/language-metric-analysis

[2] For the English subset, we use the *fineweb* dataset (https://huggingface.co/datasets/HuggingFaceFW/fineweb)

3. **Distance matrices.** Hamming distances between language vectors produce nine language–by–language matrices (one per model–corpus combination).

4. **Aggregation.** These nine matrices are averaged element-wise over the observed entries to form a single average distance matrix.

5. **Embedding.** Classical MDS on the average matrix embeds the languages space in $\mathbb{R}^{104}$, where Euclidean distance defines the final language metric.

4.2 Evaluation of k-means Clustering Against Two Linguistic Categorization

After we embed the $|X| = 106$ language vectors into $\mathbb{R}^{104}$ via classical MDS we evaluate the language embeddings using k-means. The resulting partition is compared with two reference label sets: (i) *high-level families* (18 macro-families) and (ii) *primary branches* (35 sub-families). The number of clusters in k-means is equal to the number of labels in the reference sets.

We compute the following metrics:

- **Silhouette score** (Rousseeuw, 1987): the mean difference between a point's average distance to its own cluster and to the nearest neighboring cluster. Values range from -1 (poor separation) to $+1$ (well-separated, compact clusters).

- **Adjusted Rand Index (ARI)** (Hubert and Arabie, 1985): agreement between two partitions, corrected for chance. 1 indicates perfect alignment, 0 indicates random overlap.

- **Cluster purity** (Schütze et al., 2008): the fraction of data points that share the majority label within their cluster. Values in $[0, 1]$.

Reference	Sil.	ARI	Purity
Macro families	0.047	0.116	0.755
Primary branches	0.056	0.434	0.811

Table 2: Clustering metrics for the k-means solution against two standard language classification. "Sil" is the internal silhouette score.

Table 2 shows that switching from broad families to primary branches raises the ARI from 0.116 to 0.434 and the purity from 0.755 to 0.811. Therefore, the metric space captures finer-grained language groups and can estimate similarity at a micro level. However, the internal silhouette remains low (about 0.05), meaning many languages lie almost as close to other clusters as to their own.

4.3 Language Trees

A minimum spanning tree (MST) connects all data points in the dataset with the smallest possible total edge weight, where the edge weight corresponds to the distance between language vectors. We employ the Kamada-Kawai layout, a force-directed algorithm where edge lengths are proportional to the distances (Kamada and Kawai, 1989). This layout effectively visualizes the structure and connectivity within the MST, revealing not only the clusters of closely related languages but also links between different language families.

Figure 1 shows the MST for all languages used in our work. The visualization highlights well-established clusters corresponding to known language families as well as some unexpected connections. For example, Tajik (an Indo-European language) appears linked to a cluster of Turkic languages, which can likely be explained by geographical proximity. Similarly, the branch containing Latvian and Lithuanian is connected to a cluster of Uralic languages, possibly due to regional contact with Finnish and Estonian. A less obvious connection is observed between Turkish and Hungarian, which might be attributed to historical interactions. Additionally, Vietnamese is found to be close to Chinese, despite Vietnamese using the Latin alphabet and Chinese employing logographic characters, indicating that our method captures internal language characteristics beyond mere orthographic features.

Figure 2 focuses on Indo-European and Turkic languages, with coloring based on their primary branches. This figure clearly illustrates that Crimean Tatar, although belonging to the Kipchak branch, is closely connected to Turkish, an Oghuz language. The MST also links English, a Germanic language, directly to Spanish, a Romance language, likely reflecting their close geographic and sociolinguistic contact in the Americas.

One intriguing observation is that Ukrainian does not exhibit a direct connection with Polish in the MST, which is unexpected. However, further analysis reveals that Polish consistently ranks among the top five closest languages to Ukrainian

Figure 1: Minimum spanning tree for all languages. Colors represent language families.

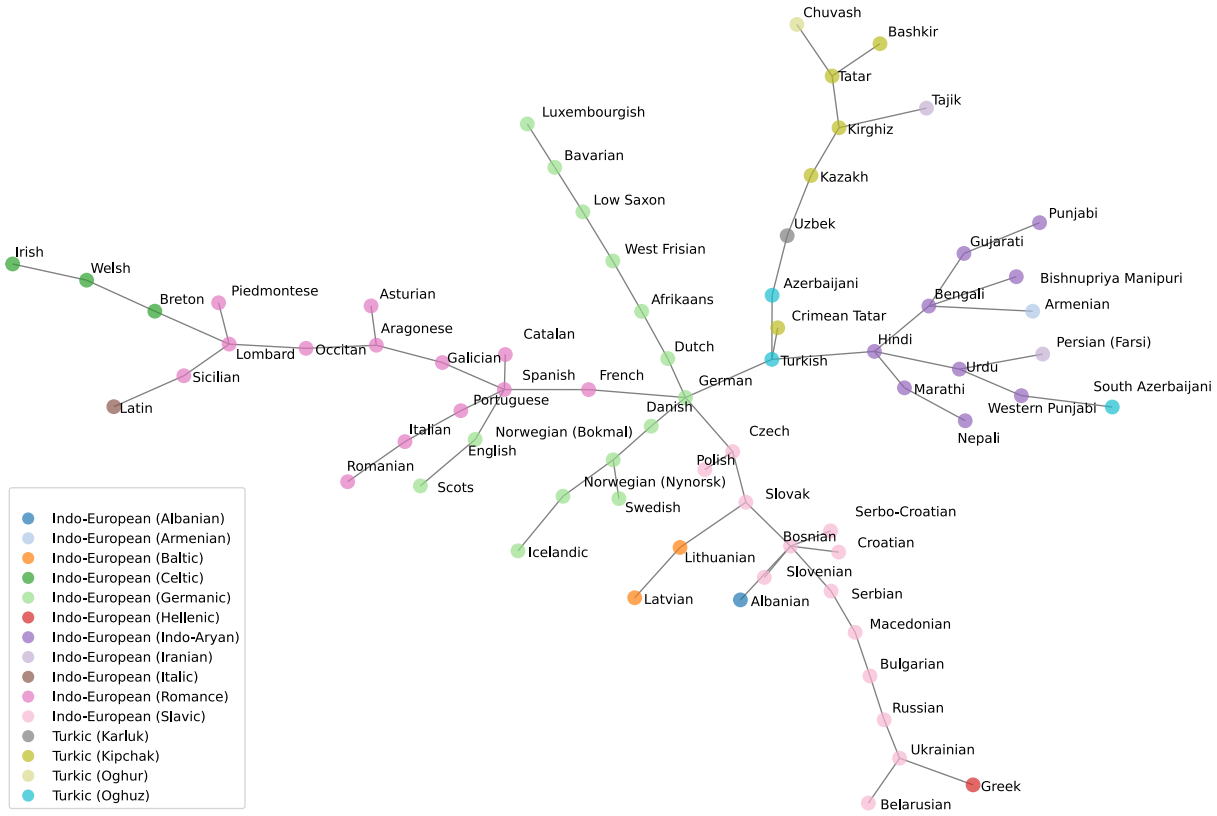

Figure 2: Minimum spanning tree for languages from the Indo-European and Turkic families. Colors represent language primary branches.

across all models and datasets, coming in third after averaging the distances.

In summary, the minimum spanning trees reveal logical relationships among languages and their families. In addition, the presence of uncommon connections suggests potential historical contacts or convergent evolution. We leave further investigation of areal influences or language borrowing phenomena to professionals.

5 Conclusion

In this work, we introduced a novel framework for constructing a metric space that quantifies language similarity by leveraging the internal weight activations of Large Language Models.

Our approach, based on computing binary vectors from weight importance metrics and reducing their dimensionality via isometric mappings, captures linguistic features, and the resulting metric space not only aligns with established linguistic families but also reveals intriguing inter-language connections.

Overall, this study lays the groundwork for a data-driven paradigm in language similarity analysis with significant implications for theoretical linguistics.

Limitations

While our approach offers a novel perspective on constructing a metric space for languages using LLM weight activations, several limitations remain:

1. **Computational Expense:** Computing the binary vectors is time-consuming. For example, on the Mistral 7B model, generating one binary vector requires approximately 20 minutes on an NVIDIA RTX 3090 GPU.

2. **Scalability to Larger Models:** We have not yet evaluated the method on LLMs with a significantly higher number of parameters due to resource constraints. It is possible that larger models might yield more accurate or robust representations.

3. **Remaining Bias from Source Models:** Averaging distances across three LLMs does not eliminate their shared weaknesses. In particular, the metric space can still reflect poor performance on low resource languages, which

may introduce inconsistencies with known language family relationships.

Additionally, we were unable to mathematically or empirically validate that the derived distance metric can serve as an effective guideline for fine-tuning and transfer learning of LLMs. Although the underlying hypothesis suggests that linguistic similarity may enhance the language modeling capabilities through transfer learning between related languages, our preliminary experiments – where we fine-tuned an LLM on similar languages using various configurations – did not yield statistically significant improvements. This indicates that a more sophisticated approach may be required, and we leave this investigation for future work. For more details see Appendix C.

Another promising direction for future research is to identify which specific weights or layers contribute most to the observed similarities and dissimilarities. It is likely that only a subset of layers significantly influences the metric. By pinpointing these layers, we may reduce computational complexity and accelerate the metric computation without compromising accuracy.

Acknowledgment

We express our gratitude to the Armed Forces of Ukraine for their protection, which has made this research possible.

References

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Ingwer Borg and Patrick JF Groenen. 2007. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2019. What unsupervised multilingual sentence representations learn about language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4598–4608. Association for Computational Linguistics.

Juan De Gregorio, Raúl Toral, and David Sánchez. 2024. Exploring language relations through syntactic distances and geographic proximity. *EPJ Data Science*, 13(1):61.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2005. *World Atlas of Language Structures*. Oxford University Press, Oxford.

Wikimedia Foundation. Wikimedia downloads.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2024. Glottolog database 5.1. `https://glottolog.org`.

Kevin Heffernan, Onur Çelebi, and Holger Schwenk. 2022. Bitext mining using distilled sentence representations for low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2101–2112, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Eric W. Holman, Søren Wichmann, and Christopher H. Brown. 2011. Automated dating of languages. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 2052–2058. European Language Resources Association.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2:193–218.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Tomihisa Kamada and Satoru Kawai. 1989. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Simon Moran, Daniel McCloy, and Sue Wright. 2014. Phonological similarity and its applications in cross-linguistic phonetics. *Journal of Phonetics*, 42:20–35.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2024. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4226–4237, Torino, Italia. ELRA and ICCL.

Kris O'Horan, Steven Galle, and Nathalie Schneider. 2016. Syntactic variation in multilingual representations: Investigating cross-lingual transfer. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Martin Jaggi, Leandro von Werra, and Thomas Wolf. 2024. Fineweb2: A sparkling update with 1000s of languages.

Seth Pettie and Vijaya Ramachandran. 2002. An optimal minimum spanning tree algorithm. *Journal of the ACM (JACM)*, 49(1):16–34.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. volume 1, page 9. OpenAI Blog, https://openai.com/blog/better-language-models/.

Taraka Rama, Lisa Beinborn, and Steffen Eger. 2020. Probing multilingual BERT for genetic and typological signals. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1214–1228, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Maksym Shamrai. 2024. Language-specific pruning for efficient reduction of large language models. In *Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP)@ LREC-COLING 2024*, pages 135–140.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Morris Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to north american indians and eskimos. *Proceedings of the American philosophical society*, 96(4):452–463.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

A Derivation of weight importance metric

$$\begin{aligned} \min_{\delta \mathbf{w}} \quad & \frac{1}{2}\delta \mathbf{w}^T \mathbf{H}\, \delta \mathbf{w}, \\ \text{s.t.} \quad & \mathbf{e}_q^T \delta \mathbf{w} + w_q = 0. \end{aligned}$$

The problem could be solved using Lagrange multiplier. We begin with the Lagrangian:

$$\mathcal{L} = \frac{1}{2}\delta \mathbf{w}^\top \mathbf{H} \delta \mathbf{w} + \lambda \left(\mathbf{e}_q^\top \delta \mathbf{w} + w_q \right).$$

Taking the derivative with respect to $\delta \mathbf{w}$ and setting it to zero:

$$\begin{aligned} \nabla_{\delta \mathbf{w}} \mathcal{L} &= \mathbf{H} \delta \mathbf{w} + \lambda \mathbf{e}_q = 0, \\ \Rightarrow \quad \delta \mathbf{w} &= -\mathbf{H}^{-1} \mathbf{e}_q \lambda. \end{aligned}$$

Substituting into the constraint:

$$\mathbf{e}_q^\top (-\mathbf{H}^{-1} \mathbf{e}_q \lambda) + w_q = 0,$$

we get:

$$\lambda = \frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1} \mathbf{e}_q}.$$

Thus, the change in weights:

$$\delta \mathbf{w} = -\mathbf{H}^{-1} \mathbf{e}_q \cdot \frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1} \mathbf{e}_q}.$$

Notice that:

$$\mathbf{H}\delta\mathbf{w} = \mathbf{H}\left(-\mathbf{H}^{-1}\mathbf{e}_q \frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q}\right)$$
$$= -\mathbf{e}_q \frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q},$$

and

$$\delta\mathbf{w}^\top = \left(-\mathbf{H}^{-1}\mathbf{e}_q \frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q}\right)^\top$$
$$= -\frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q}\mathbf{e}_q^\top \mathbf{H}^{-1}$$

Now compute the increase in error:

$$E_q = \frac{1}{2}\delta\mathbf{w}^\top \mathbf{H}\delta\mathbf{w}$$
$$= \frac{1}{2}\frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q}\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q \frac{w_q}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q}$$
$$= \frac{1}{2}\cdot\frac{w_q^2}{\mathbf{e}_q^\top \mathbf{H}^{-1}\mathbf{e}_q}$$

B Full list of languages used

Afrikaans, Albanian, Arabic, Aragonese, Armenian, Asturian, Azerbaijani, Bashkir, Basque, Bavarian, Belarusian, Bengali, Bishnupriya Manipuri, Bosnian, Breton, Bulgarian, Burmese, Catalan, Cebuano, Chechen, Chinese (Simplified), Chinese (Traditional), Chuvash, Crimean Tatar, Croatian, Czech, Danish, Dutch, Egyptian Arabic, English, Esperanto, Estonian, Finnish, French, Galician, Georgian, German, Greek, Gujarati, Haitian, Hebrew, Hindi, Hungarian, Icelandic, Ido, Indonesian, Irish, Italian, Japanese, Javanese, Kannada, Kazakh, Kirghiz, Korean, Latin, Latvian, Lithuanian, Lombard, Low Saxon, Luxembourgish, Macedonian, Malagasy, Malay, Malayalam, Marathi, Min Nan Chinese, Minangkabau, Nepali, Newar, Norwegian (Bokmal), Norwegian (Nynorsk), Occitan, Persian (Farsi), Piedmontese, Polish, Portuguese, Punjabi, Romanian, Russian, Scots, Serbian, Serbo-Croatian, Sicilian, Slovak, Slovenian, South Azerbaijani, Spanish, Sundanese, Swahili, Swedish, Tagalog, Tajik, Tamil, Tatar, Telugu, Turkish, Ukrainian, Urdu, Uzbek, Vietnamese, Volapük, Waray-Waray, Welsh, West Frisian, Western Punjabi, Yoruba.

Wikipedia includes all these languages. CulturaX lacks Chinese (Traditional), Min Nan Chinese, Scots, and Crimean Tatar. fineweb-2 does not include Chinese (Traditional), English, Serbo-Croatian, or Tagalog. For the English subset in fineweb-2, we use the fineweb dataset[3].

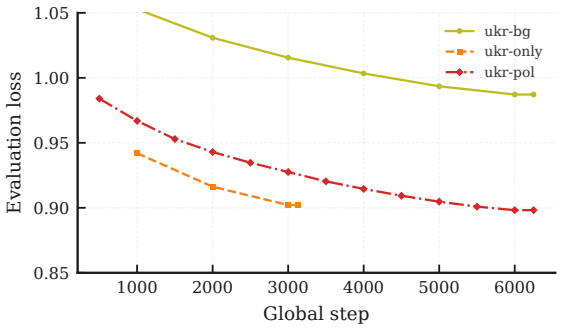

Figure 3: Evaluation loss on Ukrainian for three weighted-loss runs: *ukr-only* (baseline), *ukr-bg* (Ukrainian + Bulgarian), and *ukr-pol* (Ukrainian + Polish). Two-language datasets are twice as large, hence the longer training schedule.

C Transfer-Learning Experiments

We investigated whether adding data from a *similar* language can improve a low-resource target, where similarity is measured by the language-distance metric introduced in this paper. All experiments fine-tune Llama 3.2 1B and evaluate exclusively on a held-out set in the target language.

We perform our experiments using the following strategies:

1. **Mixed (size-matched).** An equal amount of auxiliary-language text is concatenated to the low-resource corpus; the joint data are shuffled and used for fine-tuning.

2. **Mixed (loss-weighted).** The same joint corpus is used, but the loss is re-weighted: e.g. 0.8 for target-language tokens and 0.2 for auxiliary-language tokens.

3. **Sequential.** Fine-tune first on the auxiliary language, then continue training on the low-resource corpus.

Figure 3 shows that augmenting Ukrainian with the metrically close Bulgarian does not improve evaluation loss, and Polish yields only a minor reduction.

A similar pattern emerges for sequential fine-tuning on Turkish followed by Crimean Tatar:

[3] https://huggingface.co/datasets/HuggingFaceFW/fineweb

perplexity drops from 5.48 (Crimean Tatar only) to 5.36, an insignificant change.

Across all settings, none of the three transfer regimes produced a consistent, significant gain over single-language fine-tuning. Future work should revisit these transfer strategies with substantially larger models and much larger datasets, where the benefits of distance-based language pairing may emerge more clearly.

D Additional Figures

Figure 4 displays the MST coloured by k-means clusters. We set $k = 18$ – one cluster for each category plotted in Figure 1 (15 natural families plus 3 constructed languages) – so that the cluster colours can be compared directly with the family colours. Most clusters coincide with their expected families, but not all. Notably, Turkish is grouped with Hungarian and Finnish rather than with the other Turkic languages.

Figure 5 uses HDBSCAN with a minimum cluster size of two. This gives 24 clusters. Crimean Tatar is treated as outlier, while Ukrainian now connects directly to Polish.

Figures 6 and 7 give two other views of the same data using t-SNE and UMAP. Like the MST, they highlight clear family groups.

Figure 8 shows the confusion matrix between k-means clusters and high-level language families. The clusters are first matched to families with the Hungarian algorithm for clearer alignment. Figure 9 presents the same matrix, but for the finer primary branches of each family.

Figure 4: MST of all languages. Colours show k-means clusters with $k = 18$ (one cluster for each language family).

Figure 5: MST of all languages. Colours show HDBSCAN clusters (minimum cluster size = 2). Points marked as outliers by the algorithm are left out.

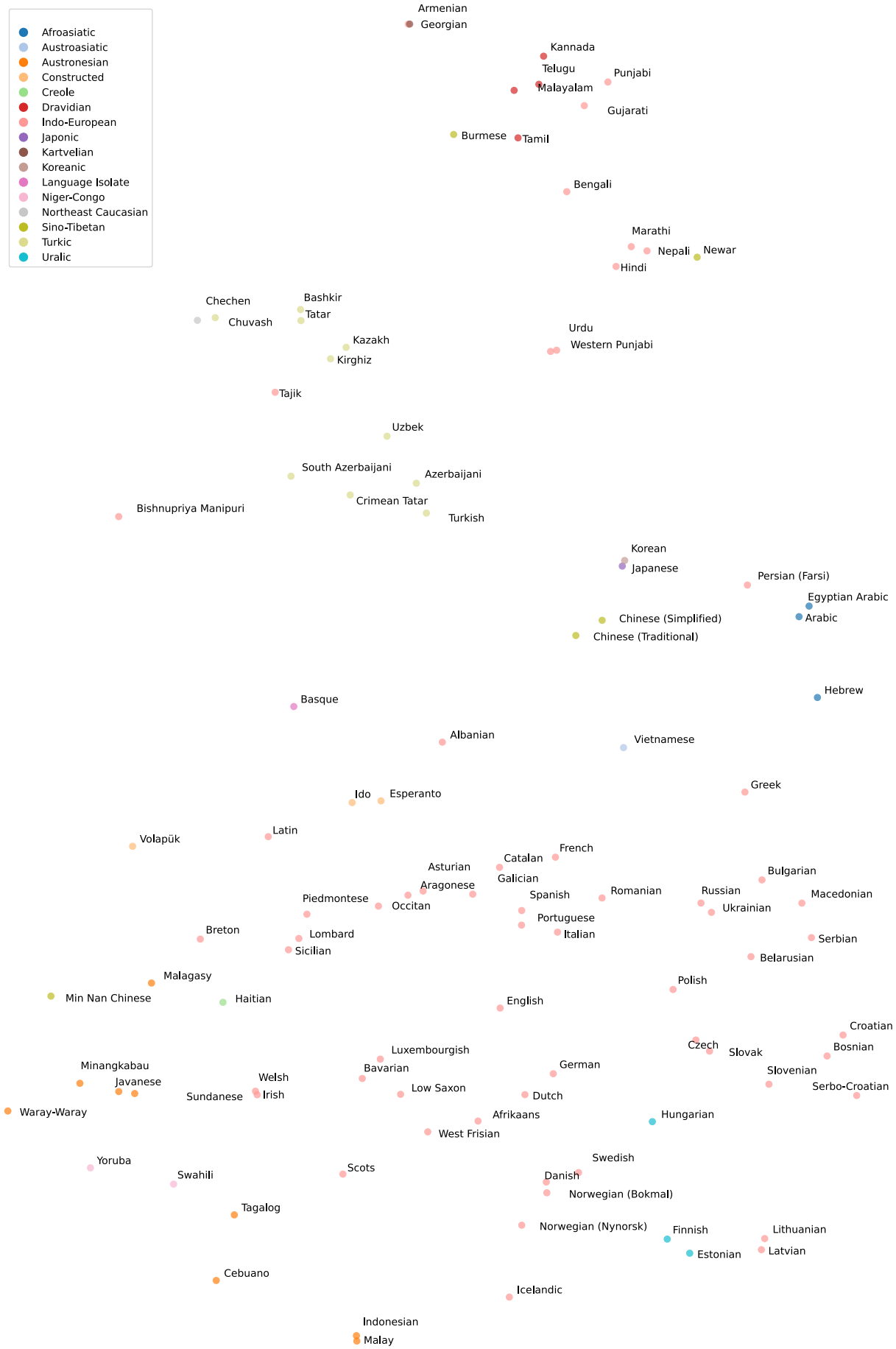

Figure 6: t-SNE plot of all languages. Colours show language families.

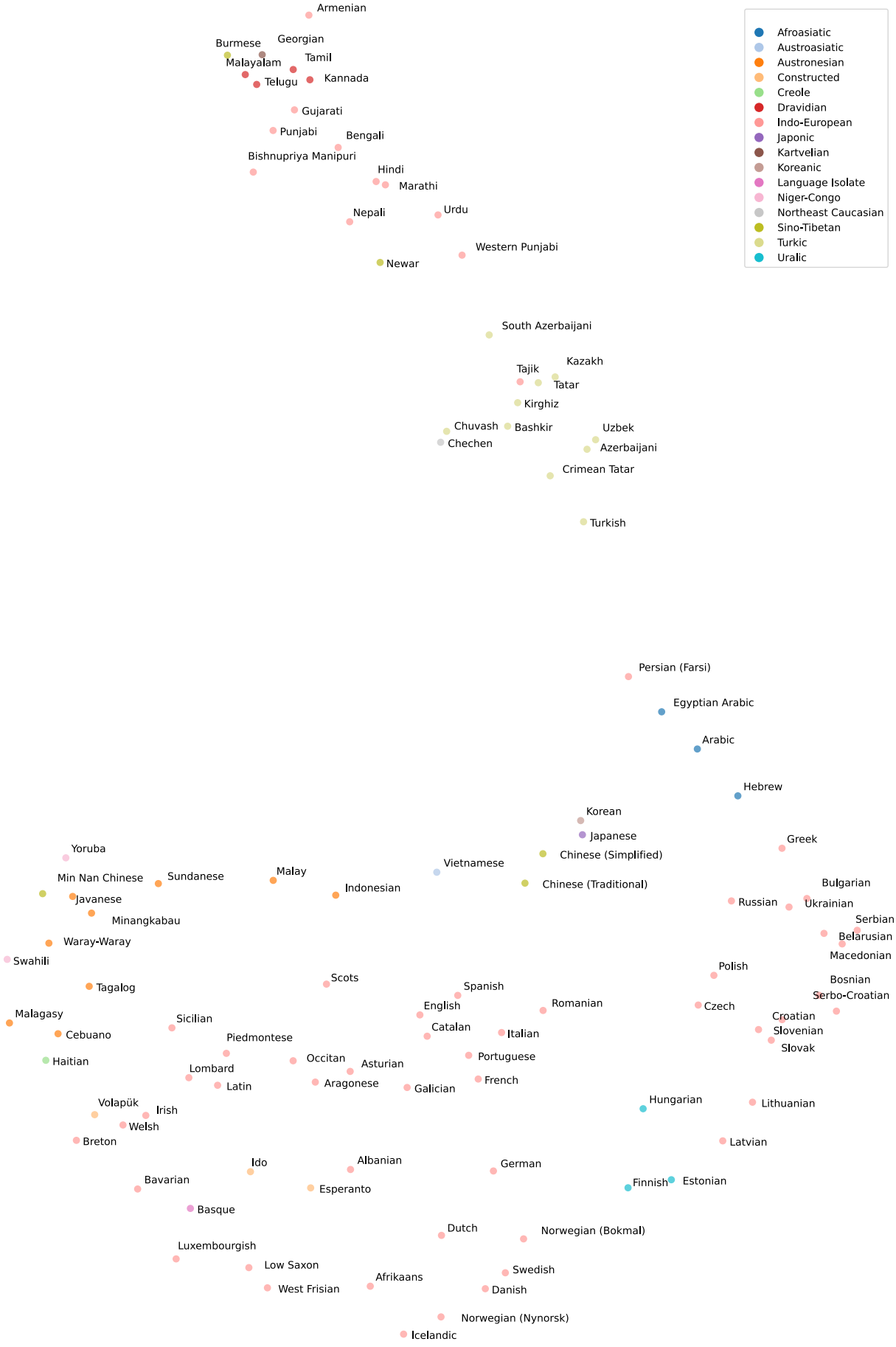

Figure 7: UMAP plot of all languages. Colours show language families.

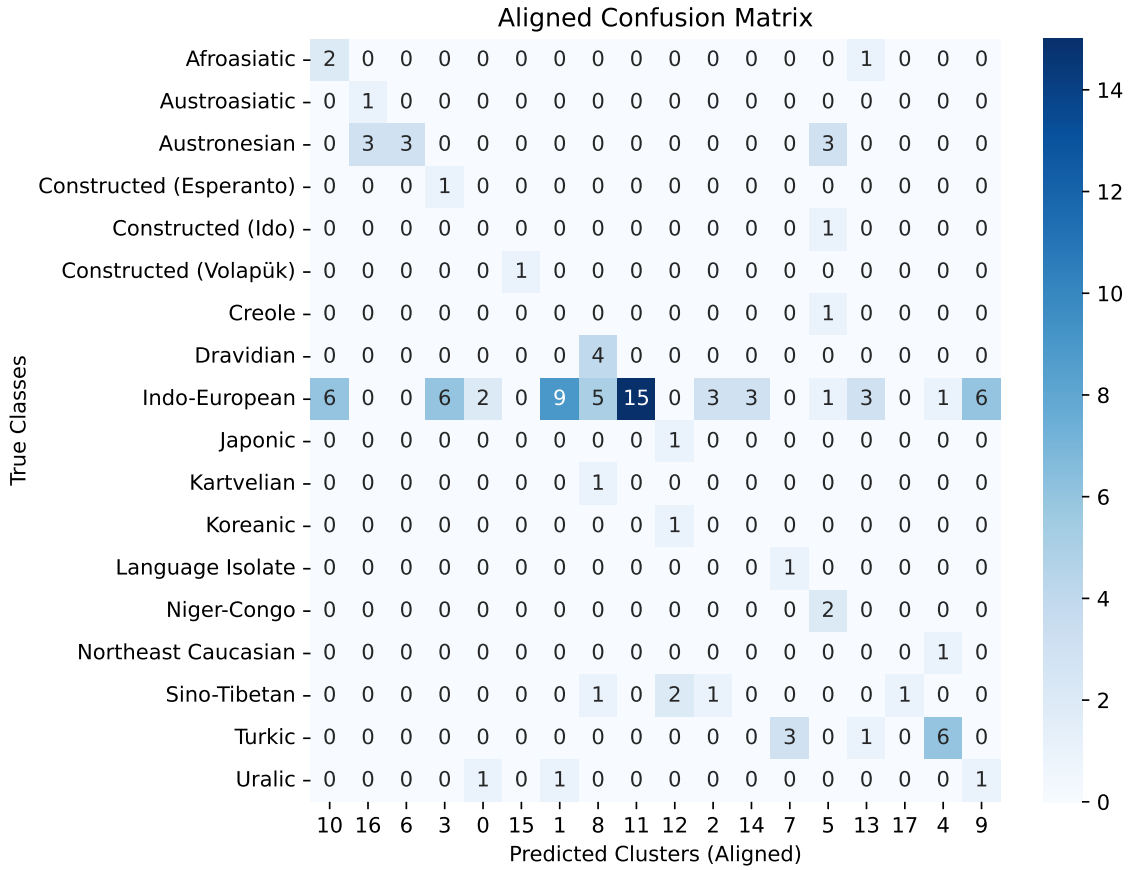

Figure 8: Adjusted confusion matrix between clusters obtained by k-means and macro families of languages. Number of clusters equal to 18.

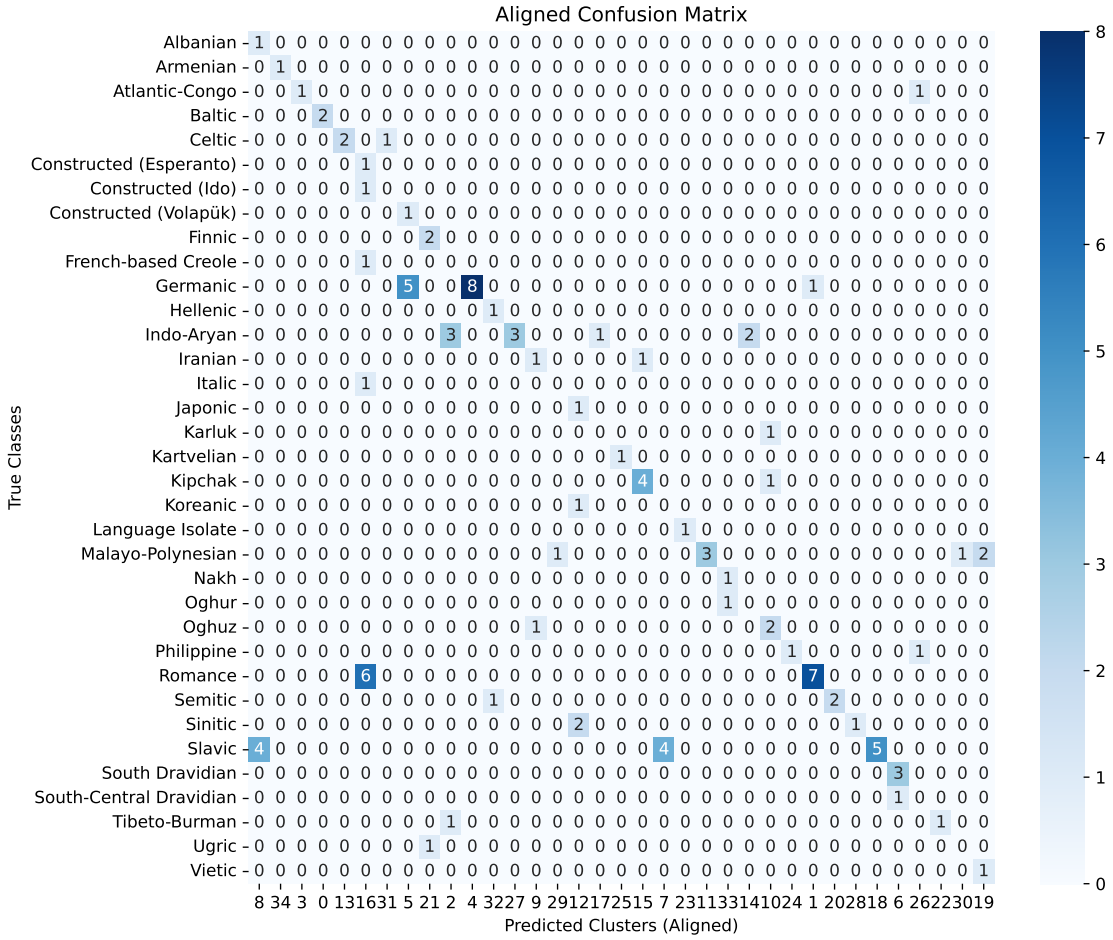

Figure 9: Adjusted confusion matrix between clusters obtained by k-means and primary branches of language families. Number of clusters equal to 35.