Stop Jostling: Adaptive Negative Sampling Reduces the Marginalization of Low-Resource Language Tokens by Cross-Entropy Loss

Galim Turumtaev
turumtaev.gz@gmail.com

Abstract

Neural language models often struggle with low-resource languages due to the limited availability of training data, making tokens from these languages rare in the training set. This paper addresses a specific challenge during training: rare tokens are disproportionately affected by marginalization, which prevents them from learning effectively. We propose a thresholding technique that reduces the impact of this marginalization, allowing rare tokens to benefit from more meaningful alignment. Through experiments with a character-level language model, we demonstrate that this method significantly improves performance on low-resource language validation data. This work is the first to show how negative sampling can be applied to improve the representation of rare tokens by limiting the harmful influence of excessive marginalization, offering a new approach to enhancing language model performance for underrepresented languages.

1 Introduction

Neural language models have revolutionized natural language processing (NLP), providing state-of-the-art results in a wide range of tasks, such as machine translation, text generation, and sentiment analysis. However, the effectiveness of these models heavily relies on the availability of large, high-quality datasets for pre-training. This dependency presents a significant challenge for low-resource languages, which often lack the extensive corpora needed for effective language model training.

One of the main issues faced by multilingual language models is the difficulty in learning effective representations for tokens from low-resource languages. These tokens, which occur infrequently during training, tend to receive less alignment and are more responsive to noise from irrelevant contexts. Recent studies have highlighted how this imbalance can negatively impact model performance

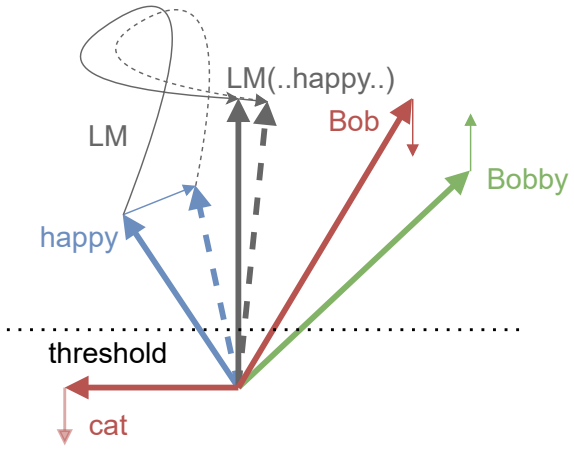

Figure 1: Three embeddings optimization types. Example for X = *'don't worry, be happy' by*; Y = *Bobby McFerrin*. **Context alignment (blue)**: adjust w_{happy} so that $g_\theta(\dots, w_{\text{happy}}, \dots)$ moves closer to w_{Bobby}. **Target alignment (green)**: move w_{Bobby} closer to $g_\theta(\dots, w_{\text{happy}}, \dots)$. **Non-target marginalization (red)**: move non-relevant w_{Bob} and w_{cat} away from $g_\theta(\dots, w_{\text{happy}}, \dots)$. The proposed method prevents marginalization of embeddings under the threshold.

(Chang et al., 2024). Existing solutions often focus on improving the general quality of embeddings (Gao et al., 2019) or limiting the influence of rare tokens on the overall training process (Yu et al., 2022).

In this paper, we identify a specific source of noise that affects rare tokens, which we call **marginalization**. Marginalization by cross-entropy loss pushes non-target embeddings away from irrelevant contexts and disproportionately impacts rare tokens, preventing them from learning meaningful representations. Unlike previous methods that address the impact of rare tokens on the overall model, we focus on reducing the negative impact *on* rare tokens themselves, which is a less explored but equally important problem.

To address this issue, we propose a simple yet

Proceedings of the First Workshop on Language Models for Low-Resource Languages (LoResLM 2025), pages 373–386
January 20, 2025.

effective adaptive negative sampling technique, which we call **thresholding**. By applying a threshold to the logits, the model effectively ignores non-relevant tokens during the training process, allowing those non-relevant tokens to receive more meaningful updates. This approach is novel in its use of negative sampling to improve the representations of unselected negative samples, rather than focusing solely on training efficiency or contrastive learning.

We validate thresholding effectiveness through experiments with a character-level multilingual language model trained on simulated mixed low-resource and high-resource language data. The results demonstrate that the proposed technique improves the representation and performance of rare tokens, making it particularly valuable for enhancing language models in low-resource settings.

The main contributions of this paper are as follows:

- Identification of **marginalization** as a key factor degrading the quality of rare token representations (§2).
- Introduction of a **thresholding** technique to mitigate marginalization (§3), with experiments showing improvements in language model performance on low-resource data (§4).

By addressing the challenges faced by tokens in low-resource languages, this paper presents a novel approach to improving multilingual language model performance, contributing to more balanced progress in NLP for underrepresented languages.

2 Problem

2.1 Intuition

Let us begin with an example. Consider the task of language modeling where the input prompt is the title of a song: *'Don't Worry Be Happy' by*. The goal is to predict the next word.

The correct continuation is *Bobby*, completing the sentence *'Don't Worry Be Happy' by Bobby McFerrin.* Now, let us reflect on the learning process of a language model:

1. **Was anything new learned about the word *Bobby*?** Yes, it was learned that *Bobby* is the nickname of the artist who performed *Don't Worry Be Happy*.
2. **Was anything new learned about the words *don't*, *worry*, *be*, *happy* and *by*?** Yes, it was learned that this sequence of words may be followed by *Bobby* in this context.
3. **Was anything new learned about the word *Bob*?** This song is often incorrectly attributed to Bob Marley. Now it was learned that *Bob* is not the correct nickname here.
4. **Was anything new learned about the word *cat*?** No, there is no new information about cats in this context.

In summary, while the model learns valuable associations between the correct words, irrelevant words, such as *cat*, should not be influenced by this example. Yet, because of **cross-entropy** loss, many modern language models still "learn" representations of irrelevant words, even when they don't belong in the context.

This issue relates to the **distributional hypothesis**, which states that words occurring in similar contexts tend to have similar meanings. Ideally, the model should learn word representations based only on relevant context. However, when using **cross-entropy** as the loss function, modern models tend to "push" non-target words, such as *cat*, slightly away from the model's last hidden state. Although this "push" is often small, in the case of rare tokens or low-resource languages, it can degrade the learned representations.

2.2 Formalization

Consider a vocabulary $V = \{v_1, v_2, \ldots, v_N\}$ of size N, and an embedding matrix $W = [w_1, w_2, \ldots, w_N]$, where row w_i corresponds to token v_i for each $i \in \{1, \ldots, N\}$. A training sentence is denoted as $(x_0, x_1, \ldots, x_M)$, with length $M + 1$, where $x_i \in V$ for each $i \in \{0, \ldots, M\}$. The last hidden state before the classification head, h_t, is produced by the model's body g_θ with parameters θ, based on the first t input tokens:

$$h_t = g_\theta(w_{x_0}, \ldots, w_{x_{t-1}})$$

When using **weight tying** (Press and Wolf, 2017), the probability of the token x_t is calculated by the language model as:

$$P_\theta(x_t | x_0, \ldots, x_{t-1}) = \frac{\exp(\langle h_t, w_{x_t} \rangle)}{\sum_{i=1}^{N} \exp(\langle h_t, w_i \rangle)}$$

Where $\langle a, b \rangle$ denotes the dot product of vectors a and b. During training, the cross-entropy loss:

$$\mathcal{L}_\theta(x_t) = -\log\left(P_\theta(x_t|x_0, \ldots, x_{t-1})\right)$$

is minimized for all $t \in \{1, \ldots, M\}$.

Embeddings W are optimized simultaneously in three distinct ways:

1. **Context alignment**: For all $k \in \{0, \ldots, t-1\}$, w_{x_k} is optimized to maximize $\frac{\exp(\langle g_\theta(\ldots, w_{x_k}, \ldots), w_{x_t}\rangle)}{\sum_{i=1}^{N} \exp(\langle g_\theta(\ldots, w_{x_k}, \ldots), w_i\rangle)}$. The gradient is $\frac{\partial \mathcal{L}_\theta(x_t)}{\partial h_t} \frac{\partial h_t}{\partial w_{x_k}}$.

2. **Target alignment**: w_{x_t} is optimized to maximize $\langle h_t, w_{x_t}\rangle$. The gradient is $\frac{\partial \mathcal{L}_\theta(x_t)}{\partial w_{x_t}}$.

3. **Non-target marginalization**: For all $v_i \in V$, where $v_i \neq x_t$, w_i is optimized to minimize $\langle h_t, w_i\rangle$. The gradient is $\frac{\partial \mathcal{L}_\theta(x_t)}{\partial w_i}$.

In the example in Figure 1, various tokens, including irrelevant ones such as *cat*, are affected by this third type of optimization, which we refer to as **marginalization**. As we show in §4.2, this noise may be significant for rare tokens and tokens from low-resource languages.

3 Method

3.1 Algorithm

To reduce marginalization, we propose a **thresholding** technique that is applied to the logits after the language model's classification head but before calculating the cross-entropy loss.

Let us revisit the song example. Assume that the model assigns probabilities as follows: $P_\theta(Bob) \gtrsim P_\theta(Bobby) \gg P_\theta(cat)$. Although it makes sense to lower the probability of *Bob*, the probability of *cat* is already very low and can be ignored. This allows the embedding of *cat* to align better in its own relevant contexts.

The core idea is to ignore the tokens v_i with $P_\theta(v_i) \ll P_\theta(x_t)$. This is achieved by thresholding logits based on a selected `margin` as described in Algorithm 1.

A simple and effective implementation of this algorithm in the PyTorch framework can be found in Appendix C.

By applying this thresholding, the probabilities $P_\theta(v_i) < P_\theta(x_t) \times e^{-\texttt{margin}}$ are effectively set to 0. This makes the marginalization gradients zero for the corresponding embeddings.

Algorithm 1 Thresholding Logits

Input: $logits, x, margin$
for each t in $[1, \ldots, M]$ **do**
  $threshold_t \leftarrow (logits_t[x_t] - margin)$
  for each i in $[1, \ldots, N]$ **do**
    if $logits_t[v_i] < threshold_t$ **then**
      $logits_t[v_i] \leftarrow -\infty$
    end if
  end for
end for

Referring back to Figure 1, after applying the threshold, the logits for the token *cat* become $-\infty$, so it is no longer marginalized. However, the logits for another token, *Bob*, remain above the threshold, meaning *Bob* will still be marginalized.

3.2 Hyperparameter

This method introduces a new hyperparameter `margin`. Although we do not cover the optimal choice of `margin` in this work, we provide an idea on how to limit the search range for `margin`.

Although the `margin` theoretically can be set to any value between 0 and $+\infty$, it is clear that as $\texttt{margin} \to +\infty$, the proposed method converges to standard **cross-entropy** loss. A large `margin` will have little to no effect on the performance of the model.

On the other hand, as $\texttt{margin} \to 0$, there will be a long tail of irrelevant tokens with small P_θ, that was not marginalized enough. Due to their large number, they will noticeably reduce $P_\theta(x_t)$, increasing the model's perplexity. We describe this phenomenon in more detail in §4.

Between these two extremes, there may be a range of suitable `margin` values that improve the representation of rare tokens without significantly affecting performance on frequent tokens.

Let $P_{\theta,T}(v_i)$ represent the probability of the token v_i after applying the temperature T. In Appendix A, we show that by choosing

$$\texttt{margin} = T \times \ln\left(\frac{(N-1) \times \texttt{top\_p}}{1 - \texttt{top\_p}}\right)$$

thresholding will not affect the tokens v_i with $P_{\theta,T}(v_i)$ within the `top_p` distribution of $P_{\theta,T}$.

Given the widespread use of nucleus sampling in modern language models, this may be sufficient to offset the negative impact on frequent tokens while still benefiting rare tokens. For example, for

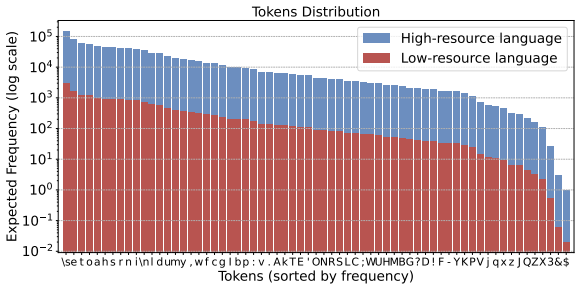

Figure 2: Token distribution in the experiment with a simulated low-resource (2%) and high-resource (98%) languages. For the high-resource language, expected frequencies range from 150,209.5 (token " ") to 0.98 (token "$"); for the low-resource language, expected frequencies range from 3,065.5 to 0.02.

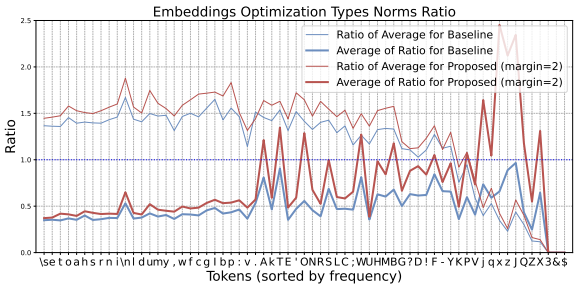

Figure 3: Ratio of embedding gradients norms for different tokens in the low-resource language. Tokens are sorted by frequency. Rare tokens have a lower **Ratio of Average**. In the baseline model, all tokens have an **Average of Ratio** below 1, indicating that **marginalization** has a strong effect on these tokens. The proposed method increases the **Average of Ratio** by 45% and the **Ratio of Average** by 12% on average.

nucleus sampling with $T = 0.9$, $\texttt{top\_p} = 0.99$, and vocabulary size $N = 100,000$, choosing

$$\texttt{margin} = 0.9 \times \ln\left(\frac{(100,000-1)\times 0.99}{1-0.99}\right) \approx 14.50$$

ensures that tokens appearing in the top 0.99 of the $P_{\theta,T}$ distribution will always be marginalized.

This choice may still be too conservative and might not provide enough improvement for rare tokens. However, this relatively low upper bound should make it easier to find a balanced `margin` between 0 and $T \times \ln\left(\frac{(N-1)\times \texttt{top\_p}}{1-\texttt{top\_p}}\right)$.

Similarly, for min-p sampling (Nguyen et al., 2024), by choosing $\texttt{margin} = T \times \ln(p_{\text{base}})$, thresholding will not affect the tokens v_i within the $\mathcal{V}_{\text{min}}$ set of min-p sampling, where p_{base} is a hyperparameter of min-p sampling.

4 Experiments

The code is available on GitHub[1]. For this project, we used the `nanoGPT` implementation by Karpathy[2]. We conducted experiments on a small dataset of Shakespeare's texts and trained a character-level language model. The dataset contains 65 unique characters, with a highly imbalanced distribution (Figure 2).

4.1 Data and Model

The Shakespeare dataset provides a toy example with significant imbalance in token occurrences. To simulate both high- and low-resource languages, following (K et al., 2020), we modified the character-level tokenizer. In 2% of randomly selected training sentences, we added $N = 65$ to character IDs, simulating a second "low-resource" language with token IDs ranging from 65 to 129. We selected this 2% ratio for the low-resource language as it is small enough to observe the negative impact of marginalization, yet realistic, as the second most popular language in GPT-3 pre-training data (French) accounts for about 2% of words[3]. Figure 2 shows the token distribution for both high-resource and low-resource simulations.

The following models were evaluated:

- **Baseline:** A GPT-2 architecture model with 800k parameters and weight tying.
- **Monolingual:** The baseline model trained solely on low-resource language data (2% of the training steps).
- **Proposed:** The baseline model with thresholding applied, tested with margins between 0 and 8 (approximating $1 \times \ln\left(\frac{(N-1)\times 0.95}{1-0.95}\right)$).
- **Proposed+SE:** The proposed model with separated embeddings for better handling by the AdamW optimizer (more details in §4.5).

The exact hyperparameters used in the experiment are listed in Table 4 in the Appendix.

4.2 Influence of Marginalization

First, we measured the influence of marginalization on each token. For all tokens, we interpreted

[1] https://github.com/turumtaev/StopJostling
[2] https://github.com/karpathy/nanoGPT
[3] https://github.com/openai/gpt-3/blob/master/dataset_statistics/languages_by_word_count.csv

the token's embedding optimization as the sum of the 3 types of optimization described above. For each embedding, we calculated its gradient from backpropagation as the sum of gradients from the 3 types of optimization. Knowing the gradients for each token and optimization type, we logged $\|grad_{type,i,step}\|$ — the norms of the gradients with type $type$ for embedding w_i in step $step$. Then, we calculated the following ratios:

- **Ratio of Average**:

$$\frac{\text{avg}_s(\|grad_{1,i,s}\| + \|grad_{2,i,s}\|)}{\text{avg}_s(\|grad_{3,i,s}\|)}$$

- **Average of Ratio**:

$$\text{avg}_s\left(\frac{\|grad_{1,i,s}\| + \|grad_{2,i,s}\|}{\|grad_{3,i,s}\|}\right)$$

Both ratios for tokens from the low-resource language are plotted in Figure 3, with tokens sorted by frequency. It is clear that the problem of marginalization exists for low-resource language data: the 14 least frequent tokens have **Ratio of Average** below 1, and all tokens have an **Average of Ratio** below 1. This indicates that for these tokens, the influence of marginalization is significant. The proposed method increases both ratios[4].

4.3 Results and Observations

The models were compared on the validation data using the following metrics:

- **PPL**: Character-level perplexity of $P_\theta(x_t)$.
- **PPL**$_{best}(T_{best})$: Best **PPL** of $P_{\theta,T}(x_t)$ among different T. The proposed method increases P_θ for the unreliable tail of tokens, and applying a lower temperature T typically helps. See §4.4 for more details and Appendix E for the proposed method to reduce the problem.
- **Accuracy, Recall@5, and Mean Reciprocal Rank (MRR)**: Although Baseline outperforms thresholded models in terms of **PPL**, there is other evidence suggesting that this is mainly due to an unreliable tail. The thresholded models rank the target token higher, even for high-resource language tokens.

[4]The proposed method makes $\|grad_{3,i,s}\| = 0$ for some (v_i, s), making it impossible to calculate the **Average of Ratio**. For such (v_i, s), $\|grad_{3,i,s}\|$ was estimated with $(\text{avg}_s(\|grad_{3,i,s}\|)$, which provides a lower bound for the **Average of Ratio**

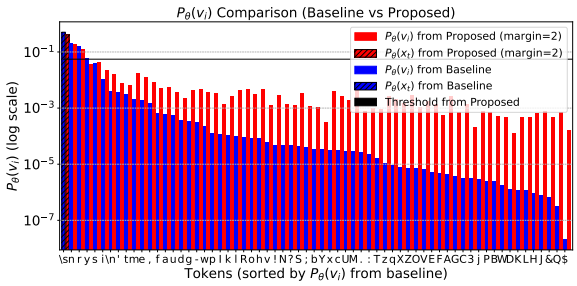

Figure 4: Example of real distribution of $P_\theta(v_i)$ from the Baseline and Proposed methods. Due to thresholding, $P_\theta(v_i)$ for non-relevant tokens is pushed down only until they fall below the threshold. This creates the issue of an unreliable tail, where even though $P_\theta(x_t)$ from the Proposed remains the highest among all tokens, its value is still lower than that of $P_\theta(x_t)$ from the Baseline.

- **I(W)**: Following (Mu and Viswanath, 2018), anisotropic embeddings may harm the quality of the language model (Yu et al., 2022). Thresholded models show better **I(W)** values, providing more isotropic embeddings.

Table 1 shows the results. Starting with a safe `margin` of 8, we observe an improvement in quality for low-resource languages as the margin decreases. The proposed method suffers from an unreliable tail, but training $P_{\theta,1/T_{best}}(v_i)$ may help reduce the problem, with slightly worse results for other metrics (Appendix E). **Separated Embeddings (SE)** (§4.5) further improve the performance of the language model in low-resource languages.

4.4 Long tail of tokens

PPL is a widely used metric to evaluate language models. However, thresholding naturally increases **PPL** due to the presence of a long tail of tokens. Figure 4 illustrates how the long tail of non-relevant tokens with higher probabilities can reduce the probability of the target token, thereby increasing the **PPL**. The thresholding process marginalizes these non-relevant tokens only until their probabilities fall below the threshold, creating what we refer to as an unreliable tail.

Today sampling methods such as nucleus sampling are widely used. Such methods exclude the long tail of tokens from generation. Similarly, reducing the temperature helps suppress the probability of the tail. In our experiments, we observe that after applying the optimal temperature T_{best}, the proposed method achieves a lower **PPL**$_{best}$ compared to the baseline.

Model	Lang.	PPL	$\text{PPL}_{best}(\text{T}_{best})$	Accuracy	Recall@5	MRR	I(W)
Baseline	HR	5.04	5.01 (1.08)	0.5187	0.8299	0.6541	0.8422
	LR	10.65	10.63 (0.95)	0.3147	0.6883	0.4803	0.4173
Monolingual	HR	-	-	-	-	-	-
(LR data only)	LR	12.24	12.08 (0.89)	0.2851	0.6632	0.4543	**0.8917**
Proposed	HR	5.02	5.00 (1.07)	0.5212	0.8296	0.6557	0.8394
(`margin`=8)	LR	10.69	10.67 (0.95)	0.3127	0.6907	0.4801	0.4450
Proposed	HR	**5.01**	**4.99** (0.94)	0.5231	0.8313	0.6571	0.8381
(`margin`=4)	LR	10.90	10.67 (0.87)	0.3244	0.6882	0.4871	0.6479
Proposed	HR	5.82	5.07 (0.71)	0.5291	0.8336	0.6614	0.8499
(`margin`=2)	LR	11.56	9.62 (0.67)	0.3581	0.7166	0.5161	0.6422
Proposed	HR	9.33	5.13 (0.48)	**0.5297**	0.8344	**0.6626**	0.8884
(`margin`=1)	LR	17.30	9.54 (0.46)	0.3714	0.7204	0.5255	0.7651
Proposed	HR	5.24	5.24 (**1.02**)	0.5241	0.8323	0.6579	0.8207
(`margin`=1, T=0.46)	LR	10.46	10.46 (**1.00**)	0.3459	0.7064	0.5060	0.6730
Proposed+SE	HR	5.86	5.08 (0.71)	0.5254	0.8318	0.6595	0.8892
(`margin`=2)	LR	**10.41**	8.41 (0.65)	0.3947	0.7417	0.5478	0.7092
Proposed+SE	HR	9.18	5.12 (0.48)	0.5274	0.8357	0.6615	0.8706
(`margin`=1)	LR	13.91	6.90 (0.44)	0.4544	0.7827	0.5986	0.7109
Proposed+SE	HR	14.94	5.15 (0.34)	0.5273	**0.8364**	0.6614	**0.8929**
(`margin`=0.6)	LR	19.94	**6.17** (0.32)	**0.4868**	**0.8090**	**0.6277**	0.7619

Table 1: Evaluation metrics for models on the validation dataset for high-resource (HR) and low-resource (LR) languages. The best result for each metric and language is bolded. As expected, a `margin` of 8 is too conservative and has minimal impact on performance. Metrics improve as the `margin` decreases, achieving the best result with a carefully selected `margin` $= 0.6$. Applying temperature scaling $T = T_{best}$ during training helps improve the perplexity **PPL**. The use of **Separated Embeddings (SE)** shows a significant improvement in model performance on low-resource languages.

4.5 Separated Embeddings

In Figure 3, we observe the ratio of embedding gradient norms. However, in practice, the actual ratio differs due to the use of the `AdamW` optimizer and its momentum calculations, which average gradients over multiple steps. Even after applying thresholding, the gradients applied are never exactly zero: `AdamW` treats embeddings as rows of a single matrix, meaning that if at least one embedding has non-zero gradients, the momentum for all embeddings will be updated, and those updated momenta will be applied.

To avoid this issue, we modified the setup by saving the embeddings as a list of weights, with each token having its own independent embedding vector. This allows `AdamW` to skip updates for an embedding w_i if there are no new gradients specifically for token v_i. This approach effectively isolates the updates for each token, ensuring that the optimization only affects tokens with relevant gradients.

Experiments show that using separated embeddings significantly improves model quality, with improvements in several key metrics: $\times 0.72$ $\textbf{PPL}_{best}$, $\times 1.22$ **Accuracy**, $\times 1.09$ **Recall@5**, and $\times 1.14$ **MRR**.

It should be noted that, unlike thresholding, **SE** only affects the optimization of **unselected** tokens. The significant improvement in quality by **SE** suggests that this improvement occurs precisely by reducing marginalization, and not by contrastive learning.

A simple implementation of the Separated Embeddings layer in the PyTorch framework is provided in Appendix C.

4.6 Learned "Translations"

Figures 5 and 6 demonstrate that the proposed method helps the model to learn meaningful relationships between characters in different languages. It brings the embeddings of the same character from high-resource and low-resource languages closer together.

Table 2 presents the top-3 neighbors based on cosine similarity for the embeddings of characters

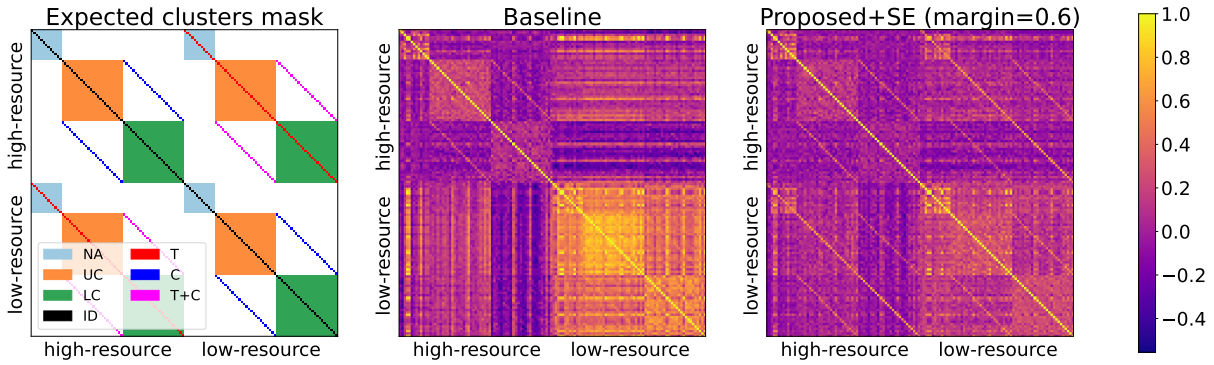

Figure 5: **Expected clusters mask and cosine similarity of embeddings**. The mask highlights clustering patterns: **NA** (13 non-alphabetical characters), **UC** (26 uppercase letters), **LC** (26 lowercase letters), **ID** (identity diagonal, always 1), **T** (translation of the same letter across languages), **C** (capitalization of the same letter), and **T+C** (capitalization of the same letter across languages). The embeddings are sorted by language and then alphabetically. The baseline model tends to marginalize low-resource (LR) embeddings, pushing them in the same direction. It only learns clusters and capitalization patterns for the high-resource language. In contrast, the proposed model captures all relationships described by the mask, revealing meaningful connections between characters across languages, **without any parallel corpus** in training data.

Model	$\mathbf{A}_{HR}$	$\mathbf{a}_{HR}$	$\mathbf{A}_{LR}$	$\mathbf{a}_{LR}$
Baseline	B_{HR} (0.41)	o_{HR} (0.38)	e_{LR} (0.82)	i_{LR} (0.86)
	E_{HR} (0.37)	i_{HR} (0.35)	O_{LR} (0.81)	o_{LR} (0.71)
	O_{HR} (0.36)	e_{HR} (0.34)	I_{LR} (0.77)	e_{HR} (0.71)
Proposed+SE (`margin=0.6`)	$\mathbf{A}_{LR}$ (0.54)	$\mathbf{a}_{LR}$ (0.69)	$\mathbf{A}_{HR}$ (0.54)	$\mathbf{a}_{HR}$ (0.69)
	$\mathbf{a}_{HR}$ (0.36)	$\mathbf{A}_{HR}$ (0.36)	B_{LR} (0.45)	i_{LR} (0.44)
	$\mathbf{a}_{LR}$ (0.32)	$\mathbf{A}_{LR}$ (0.28)	$\mathbf{a}_{LR}$ (0.43)	$\mathbf{A}_{LR}$ (0.43)

Table 2: Top-3 neighbors by cosine similarity for embeddings of characters from high-resource (HR) and low-resource (LR) languages. In this example, the proposed method places capitalizations and "translations" among the top-3 neighbors in 10 out of 12 cases.

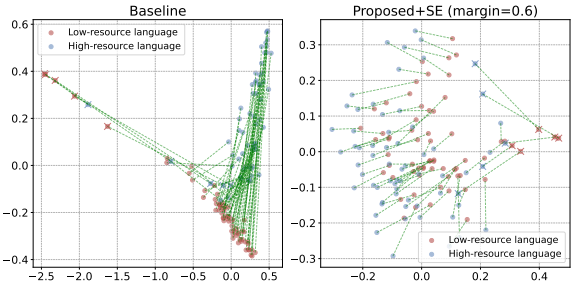

Figure 6: Comparison of PCA decomposition of embeddings. Embeddings from different languages are differentiated by color. The five embeddings of the rarest characters of each language are marked with crosses. Embeddings of the same character from 2 languages are connected with green lines.

A and *a* from both high-resource (HR) and low-resource (LR) languages. The model with thresholding and SE places capitalizations and "translations" of the character as the top-3 neighbors in 10 out of 12 cases, whereas the baseline model does so in 0 out of 12 cases.

This evidence shows that the proposed method not only improves ranking metrics but also helps the model learn more meaningful character representations across languages without any parallel corpus in the training data.

5 Related work

Chang et al. (2024) show that the addition of too much multilingual data can negatively impact the performance of language models in low- and high-resource languages due to limited model capacity, a phenomenon known as the **curse of multilinguality**.

Gao et al. (2019) explore the **representation degeneration problem**, where token embeddings

degenerate into a narrow cone, reducing the capacity of the model. To address this, they proposed **cosine regularization** to increase the expressiveness of the embeddings. Similarly, Zhang et al. (2020) propose using **laplacian regularization** to tackle the same issue.

Yu et al. (2022) link the **representation degeneration problem** with **anisotropy** and use **I(W)** to measure **isotropy**. The authors identify specific parts of the negative log likelihood loss gradient as the main cause of the problem, which aligns with the ideas presented in this paper. In addition, they propose **adaptive gradient gating (AGG)**. While the concept of **AGG** is similar to the thresholding technique proposed in this paper, **AGG** is more complex and requires counting token frequencies during training.

Negative sampling (NS) is widely used in many machine learning tasks (Yang et al., 2024). **NS** helps reduce computational complexity in tasks with large or "infinite" sample spaces, such as images or word-level tokenizers. For example, Mikolov et al. (2013) introduced random sampling to select negative samples during Word2Vec training. **NS** is also commonly used in contrastive learning: Godey et al. (2024) proposed **contrastive weight tying (CWT)**, which uses in-batch tokens as negatives. Contrastive learning is widely used to train sentence-level embeddings (Feng et al., 2022; Wang et al., 2024b; Sturua et al., 2024). Wang et al. (2024a) show how contrastive learning and in-batch negative sampling help to reduce the "language gap".

We applied some of the methods proposed in these related works and compared them with our approach. The corresponding metrics and comparisons are provided in the Appendix.

6 Conclusion

In this paper, we propose a method to improve the performance of language models in low-resource languages by reducing the impact of marginalization through logit thresholding.

The experimental results demonstrate significant improvements. The language modeling accuracy for the low-resource language increased from 0.31 with baseline to 0.49, which is close to the accuracy for the high-resource language (0.53). Additionally, the $\mathbf{PPL}_{best}$ for the low-resource language was reduced from 10.63 to 6.11, almost reaching the $\mathbf{PPL}_{best}$ for the high-resource language (4.97). The proposed approach not only improves performance metrics but also helps the model learn better representations, as evidenced by the alignment of "translations" of the same characters across different languages.

Furthermore, while previous work on negative sampling has primarily focused on enhancing training efficiency or improving the representation of positive examples, this method is, to the best of our knowledge, the first to show how negative sampling can directly improve the representation of non-sampled tokens.

7 Limitations of the work

We conducted experiments only with a small model and dataset. This introduces several limitations to the work.

Data and model size: Experiments with larger models could potentially alter the results. Chang et al. (2024) show that increasing the size of the model tends to improve the performance on multilingual data. At the same time, in Appendix B, we share our intuition about why a larger embedding dimension size could enhance the positive effect of thresholding.

Tokenizer: Using different tokenization techniques, such as Byte Pair Encoding (BPE), could affect the outcome. Since BPE alters the token distribution, Zouhar et al. (2023) demonstrate that the performance of the model correlates with the entropy of the token distribution generated by the tokenizer.

Languages: We tested the proposed method only on simulated multilingual data. Testing with real languages might lead to different results. Chang et al. (2024) also show that adding data from similar languages improves model performance more than adding data from dissimilar languages. In our simulated setup, each character in the original language has a corresponding character in the simulated language with exactly the same meaning. This 1:1 correspondence does not exist in natural multilingual data. However, we are optimistic that our method will still perform well with natural languages. As shown in Table 2, our thresholding approach brings lowercase and uppercase forms of the same character closer together. Importantly, capitalization does not rely on a 1:1 mapping. Based on this evidence, we believe thresholding has potential for success in real-world multilingual scenarios.

Downstream performance: While the pro-

posed method shows a significant improvement in the validation data, this does not necessarily guarantee improved performance in downstream tasks. Further testing on various downstream tasks is needed to confirm the method's effectiveness.

Model architecture: Although in our experiments we use a decoder transformer architecture, the method is not restricted to it. Since it modifies logits, a common component in many architectures, this method could also be applied to other model types.

Weight tying: While our explanation is tailored to models with weight tying, the method is not limited to such models. The results of a similar model without weight tying can be found in Appendix F.

Comparison with other methods: An extended comparison of metrics for further modifications of the proposed method, along with comparisons to related work, is available in Table 5 in the Appendix. However, not all methods from related works were implemented or hyperparameter-tuned well.

References

Shaked Brody, Uri Alon, and Eran Yahav. 2023. On the expressivity role of LayerNorm in transformers' attention. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14211–14221, Toronto, Canada. Association for Computational Linguistics.

Tyler A. Chang, Catherine Arnett, Zhuowen Tu, and Ben Bergen. 2024. When is multilinguality a curse? language modeling for 252 high- and low-resource languages.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Nathan Godey, Éric Villemonte de la Clergerie, and Benoît Sagot. 2024. Headless language models: Learning without predicting with contrastive weight tying. In *The Twelfth International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual bert: An empirical study. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Preprint*, arXiv:1301.3781.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. 2024. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *Preprint*, arXiv:2407.01082.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *Preprint*, arXiv:2409.10173.

Bo Wang, Scott Martens, and Alex C-G. 2024a. Bridging language gaps in multilingual embeddings via contrastive learning. `https://jina.ai/news/bridging-language-gaps-in-multilingual-embeddings-via-contrastive-learning/`. Accessed: 2024-10-22.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024b. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Zhen Yang, Ming Ding, Tinglin Huang, Yukuo Cen, Junshuai Song, Bin Xu, Yuxiao Dong, and Jie Tang. 2024. Does negative sampling matter? a review with insights into its theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5692–5711.

Sangwon Yu, Jongyoon Song, Heeseung Kim, Seongmin Lee, Woo-Jong Ryu, and Sungroh Yoon. 2022. Rare tokens degenerate all tokens: Improving neural text generation via adaptive gradient gating for rare token embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29–45, Dublin, Ireland. Association for Computational Linguistics.

Zhong Zhang, Chongming Gao, Cong Xu, Rui Miao, Qinli Yang, and Junming Shao. 2020. Revisiting representation degeneration problem in language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 518–527, Online. Association for Computational Linguistics.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

A Estimation of `margin`

Modern language models commonly utilize sampling techniques such as top-k sampling or nucleus sampling to eliminate the "unreliable tail" of low-probability tokens (Holtzman et al., 2020). By setting a top_p and temperature T for nucleus sampling, with a sufficient margin, the model can be trained so that $P_\theta(v_i)$ is optimized until $P_{\theta,T}(v_i)$ falls outside of top_p. Here, $P_{\theta,T}(v_i)$ represents the probability after applying temperature T.

Lemma 1: If there exists at least one token v_k, such that $P_\theta(v_k) > P_\theta(v_i)$ and

$$P_\theta(v_i) < \frac{1 - \texttt{top\_p}}{N-1},$$

then v_i is outside of top_p in nucleus sampling.

Proof: Assume the contrary — that v_i is inside top_p. This implies that v_i is not among the lowest probability tokens that make up $1 - \texttt{top\_p}$ of the distribution. Consequently, there exist n other tokens $\{v_{j_1}, \ldots, v_{j_n}\}$ such that for any v_j, $P_\theta(v_j) \leq P_\theta(v_i)$, and

$$1 - \texttt{top\_p} < P_\theta(v_i) + \sum_j P_\theta(v_j).$$

Since v_i and v_k cannot be v_j, it follows that $n \leq N - 2$ and

$$\begin{aligned} 1 - \texttt{top\_p} &< P_\theta(v_i) + \sum_j P_\theta(v_j) \\ &\leq (N-1) \times P_\theta(v_i) \\ &< (N-1) \times \frac{1 - \texttt{top\_p}}{N-1}. \end{aligned}$$

This results in a contradiction; therefore, v_i must be outside of top_p.

Lemma 2: If token v_i has been thresholded, then:

$$P_{\theta,T}(v_i) < P_{\theta,T}(x_t)\, e^{-\texttt{margin}/T}.$$

Proof:

$$\begin{aligned} P_{\theta,T}(v_i) &= \frac{P_\theta(v_i)^{1/T}}{\sum_V P_\theta(v_k)^{1/T}} \\ &< \frac{\left(P_\theta(x_t)\, e^{-\texttt{margin}}\right)^{1/T}}{\sum_V P_\theta(v_k)^{1/T}} \\ &= P_{\theta,T}(x_t)\, e^{-\texttt{margin}/T}. \end{aligned}$$

Lemma 3

For a margin defined as $\texttt{margin} = T \times \ln\left(\frac{(N-1)\times \texttt{top\_p}}{1-\texttt{top\_p}}\right)$, any thresholded token v_i will have $P_{\theta,T}(v_i)$ outside of the top_p distribution.

Proof:

- **Case 1**: If $P_{\theta,T}(x_t) \geq \texttt{top\_p}$, then $P_{\theta,T}(v_i)$ cannot be in top_p, as $P_{\theta,T}(x_t)$ already accounts for at least top_p of the probability mass.

- **Case 2**: If $P_{\theta,T}(x_t) < \texttt{top\_p}$, then, according to Lemma 2:

$$\begin{aligned} P_{\theta,T}(v_i) &< P_{\theta,T}(x_t)\, e^{-\texttt{margin}/T} \\ &< \texttt{top\_p}\, e^{-T \ln\left(\frac{(N-1)\times \texttt{top\_p}}{1-\texttt{top\_p}}\right)/T} \\ &= \texttt{top\_p}\, \frac{(1 - \texttt{top\_p})}{(N-1) \times \texttt{top\_p}} \\ &= \frac{1 - \texttt{top\_p}}{N-1}. \end{aligned}$$

Since $P_{\theta,T}(v_i) < \frac{1-\texttt{top\_p}}{N-1}$ and $P_{\theta,T}(v_i) < P_{\theta,T}(x_t)$, according to Lemma 1, $P_{\theta,T}(v_i)$ must be outside of top_p.

This margin estimation allows us to limit the search range between 0 and $T \times \ln\left(\frac{(N-1)\times \texttt{top\_p}}{1-\texttt{top\_p}}\right)$, which increases slowly as the vocabulary size N increases.

B `margin` and `d_model`

While Appendix A provides estimations for margin based solely on N, intuitively, the optimal margin should also depend on the dimension size of the embedding space d_model. To estimate the dependence of margin on d_model, we propose the following idea.

The intuition is that the effective margin should prevent embedding w_i from being marginalized in as many non-relevant contexts as possible. To model this behavior of the effective margin, let us denote the effective margin for each w_i as $\texttt{margin}_i$ and assume that we want $\texttt{margin}_i$ to prevent w_i from marginalization in 95% of non-relevant contexts. In other words:

$$\langle h_t, w_i \rangle < \langle h_t, w_{x_t} \rangle - \texttt{margin}_i$$

Model	d_model	$\texttt{margin}_{\text{init}}(\sigma)$	$\texttt{margin}_{\text{PT}}(\sigma)$	$\alpha_{\text{init}}(\sigma)$	$\alpha_{\text{PT}}(\sigma)$
Small	768	14.04 (0.02)	32.96 (5.4)	0.940 (1.6e-3)	0.029 (4.5e-3)
Medium	1024	18.93 (0.02)	49.05 (8.66)	0.948 (1.2e-3)	0.043 (7.1e-3)
Large	1280	23.83 (0.02)	70.04 (2.75)	0.954 (9.5e-4)	0.816 (0.047)
XL	1600	29.99 (0.02)	75.52 (2.28)	0.958 (7.7e-4)	0.840 (0.041)

Table 3: Estimation for effective margin and α for pre-trained (PT) and randomly initialized models from GPT-2 family. Increasing d_model increases the estimation of effective margin from Appendix B. For initialized models, α doesn't change much with the increase of d_model; after pre-training, α decreases without explicit dependency on d_model.

should be true for 95% of (h_t, w_{x_t}). We can rewrite this condition as:

$$\texttt{margin}_i < \langle h_t, w_{x_t} - w_i \rangle$$

Let $\mathcal{P}_{0.05}$ denote the 5th percentile operator. Then we want:

$$\texttt{margin}_i = \mathcal{P}_{0.05}(\langle h_t, w_{x_t} - w_i \rangle)$$

Having obtained $\texttt{margin}_i$, we can estimate the average effective $\overline{\texttt{margin}}$ as the average of all $\texttt{margin}_i$:

$$\overline{\texttt{margin}} = \frac{1}{N} \sum_{w_i \in W} \texttt{margin}_i$$

To sample (h_t, w_{x_t}) for each possible w_{x_t} we take $h_t = \texttt{LayerNorm}(\gamma \circ w_t)$, where LayerNorm is the final layer normalization in the transformer and γ is the weight in this layer normalization. This h_t gives $\langle h_t, w_t \rangle = \max_h(\langle h, w_t \rangle)$ (Brody et al., 2023).

In Table 3, we show an estimate of the effective margin for pre-trained GPT-2 models and for randomly initialized versions of the model.

We observe that, with random initialization, the effective margin grows approximately as $0.02 \times$ d_model. However, for trained models, the dependence is more complex, but still increases with increasing d_model. This suggests that as d_model increases, the same fixed margin (e.g., the margin from Appendix A) will become effective and provide more positive effects.

We noticed that the margin score mainly depends on $\|h_t\|\|w_{x_t}\|$. Therefore, we tried to use:

$$\overline{\alpha} = \frac{1}{N} \sum_{w_i \in W} \mathcal{P}_{0.05} \left(\frac{\langle h_t, w_{x_t} - w_i \rangle}{\|h_t\|\|w_{x_t}\|} \right)$$

This estimation of α remains almost constant with changes in d_model for a randomly initialized model. The metrics for experiments using the hyperparameter α can be found in Table 5.

C PyTorch implementations

```python
import torch

def thresholding(logits, targets, margin):
    threshold = torch.gather(logits, 2,
        targets.unsqueeze(-1)) - margin
    logits = torch.where(logits < threshold,
        torch.tensor(-float('Inf'),
        device=logits.device), logits)
    return logits
```

Listing 1: PyTorch implementation of Proposed method

```python
import torch
import torch.nn as nn

class SeparatedEmbedding(nn.Module):
    def __init__(self, num_embeddings,
        embedding_dim):
        super().__init__()
        self.weights = nn.ParameterList([
            nn.Parameter(
                torch.randn(embedding_dim)) for _
                    in range(num_embeddings)
        ])

    def forward(self, input):
        weight = torch.stack([w for w in
            self.weights]).requires_grad_(True)
        return weight[input]
```

Listing 2: PyTorch implementation of Separated Embeddings

D Hyperparameters

The hyperparameters used for the experiments are listed in Table 4.

E PPL vs. Temperature

Figures 7 and 8 illustrate the relationship between **PPL** and T for low- and high-resource languages. The plots highlight how the perplexity of different models, calculated for $P_{\theta,T}$, changes as the temperature is adjusted. These visualizations emphasize the impact of temperature scaling on model performance, with each model achieving its optimal **PPL** at different temperature values.

Hyperparamenter	Value
block_size	64
batch_size	12
n_layer	4
n_head	4
n_embd	128
max_iters	8000
lr_decay_iters	8000
dropout	0
eval_iters	20
eval_interval	250
learning_rate	1e-3
min_lr	1e-4
weight_decay	1e-1
beta1	0.9
beta2	0.99
grad_clip	1.0

Table 4: Hyperparameters for the GPT-2 arcitecture model used for experiments.

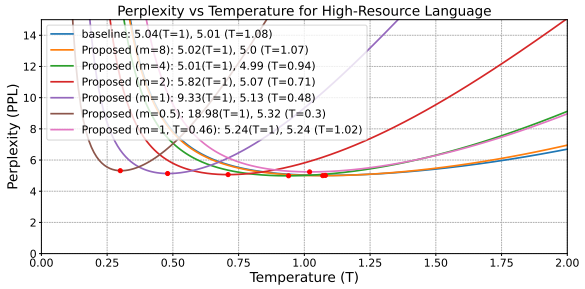

Figure 7: **PPL** vs. Temperature for high-resource language. The plot shows the **PPL** of different models as a function of temperature (T) for a high-resource language, calculated for $P_{\theta,T}$. **PPL** values are plotted for each model, highlighting the minimum perplexity achieved at various temperature levels.

The issue can be addressed with some decrease in quality by optimizing $P'_\theta = P_{\theta,1/T_{best}}$. The experiment shows that T'_{best} is 1 for P'_θ.

F Weight tying

Although we justified the usage of the proposed method using weight tying, the proposed method can also be applied to models without weight tying. Table 6 shows that the proposed method slightly improves the metrics for models without weight tying.

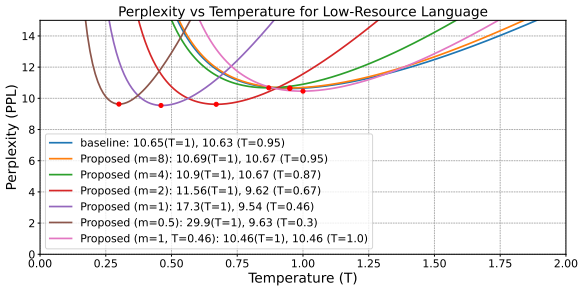

Figure 8: **PPL** vs. Temperature for low-resource language. The plot shows the **PPL** of different models as a function of temperature (T) for a low-resource language, calculated for $P_{\theta,T}$. **PPL** values are plotted for each model, highlighting the minimum perplexity achieved at various temperature levels.

G Other Modifications

G.1 Softminus

We tried to subtract $e^{-\text{margin}}$ from all $P_\theta(v_i)$ values above the margin:

$$P'_\theta(v_i) = \max(0, P_\theta(v_i) - e^{-\text{margin}}).$$

This operation ensures that $P'_\theta(v_i) \to 0$ when $P_\theta(v_i) \to e^{-\text{margin}}$. The results of this experiment are presented in Table 7.

G.2 Detached Logits Under Threshold

We attempted to eliminate logits below the threshold, but instead of fully removing them, we detached them to prevent marginalization gradient flow for rare tokens. The results of this experiment are shown in Table 8.

Model	Lang	PPL	$\text{PPL}_{best}(\text{T}_{best})$	Accuracy	Recall@5	MRR	I(W)
Baseline	HR	5.04	5.01 (1.08)	0.5187	0.8299	0.6541	0.8422
	LR	10.65	10.63 (0.95)	0.3147	0.6883	0.4803	0.4173
AGG	HR	5.01	4.99 (1.09)	0.5181	0.8303	0.6540	0.8142
($\alpha = 0.02$, K=1600)	LR	11.82	11.81 (0.97)	0.2915	0.6691	0.4605	0.5298
AGG	HR	5.01	4.98 (1.08)	0.5212	0.8308	0.6560	0.7926
($\alpha = 0.2$, K=1600)	LR	12.24	12.21 (0.95)	0.2875	0.6539	0.4539	0.2600
CosReg	HR	5.01	4.99 (1.08)	0.5199	0.8329	0.6558	0.8381
($\gamma = 1$)	LR	10.64	10.62 (0.95)	0.3177	0.6849	0.4817	0.5450
Adv	HR	5.04	5.02 (1.08)	0.5204	0.8294	0.6551	0.8275
($\alpha = 0.05$)	LR	10.81	10.77 (0.94)	0.3137	0.6857	0.4789	0.2514
CWT	HR	7.45	7.44 (1.04)	0.4133	0.7333	0.5586	0.7710
	LR	17.70	17.67 (0.95)	0.1949	0.4896	0.3442	0.6417
Proposed	HR	9.33	5.13 (0.48)	0.5297	0.8344	0.6626	0.8884
(margin=1)	LR	17.30	9.54 (0.46)	0.3714	0.7204	0.5255	0.7651
Proposed	HR	18.98	5.32 (0.30)	0.5208	0.8315	0.6554	0.8916
(margin=0.5)	LR	29.90	9.63 (0.30)	0.3716	0.7229	0.5269	0.7767
Proposed	HR	109.71	18.62 (0.02)	0.1610	0.5862	0.3574	**0.9581**
(margin=0)	LR	100.66	43.92 (0.12)	0.0594	0.4065	0.2162	**0.9352**
Proposed	HR	5.24	5.24 (**1.02**)	0.5241	0.8323	0.6579	0.8207
(margin=1, T=0.46)	LR	10.46	10.46 (**1.00**)	0.3459	0.7064	0.5060	0.6730
Proposed+SE	HR	9.18	5.12 (0.48)	0.5274	**0.8357**	0.6615	0.8706
(margin=1)	LR	13.91	6.90 (0.44)	0.4544	0.7827	0.5986	0.7109
Proposed+SE	HR	5.22	5.19 (1.07)	0.5243	0.8326	0.6584	0.8552
(margin=1, T=0.44)	LR	10.61	10.58 (1.05)	0.3439	0.7044	0.5041	0.5944
Proposed+α	HR	5.36	**4.97** (0.77)	**0.5317**	0.8354	**0.6640**	0.8660
(α=0.25)	LR	10.61	9.25 (0.71)	0.3720	0.7239	0.5266	0.6823
Proposed+α	HR	32.70	5.29 (0.19)	0.5213	0.8333	0.6572	0.9061
(α=0.0625)	LR	50.38	9.46 (0.18)	0.3759	0.7244	0.5296	0.7812
Proposed+α+SE	HR	31.75	5.25 (0.19)	0.5243	0.8330	0.6588	0.8895
(α=0.0625)	LR	38.50	**6.11** (0.17)	**0.4885**	**0.8132**	**0.6299**	0.8298

Table 5: Extended table with results from other papers and the proposed method with different hyperparameters.

Model	Lang	PPL	$\text{PPL}_{best}(\text{T}_{best})$	Accuracy	Recall@5	MRR	I(W)
Baseline	HR	**4.98**	**4.94** (1.11)	0.5240	0.8329	0.6583	0.8157
(w/o WT)	LR	8.82	8.79 (0.94)	0.3725	0.7247	0.5285	0.5403
Proposed	HR	5.80	5.07 (0.71)	0.5268	0.8333	0.6603	0.8699
(w/o WT, margin=2)	LR	10.75	8.94 (0.67)	**0.3784**	**0.7297**	**0.5342**	0.6188
Proposed	HR	9.43	5.12 (0.48)	0.5292	**0.8364**	0.6623	0.8693
(w/o WT, margin=1)	LR	16.43	9.14 (0.47)	0.3781	0.7285	0.5331	**0.6985**
Baseline+SE	HR	4.99	4.95 (1.11)	0.5219	0.8316	0.6567	0.6395
(w/o WT)	LR	**8.71**	**8.71** (0.98)	0.3722	0.7281	0.5279	0.3457
Proposed+SE	HR	5.61	4.98 (0.72)	**0.5304**	0.8344	**0.6630**	0.7570
(w/o WT, margin=2)	LR	10.42	9.00 (0.7)	0.3700	0.7294	0.5283	0.6182
Proposed+SE	HR	8.83	5.07 (0.49)	0.5284	0.8352	0.6623	**0.8721**
(w/o WT, margin=1)	LR	15.37	9.11 (0.49)	0.3767	0.7289	0.5321	0.6732

Table 6: Results for models without weight tying.

Model	Lang	PPL	$\text{PPL}_{best}(\text{T}_{best})$	Accuracy	Recall@5	MRR	I(W)
Proposed	HR	5.82	5.07 (0.71)	0.5291	0.8336	0.6614	0.8499
(margin=2)	LR	11.56	9.62 (0.67)	0.3581	0.7166	0.5161	0.6422
Proposed	HR	9.33	5.13 (0.48)	**0.5297**	0.8344	**0.6626**	0.8884
(margin=1)	LR	17.30	9.54 (0.46)	0.3714	0.7204	0.5255	**0.7651**
Softminus	HR	**5.81**	5.15 (0.73)	0.5287	0.8327	0.6604	0.8733
(margin=2)	LR	**11.47**	9.91 (0.71)	0.3623	0.7086	0.5169	0.6384
Softminus	HR	9.06	5.30 (0.51)	0.5281	0.8323	0.6604	0.8774
(margin=1)	LR	16.38	9.74 (0.5)	0.3750	0.7208	0.5279	0.7488
Softminus	HR	14.21	**4.95** (0.35)	**0.5297**	**0.8352**	0.6623	**0.9048**
(margin=2, C=5)	LR	25.44	9.13 (0.32)	0.3773	0.7233	0.5299	0.7358
Softminus	HR	30.72	4.99 (0.20)	0.5283	0.8336	0.6615	0.8844
(margin=1, C=5)	LR	46.76	**9.04** (0.18)	**0.3829**	**0.7249**	**0.5339**	0.6975
Softminus	HR	20.96	10.83 (0.47)	0.3741	0.6976	0.5218	0.0000
(DT, margin=2)	LR	49.71	39.13 (0.55)	0.1490	0.3650	0.2726	0.0000
Softminus	HR	47.19	18.71 (0.31)	0.2415	0.5607	0.3858	0.0000
(DT, margin=1)	LR	97.10	91.01 (0.59)	0.0000	0.1434	0.0803	0.0000
Softminus	HR	17.50	10.95 (0.39)	0.3228	0.6882	0.5017	0.4870
(DT, margin=2, C=5)	LR	44.69	31.67 (0.42)	0.1242	0.3982	0.2620	0.1795
Softminus	HR	33.91	10.55 (0.2)	0.3463	0.6956	0.6595	0.5017
(DT, margin=1, C=5)	LR	81.32	49.20 (0.22)	0.1417	0.3162	0.2464	0.2880

Table 7: Results for the softminus method. DT stants for detached threshold, an experiment where the threshold was detached before subtracting it from logits.

Model	Lang	PPL	$\text{PPL}_{best}(\text{T}_{best})$	Accuracy	Recall@5	MRR	I(W)
Proposed	HR	inf	inf (1.00)	0.4148	0.7565	0.5638	0.0002
(DUT, margin=2)	LR	18.39	18.25 (1.11)	0.2343	0.5446	0.3810	0.2724
Proposed	HR	15.78	15.71 (0.92)	0.2757	0.6368	0.4386	0.7866
(DUT, margin=1)	LR	29.09	28.98 (0.92)	0.1491	0.3771	0.2819	0.3075

Table 8: Results for experiments where logits under the threshold were not eliminated but only detached. DUT stands for detached under threshold.