

More Efficient Text Diffusion via Length Prediction

Davide Beltrame* Giacomo Cirò* Luca Gandolfi* Vittorio Rossi*

Bocconi University
Milan, Italy

{davide.beltrame,giacomo.ciro,luca.gandolfi,vittorio.rossi} @studbocconi.it

Abstract

Diffusion language models (DLMs) offer a promising alternative to autoregressive models (ARMs) for text generation, but their fixed-length decoding process leads to significant computational inefficiencies. We address this limitation by predicting an upper bound to the output sequence length before generation begins, and reduce the context window accordingly. Our approach relies solely on the internal representations of the model and explores both zero-shot and embedding-based techniques. When considering a state-of-the-art DLM, LLaDa-8B, a token-level classifier built on top of the encoded token embeddings successfully predicts an upper bound to the sequence length 80% of the times. Our results show that output length prediction is an effective and lightweight strategy to improve DLM efficiency, enabling significant computational savings with minimal overhead.

1 Introduction

Language generation with Large Language Models (LLMs) has traditionally been dominated by autoregressive models (ARMs), which generate text one token at a time (Zou et al., 2023). While effective, their sequential nature limits inference speed. Diffusion Language Models (DLMs) (Austin et al., 2021; Lou et al., 2024) have emerged as a promising alternative, offering potential for faster generation through a denoising process.

DLMs operate by iteratively unmasking a sequence of tokens, where initial tokens represent the prompt and remaining ones are placeholder mask tokens, revealed progressively. At each denoising step, the model predicts logits for the full masked sequence and only unmasks tokens when confident. The context length must be fixed at the start, and DLMs handle variable length output by appending

special end-of-sentence (EoS) tokens (Zou et al., 2023). This approach is effective, but computationally inefficient: the entire context window must be processed during each forward call of the denoising process, regardless of output length.

In this work, we focus on LLaDa (Nie et al., 2025), an 8B parameters DLM, and propose a simple yet effective method to reduce this inefficiency by predicting an upper bound for the generated sequence length at the initial stage of the denoising process and restricting the effective context to this predicted window, reducing unnecessary computations. We explore sentence-length prediction using the model’s internal signals in a zero-shot fashion by analyzing the logits corresponding to the (EoS) token. We then try embedding-based method such as a regression model using average prompt embeddings and a token-wise approach to classify each token as being EoS or not. For comparison, we fine-tune two DistilBERT models (Sanh et al., 2019) on predicting the length of the answers.

Our experiments show that the classifier based on LLaDa embeddings correctly predicted an upper bound on sentence length in over 80% of test cases. Although DistilBERT exhibited lower average error, it frequently underestimated the length, which is problematic for generation. In contrast, LLaDa tended to overestimate, which is safer since extra EoS tokens can still appear in the output. We also find the zero-shot approach to be a viable starting point for reducing computational overhead.

As LLMs continue to scale, improving inference efficiency becomes increasingly important. We show that DLM embeddings encode valuable signals for output length prediction, and presents a simple method to enhance DLM performance by leveraging existing model representations. Our code is available on GitHub¹.

*Equal contribution, the ordering is alphabetical.

¹<https://github.com/giacomo-ciro/diffusion-llms>

2 Methodology

We reduce the computational cost of DLMs by upper bounding the number of tokens required for generation, and truncating the context window to avoid unnecessary computation on padding EoS tokens. We frame this task as predicting the position of the EoS token and, since perfect accuracy is unlikely, methods that tend to overestimate the EoS position are preferable (for generation quality, overestimation is safer than underestimation). Accordingly, we evaluate various approaches based on their tendency to overestimate or underestimate the true sentence length.

To enhance existing DLM pipelines without modification, we focus on techniques that exploit only the model’s internal representations, and compare their performance against a fine-tuned DistilBERT baseline.

2.1 Model-native Methods

We initially experimented with DiffuGPT (Gong et al., 2025), an adapted GPT2-S (Radford et al., 2019) diffusion model. However, due to performance limitations and lack of support for variable-length output, we transitioned to the instruct version of LLaDa-8B (Nie et al., 2025), a scalable DLM with native variable-length support. We propose three techniques to predict an upper bound to the output length leveraging LLaDa embeddings.

Logit Quantile Heuristic A zero-shot method examining token-wise probabilities for the EoS token(see Appendix A, Figure 3). We predict the first token exceeding a fixed quantile threshold (0.25, 0.5, 0.75) as the end of sequence.

Embeddings-based Regression A two-layer MLP that regresses sequence length from the average token embeddings of the prompt.

Token-wise Classification A two-layer MLP classifying each token as EoS or not, using a probability threshold to identify the predicted endpoint.

2.2 DistilBERT Baseline

For comparison, we implement equivalent strategies using a pre-trained DistilBERT model:

DistilBERT Regressor A regression model trained on the [CLS] token (BERT’s classification

token) representations to predict sentence length.

DistilBERT Classifier A sentence-level classifier built on top of the [CLS] embedding, trained to predict sequence length from a predefined set of lengths (32, 64, 128, 512 and 1024).

2.3 Evaluation

We assess methods using *Bound Correctness* (share of valid upper bounds), *Tightness* (gap between prediction and true EoS), *Saved Tokens* (reduction from 1024-token context) and *RMSE* (between predicted and true EoS positions). *Bound Tightness*, *Saved Tokens* and *Root MSE* are measured only for the correct bounds.

Our evaluation assumes that overestimating the output length does not degrade generation quality. The model is expected to simply emit EoS for any surplus positions and should, by design, handle varying context windows. This makes conservative upper bounds preferable to underestimates, which could truncate meaningful content. Future work should focus on extensively analyzing the performance of DLMs as a function of the context length, verifying these models can adapt to varying context windows without quality loss.

3 Experiments

3.1 Dataset

To train our models, we use a dataset of approximately 15,000 prompts and corresponding answers, collected from real interactions with an ARM². This choice is motivated by our focus on predicting output length given a prompt, and using model-generated answers aligns well with this objective. Ideally, using prompt-answer pairs generated by LLaDa itself would offer an even closer fit, and we consider this a promising direction for future work. The dataset is split into training and test sets, with 5% of the training set reserved for validation.

We use the LLaDa tokenizer to compute the answer lengths, which serve as the ground truth targets for our prediction task. In the training set, the average answer length is 201 tokens, with a maximum of 862 tokens. In the test set, the average length is 200 tokens, with a maximum of 736.

²The dataset was provided by Professor Dirk Hovy during the course *Language Technology* (20879) at Bocconi University, Milan, Italy.

3.2 Model-native Methods

For computational reasons, we conduct our experiments on a restricted context window of 1024 tokens, which is shorter than the default value of 4096 tokens. This is not a limiting factor since this context length is never exceeded in our dataset.

Logit Quantile Heuristic For the zero-shot baseline, we forward the prompt using LLaDa’s fixed template by setting the first part of the context to be the prompt, and the rest masked tokens to be predicted. We extract the EoS probability at each masked position and select the first above the chosen quantile (either 0.25, 0.50, or 0.75).

We use the same masked input setup to train both a regressor and a classifier.

Embeddings-based Regression To obtain a sentence-level input for the regressor we average the hidden states over the entire sequence into a single pooled vector. We pass this vector through a multilayer perceptron, transform the labels using $\log(1 + y)$ for numerical stability and train using MSE loss in log space. During evaluation, we apply the inverse transform and compute evaluation metrics in the original token length scale.

Token-wise Classification Most tokens in the masked sequence are EoS, which makes this an imbalanced classification task. We tackle this problem using binary cross-entropy loss with different class weights. The classifier uses the same architecture as the regressor, but it is applied token-wise. For evaluation, we report results using a fixed 0.5 classification threshold over the predicted EoS probabilities.

Both methods use identical MLP architecture: a linear layer from 4096 to 2048, a GELU activation, and another 2048-layer that maps to a scalar output with sigmoid or ReLU.

3.3 DistilBERT baseline

As a baseline, we fine-tune DistilBERT in two ways:

DistilBERT Classifier We attach a two-layer MLP with hidden size 128 to the [CLS] embedding from the final hidden layer of DistilBERT, following standard practice. The model is trained to classify answers into one of six predefined length categories: 32, 64, 128, 256, 512, and 1024 tokens.

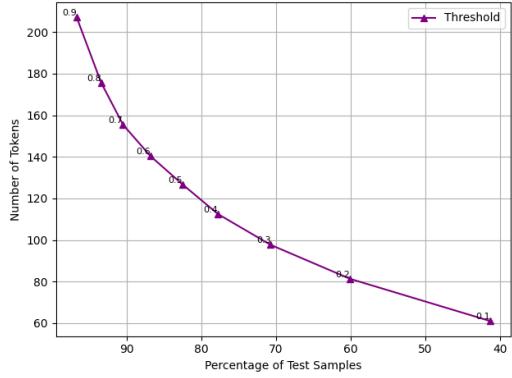


Figure 1: **Pareto Frontier (LLaDa Classifier)**. Points correspond to threshold values. By increasing the threshold, we correctly upper bound more test samples, but less tightly.

Training is done in two stages: first, the DistilBERT encoder is frozen and only the classification head is trained; then, the entire model is fine-tuned.

DistilBERT Regressor We attach a two-layer MLP with hidden size 128 to the [CLS] embedding, but train it to directly predict the exact answer length as a regression task. The training procedure mirrors that of the classifier.

4 Results & Discussion

We report the performance of the proposed methods in Table 1 (see also Appendix A, Table 3).

The LLaDa Classifier offers the best trade-off between bound correctness and tightness, providing valid upper bounds for over 80% of test samples with an average bound looseness of just over 120 tokens. While the DistilBERT Classifier achieves comparable correctness, its higher RMSE indicates more severe over/underestimation errors.

In terms of tightness, the DistilBERT Regressor achieves the lowest RMSE (91.68 tokens) and yields especially tight bounds when overestimating (71.16 tokens). However, it only produces valid bounds for 57% of samples. A similar trend holds for the LLaDa Regressor. This is because they were trained only to minimize RMSE with respect to the correct answer length, and a possible fix could be to add a constant to each prediction, thus trading off the MSE for improved percentage of upper bounds.

We also report the performance of a baseline that always predicts a constant value for the sequence length. This constant is selected post hoc to match

Table 1: **Length Prediction Approaches.** Comparison of different prediction mechanisms used for bounding sentence length. "% UB" refers to the percentage of test samples each method correctly estimates an upper bound for. "Tightness" quantifies how tight this upper bound is, in terms of the average number of tokens from the predicted upper bound and the true end of the sentence. "RMSE" is the square root of the mean squared error. "# Saved" is the average number of tokens saved by each method, measured as the average number of tokens from the predicted upper bound to the end of the context window (1024-th token).

Method	Variant	% UB	Tightness	RMSE	# Saved
DistilBERT	Classifier	81.75	158.49	183.83	768.00
	Regressor	57.46	71.16	91.68	869.00
LLaDa	Q25	59.06	117.33	141.42	775.00
	Q50	97.36	292.82	310.55	528.00
	Q75	99.88	510.05	520.31	472.00
	Classifier	82.51	126.69	137.33	762.00
	Regressor	46.38	63.97	106.24	869.35
Constant	312	82.09	153.15	164.91	712.00

the LLaDa classifier’s upper-bound prediction rate of 82%, enabling a clear comparison.³ The RMSE of this constant-length prediction is higher than that of the LLaDa classifier, indicating that our model outperforms this baseline by adapting the upper bound dynamically.

The zero-shot quantile heuristics (Q25, Q50, Q75) display a clear trade-off: higher quantiles yield higher correctness (Q75 achieves 99.88% valid bounds) but much looser estimates (~ 510 tokens for Q75). They still are still practical for efficiency, as simply discarding tokens whose predicted EoS probability exceeds the 75-th or 50-th percentile can save up to 500 tokens and reduce computations with zero training overhead.

Regarding tokens saved at generation time, regression methods offer the highest savings (both exceeding 800 tokens), while the LLaDa Classifier follows closely with around 770 tokens saved and demonstrating a stronger balance between accuracy and efficiency. In Figure 1, we further explore the LLaDa Classifier by plotting the Pareto frontier between bound correctness and tightness across classification thresholds (see also Appendix A, Figure 2). As expected, increasing the threshold improves correctness, but yields looser bounds.

On a 1024-token context window, the 80% sav-

ing rate achieved by our classifier is substantial—underscoring the significance of addressing inefficiencies in current DLMs.

The actual generation speed of LLaDa is evaluated and discussed in Appendix A.1.

5 Limitations & Future Work

Our work is subject to several limitations⁴, suggesting multiple directions for future research.

Our method requires DLMs with embeddings pre-trained for variable-length output generation. We began exploring ways to adapt existing diffusion language models to support variable-length generation, as discussed in Section 2, but this remains an open challenge. The same pretraining strategy can improve the ability of the model to predict EoS in a zero-shot setting, thus improving native length awareness.

Additionally, we did not test zero-shot EoS prediction at different stages of the denoising process. Evaluating predictions at later diffusion steps, when the model has partially formed an output, may yield improved confidence and accuracy.

Output quality as a function of the number of masked tokens can be explored as well. Our assumption that the model maintains performance under varying upper bounds is sensible, but needs

³In this case, the constant is similar to the mean plus one standard deviation of the answer lengths in our dataset. This could be considered a priori as a simple baseline.

⁴Computational constraints have been a significant bottleneck.

to be verified, especially given that attention scores may differ substantially between short and long contexts.

Our chosen dataset consist of clean, multilingual, conversational prompts. A more diverse and length-balanced dataset could yield more generalizable insights. Finally, we only superficially explored the model’s multilingual capabilities and its sensitivity to prompt phrasing (see Appendix A.4). Testing how EoS prediction responds to linguistic variation and trigger phrases such as ‘answer briefly’ or ‘respond in ten words’ represents a crucial direction for future exploration.

Finally, another possible direction of work is to add a position for the [CLS] token for LLaDa in a similar way as the one in BERT models and fine-tune LLaDa on sentence-level tasks so that the final embedding related to the [CLS] token encodes the meaning of the sentence and can be used for length prediction.

6 Related Work

Diffusion Models. The concept of diffusion models originates from non-equilibrium thermodynamics (Sohl-Dickstein et al., 2015). These models define a forward process that incrementally adds noise to data until it resembles pure noise, and a reverse denoising process that learns to transform noise back into data. Diffusion Probabilistic Models (DDPMs) (Ho et al., 2020) significantly advanced this area, demonstrating high-quality image generation.

Diffusion Language Models (DLMs). Adapting diffusion models to text has presented unique challenges. Early approaches include D3PMs (Austin et al., 2021), which use structured discrete corruption processes, and Diffusion-LM (Li et al., 2022), which applies diffusion in the continuous embedding space of words. Other notable DLMs include DiffuSeq (Gong et al., 2022) for sequence-to-sequence tasks, and DiffusionBERT (He et al., 2022), which integrates diffusion principles with pre-trained masked language models. A comprehensive overview of these and other models can be found in (Zou et al., 2023). The LLaDa model (Nie et al., 2025), central to this work, is a large-scale diffusion model trained from scratch, demonstrating competitive performance with ARMs.

Variable-Length Generation and Length Prediction. For NAR models, generating variable-length sequences often involves predicting an explicit length, training with EoS padding, but previous literature focuses on classical NLP tasks evaluation rather than analysis of efficiency (Gu et al., 2021). Predicting sequence length or determining when to stop generation prematurely based on internal model states has been explored to reduce computation in ARMs (Kaneko and Okazaki, 2023).

7 Conclusion

Our experiments addressed a significant limitation of DLMs by investigating how to upper bound the output sequence length.

The core challenge in predicting output length for efficiency is balancing the tightness of the predicted bound with the risk of underestimation, which can lead to premature truncation of the generated text, a more severe error than the computational overhead incurred by a slightly loose upper bound.

We showed that the problem of upper bound prediction can be successfully rephrased as a classification or regression problem on a DLM’s internal representations. We also investigated zero-shot methods, which involve inspecting the raw logits predicted by the model, and compared these to a baseline using external embeddings.

We found that a classifier relying on the DLM’s internal representation strikes the best balance between bound tightness and accuracy, also compared to methods relying on sentence-level DistilBERT embeddings. We argue that the DLM’s own internal representations likely encode signals related to sequence completion that an external model might not fully capture.

This study opens up several directions of future work. Adapting and evaluating these techniques on other DLMs will be crucial for assessing broader applicability, alongside quantifying true efficiency gains through wall-clock time and energy reduction metrics on diverse hardware.

We demonstrated that predicting output sequence length is a viable strategy for enhancing the efficiency of DLMs like LLaDa, showing the potential of zero-shot and ad-hoc solutions to address the computational challenges in deploying large-scale generative models.

References

- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. **Structured denoising diffusion models in discrete state-spaces**. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28300–28313.
- Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. 2025. **Scaling diffusion language models via adaptation from autoregressive models**. In *International Conference on Learning Representations (ICLR)*.
- Shansan Gong, Shijie Li, Kyunghyun Cho, and Graham Neubig. 2022. **Diffuseq: Sequence to sequence text generation with diffusion models**. *arXiv preprint arXiv:2210.08933*.
- Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Su. 2021. **BERT-crf for non-autoregressive text generation with dynamic output length**. *arXiv preprint arXiv:2102.08220*.
- Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. **Diffusionbert: Improving generative masked language models with diffusion models**. *arXiv preprint arXiv:2211.15029*.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. **Denoising diffusion probabilistic models**. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Masahiro Kaneko and Naoaki Okazaki. 2023. **Reducing sequence length by predicting edit operations with large language models**. *Preprint, arXiv:2305.11862*.
- Xiang Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. **Diffusion-LM improves controllable text generation**. In *Advances in Neural Information Processing Systems 35, NeurIPS 2022, November 28 - December 9, 2022, New Orleans, LA, USA*, pages 3215–3230.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. **Discrete diffusion modeling by estimating the ratios of the data distribution**. *Preprint, arXiv:2310.16834*.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. **Large language diffusion models**. *arXiv preprint arXiv:2502.09992*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. **Language models are unsupervised multitask learners**.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. **Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter**. *arXiv preprint arXiv:1910.01108*.
- Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. **Deep unsupervised learning using nonequilibrium thermodynamics**. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 37:2256–2265.
- Hao Zou, Zae Myung Kim, and Dongyeop Kang. 2023. **A survey of diffusion models in natural language processing**. *arXiv preprint arXiv:2305.14671*.

A Appendix

A.1 Performance Test

We evaluate LLaDa’s generation performance using our DistilBERT-based regressor and compare it against a fixed-generation length baseline. Here, the maximal length of the answer is denoted by generation length, while the final context length is the sum of the prompt length and the generation length.

The goal of this test is to assess:

1. How the generation length affects generation speed.
2. The performance gain from our length prediction pipeline.
3. The impact of the generation length on the final answer length.

Experiments are conducted on a single NVIDIA A100 GPU using the first 50 prompts from our test set. We employ LLaDa’s semi-autoregressive generation pipeline with a block size of 32 and low-confidence re-masking. Diffusion steps are set equal to the generation length, ensuring that only one token is unmasked per step. While this setup slows generation, it maximizes answer accuracy.

To apply dynamic generation length, we use the output of the DistilBERT regressor, round it to the nearest multiple of the block size, which is 32, and use the result as the generation length.

Results are reported in Table 2. For computational reasons, we did not test it with a fixed generation length of 1024 tokens. However, the advantages of using a length prediction model are already evident when compared to a fixed length of 512.

Increasing the maximum generation length significantly raises overall generation time and slows down per-token generation speed, particularly at 512 tokens. As expected, a model that can accurately predict the appropriate answer length allows for more efficient use of computational resources. The DistilBERT regressor obtains better average answer tokens speed generation even with respect to fixed generation length of 128 and 256, with the additional advantage that allows larger generation length if needed.

Moreover, we observe that the length of the generated answer is strongly influenced by the generation length; on average, it covers around 80% of

the available context, regardless of its size. This means the first <EoS> token is rarely predicted far from the end of the context window, indicating that the LLaDa approach to variable-length generation may be suboptimal. This finding reinforces the motivation behind our work: to restrict the context window based on the ground truth answer length. It highlights the importance of accurately predicting answer length and motivates further research in this direction.

A.2 DiffuGPT Continual Pre-training

Following the LLaDa methodology (Nie et al., 2025), we adapt DiffuGPT to predict end-of-sequence (EoS) tokens by exposing it to variable sequences with padding EoS tokens. To ensure smooth training, we gradually increase the percentage of EoS tokens among the masked ones from 0% to 50%.

A.3 Supplementary Figures

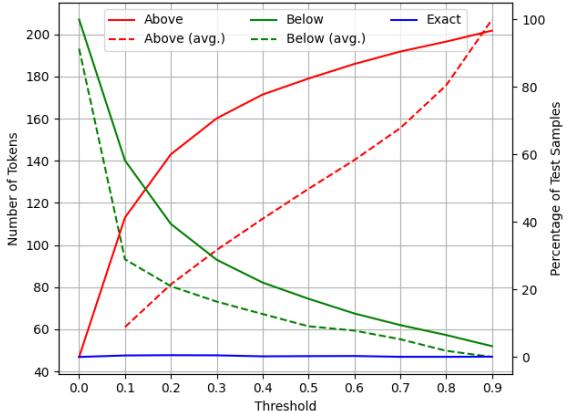


Figure 2: Threshold Analysis (LLaDa Classifier). Metrics are reported as a function of the value used for thresholding the probability estimated by the model. "Below", "Above", "Exact" refer to the percentage of test samples whose length we underestimate (lower bound), overestimate (upper bound), or predict exactly, respectively. "Above (avg.)" and "Below (avg.)" refer to the tightness of the upper and lower bounds respectively (measured in number of tokens from the true end of the sentence). For example, when all tokens with probability of being EoS is above 0.6 are predicted as EoS, we correctly upper bound 95% of the test samples, but with a loose bound of 180 tokens on average.

Table 2: **Performance Test.** "Answer Length" refers to the length of the model output up to the first EoS token. "Ratio" refers to the ratio between "Answer Length" and the entire generated response (including EoS tokens). "s/answer" is the average time needed to generate one answer. "tokens/s" is the speed of generation for non-EoS tokens in the answer.

Method	Variant	Answer Length	EoS Tokens	Ratio	s / answer	tokens / s
Fixed	64	51.4	12.6	0.803	2.05	25.0
	128	88.6	39.4	0.692	4.40	20.1
	256	213.0	43.0	0.832	9.97	21.3
	512	395.0	117.0	0.771	31.6	12.5
DistilBERT	Regressor	175.0	30.5	0.778	7.68	22.8

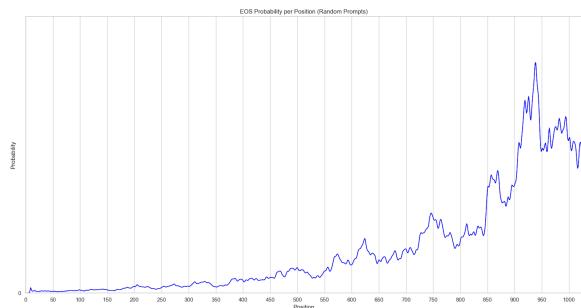


Figure 3: **Probability Distribution of the EoS Token.**

As expected, later positions in the context are much more likely to be EoS token, which motivates our heuristic for the zero-shot approach. The plot is obtained by averaging the distribution over 100 encoded sentences.

A.4 Multilingual EoS Distributions

We provide the starting point for a quantitative analysis of EoS token probability distributions which reveals several language-specific and cross-linguistic patterns with implications for sequence length prediction in multilingual contexts. This work can be expanded by implementing a detailed analysis of the statistical significance of our findings and exploring more languages. We summarize the results in Figure 4 and Figure 5.

Multilingual Prompt Dataset Our analysis of EoS token probability distributions was conducted using a balanced prompt dataset consisting of 60 prompts across three languages (English, Italian, and German) and two prompt types (regular and explanation triggers). For each language, we created 10 regular prompts requesting information and 10 paired explanation prompts containing explicit explanation triggers.

English Prompts. Regular prompts request information directly (e.g., "Tell me about climate change"), while trigger prompts employ the explanation directive "Explain" (e.g., "Explain climate change").

Italian Prompts. The Italian dataset employs a similar structure with information requests in regular prompts and the Italian explanation directive "Spiega" in trigger prompts.

German Prompts. The German dataset employs a similar structure with information requests in regular prompts and the German explanation directive "Erkläre" in trigger prompts.

Methodology. To analyze the EoS token probability distributions, we use the prompts as input to the model, and collect the output probability assigned to the EoS token at each position in the context window. For each prompt category (language and trigger type), we compute the average EoS probability across all 10 prompts at each position. We then plot these distributions to visualize differences across languages and prompt types, focusing on positions 600-1024 where most of the probability mass is observed. This method allows to visually inspect and quantify the positional shifts and magnitude changes in EoS probability when using explanation triggers compared to regular prompts across different languages.

Cross-linguistic Hierarchical Structure. The data reveals a clear hierarchical ordering of EoS token probabilities:

$$P_{EoS}^{English} > P_{EoS}^{German} > P_{EoS}^{Italian}$$

Table 3: **Full Results.** We report all the measurements to understand different prediction mechanisms used for bounding sentence length. "% Test Samples" refers to the percentage of test samples for which each method correctly estimates a lower bound, upper bound or the exact end. "Tightness" quantifies the tightness of the bounds, in terms of the average number of tokens from the specified bound and the true end of sequence. "RMSE" is the square root of the mean squared error. "# Saved" is the average number of tokens from the end of the context window and the predicted bound, computed only for the test samples which were correctly upper bounded.

Method	Variant	% Test Samples			Tightness		RMSE ↓	# Saved ↑
		LB↓	UB↑	Exact↑	LB↓	UB↓		
DistilBERT	Classifier	18.03	81.75	0.22	69.55	158.49	183.83	768.00
	Regressor	42.10	57.46	0.44	72.78	71.16	91.68	869.00
LLaDa	Q25	40.66	59.06	0.28	115.65	117.33	141.42	775.00
	Q50	2.64	97.36	0.00	104.63	292.82	310.55	528.00
	Q75	0.12	99.88	0.00	173.67	510.05	520.31	472.00
	Classifier	17.25	82.51	0.24	61.45	126.69	137.33	762.00
	Regressor	53.62	46.38	0.00	98.72	63.97	106.24	869.35
	Constant	312	17.91	0.18	74.27	153.15	164.91	712.00

Table 4: English prompt pairs used in the EoS probability analysis

Regular Prompt	Explanation Trigger
Tell me about climate change	Explain climate change
What is the capital of France	Explain why Paris is the capital of France
How does photosynthesis work	Explain photosynthesis
Define artificial intelligence	Explain artificial intelligence
Summarize the plot of Hamlet	Explain the plot of Hamlet
Who was Albert Einstein	Explain Albert Einstein's contributions
Describe the water cycle	Explain the water cycle
What happened during the French Revolution	Explain the causes of the French Revolution
List the planets in our solar system	Explain our solar system's planets
How does the immune system function	Explain how the immune system works

Table 5: Italian prompt pairs used in the EoS probability analysis

Regular Prompt	Explanation Trigger
Parlami del cambiamento climatico	Spiega il cambiamento climatico
Qual è la capitale della Francia	Spiega perché Parigi è la capitale della Francia
Come funziona la fotosintesi	Spiega la fotosintesi
Definisci l'intelligenza artificiale	Spiega l'intelligenza artificiale
Riassumi la trama di Amleto	Spiega la trama di Amleto
Chi era Albert Einstein	Spiega i contributi di Albert Einstein
Descrivi il ciclo dell'acqua	Spiega il ciclo dell'acqua
Cosa è successo durante la Rivoluzione Francese	Spiega le cause della Rivoluzione Francese
Elenca i pianeti del nostro sistema solare	Spiega i pianeti del nostro sistema solare
Come funziona il sistema immunitario	Spiega come funziona il sistema immunitario

Table 6: German prompt pairs used in the EoS probability analysis

Regular Prompt	Explanation Trigger
Erzähl mir über den Klimawandel	Erkläre den Klimawandel
Was ist die Hauptstadt von Frankreich	Erkläre warum Paris die Hauptstadt von Frankreich ist
Wie funktioniert die Photosynthese	Erkläre die Photosynthese
Definiere künstliche Intelligenz	Erkläre künstliche Intelligenz
Fasse die Handlung von Hamlet zusammen	Erkläre die Handlung von Hamlet
Wer war Albert Einstein	Erkläre Albert Einsteins Beiträge
Beschreibe den Wasserkreislauf	Erkläre den Wasserkreislauf
Was geschah während der Französischen Revolution	Erkläre die Ursachen der Französischen Revolution
Liste die Planeten in unserem Sonnensystem auf	Erkläre die Planeten unseres Sonnensystems
Wie funktioniert das Immunsystem	Erkläre wie das Immunsystem funktioniert

and this is consistent for all the positions in the context window. English exhibits approximately $2.7 \times$ higher peak probability (5.14×10^{-3}) compared to Italian (1.92×10^{-3}), suggesting the model has developed more confident representations of sequence boundaries in English—likely reflecting its predominance in the training corpus.

Language-Specific Trigger Effects. We observe a difference in how different languages respond to explanation trigger words:

- **Positional Invariance in English:** For English, both regular prompts and "Explain" trigger prompts exhibit maximum EoS probability at precisely the same position ($x_{max} = 938$), indicating the model's expected output length is not influenced by the instruction to explain. However, we observe a 14.4% reduction in maximum probability ($5.14 \times 10^{-3} \rightarrow 4.40 \times 10^{-3}$), suggesting slightly reduced confidence in sequence termination.
- **Consistent Positional Shift in Non-English Languages:** Both Italian and German demonstrate a uniform +4 token positional shift when trigger words are introduced (Italian: $942 \rightarrow 946$; German: $940 \rightarrow 944$). This systematic shift suggests the model has learned that explanation requests in these languages typically require slightly longer responses—a nuance not reflected in its English representations.

Probability Magnitude Effects. Across all languages, trigger-word prompts result in attenuated maximum probabilities compared to their regular counterparts (English: -14.4%, Italian: -24.5%, German: -10.7%). This consistent reduction in probability mass at the most likely termination po-

sition suggests the model exhibits greater uncertainty about sequence boundaries when explicitly prompted to explain concepts.

Distributional Shape Conservation. Despite the differences noted above, the overall morphology of EoS probability distributions remains remarkably consistent across languages, showing a multimodal pattern with three distinct peaks.

Implications for Length Prediction. These findings have direct applications to the techniques proposed in Section 2. The classification approach must account for trigger-word and language-specific effects, potentially incorporating prompt-type detection to adjust classification thresholds.

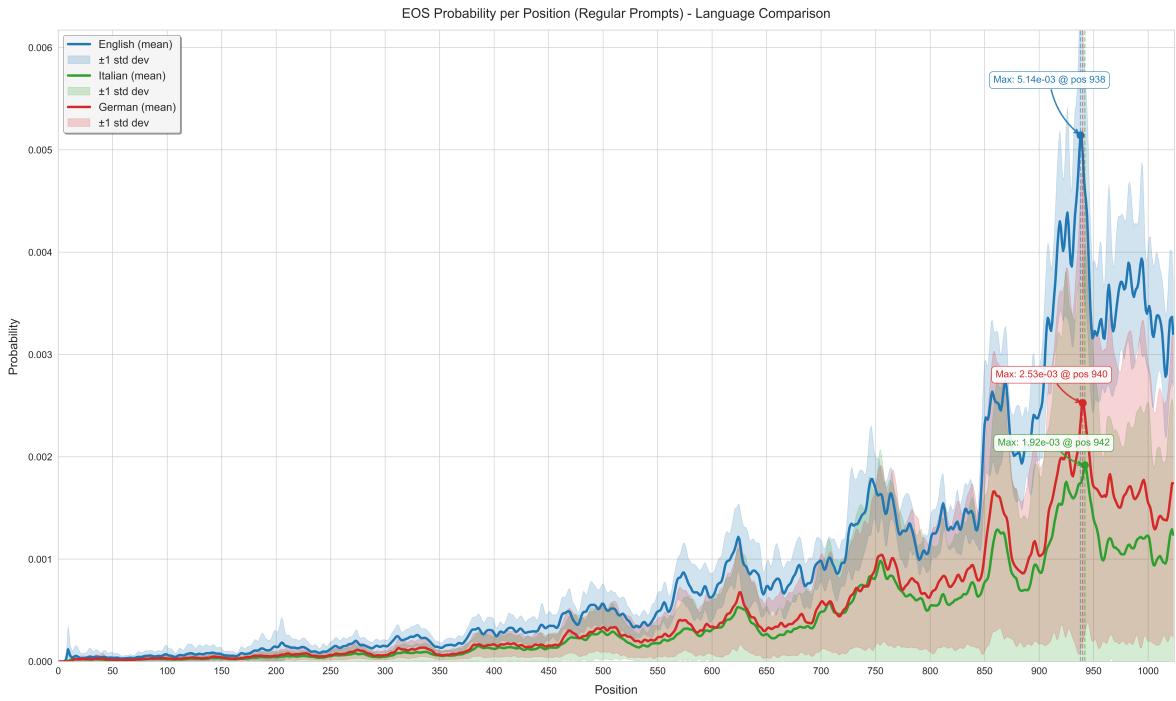


Figure 4: Regular Prompts Language Comparison. Comparison of EoS probability distributions across languages for regular prompts without trigger words. The hierarchical structure of EoS probabilities (English > German > Italian) is clearly visible, with all languages showing similar distribution shapes but different magnitude scales.

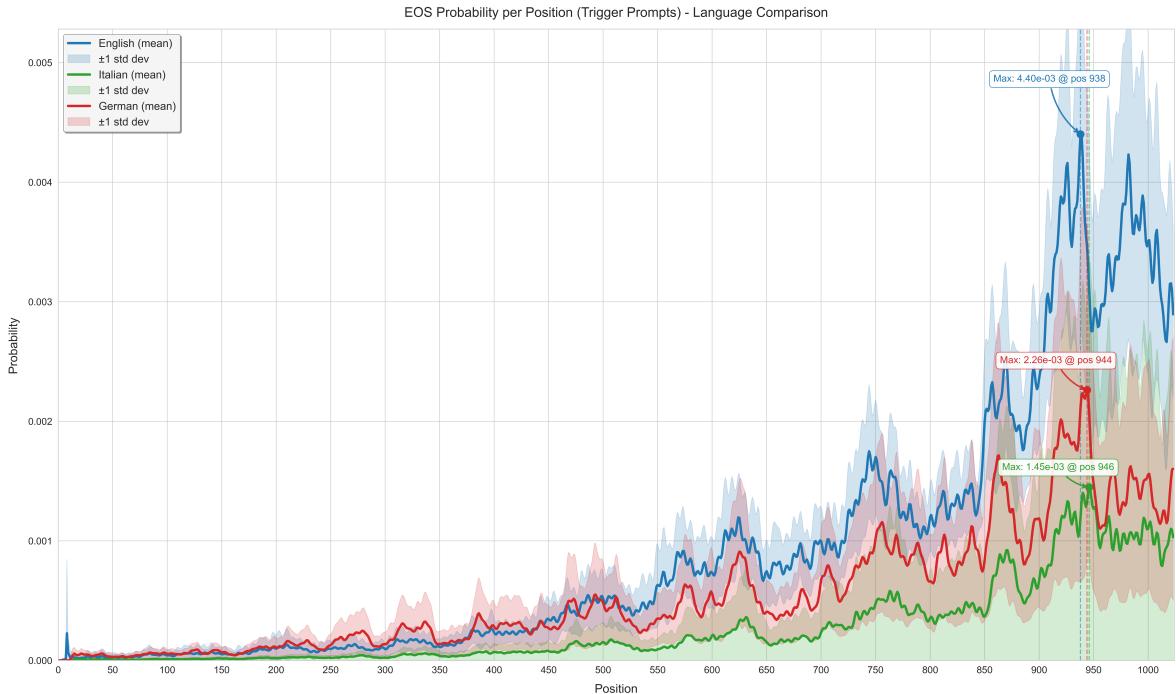


Figure 5: Trigger Prompts Language Comparison. Comparison of EoS probability distributions across languages for prompts containing explanation trigger words ("Explain", "Spiega", "Erkläre"). The same hierarchical structure of probabilities is maintained as in regular prompts, but with slightly different peak positions and reduced peak probabilities in non-English languages.

A.5 Examples of prompts and generated answers

We report examples taken from the tests conducted in Appendix A. The answers are obtained by predicting the length with the DistilBERT regressor and using the predicted length as generation length for LLaDa. Lengths are expressed in number of tokens.

- **Correct length** stands for the length of the answer in the original dataset (using the LLaDa tokenizer)
- **Predicted length** stands for the prediction of the DistilBERT regressor rounded up to the closest multiple of the block length, which is in this case 32.
- **Actual answer length** stands for the length of the final answer given by LLaDa, which is less than or equal to the predicted length.

A.6 Prompt–Answer Examples

Prompt: Who s the best team in League of Legends ?

Correct length: 154 Predicted length: 128 Actual answer length: 67

Answer: As an AI language model, I do not have personal opinions or preferences. However, the best team in League of Legends is subjective and depends on various factors such as team performance, player skill, and fan support. Some of the most successful teams in recent years include TSM, Cloud9, Fnatic, and Team Spirit.

Prompt: russia-ukraine crisis: why russia attcks in 2022?

Correct length: 329 Predicted length: 256 Actual answer length: 256

Answer: The Russia-Ukraine crisis in 2022 is a complex issue with multiple factors contributing to the conflict. Some of the main reasons for Russia's attacks on Ukraine include:

1. Historical tensions: Russia and Ukraine have a long history of conflict, dating back to the Soviet era. Russia has annexed Crimea in 2014 and has continued to support separatism in eastern Ukraine.
2. Economic concerns: Russia has expressed concerns about Ukraine's economic ties with the West, particularly the European Union. Russia has accused Ukraine of compromising its sovereignty and selling energy to the West.
3. Security concerns: Russia has accused Ukraine of being a NATO state and of allowing NATO bases on its territory. Russia has also accused Ukraine of being a terrorist state and of supporting terrorist groups.
4. Political concerns: Russia has accused Ukraine of being a democratic state and of having an authoritarian government. Russia has also accused Ukraine of being a corrupt state and of having rigged elections.
5. Environmental concerns: Russia has accused Ukraine of being a corrupt state and of having a poor environmental record.

Overall, the Russia-Ukraine crisis in 2022 is a complex issue with multiple factors contributing to the conflict.

Prompt: can coriander help with thallium poisoning?

Correct length: 97 Predicted length: 192 Actual answer length: 167

Answer: Coriander, also known as cilantro, is a herb that is commonly used in cooking and has been used in traditional medicine for various purposes. However, there is no scientific evidence to suggest that coriander has any therapeutic benefits for thallium poisoning.

Thallium is a highly toxic metal that can cause severe health problems if ingested or inhaled. Thallium poisoning typically requires immediate medical treatment, including activated charcoal, supportive care, and other interventions to manage the symptoms and prevent further damage.

If you suspect someone has been exposed to thallium, it is crucial to seek medical attention immediately. Thallium poisoning can be life-threatening, and prompt medical intervention is essential for a positive outcome.

In summary, while coriander is a valuable herb, it is not a reliable remedy for thallium poisoning.

Prompt: If someone says "The moon is not made of Coca-Cola", is it true or false?

Correct length: 39 Predicted length: 96 Actual answer length: 44

Answer: The statement "The moon is not made of Coca-Cola" is true. The moon is a natural satellite of the Earth, primarily composed of rock and metal, and it does not contain any trace of Coca-Cola.

Prompt: I have a shift in seconds from UTC, and I want to convert it into timezone (example +1): how can I do that in Python?

Correct length: 272 Predicted length: 352 Actual answer length: 351

Answer: To convert a shift in seconds from UTC into a timezone in Python, you can use the 'datetime' module along with the 'pytz' library. Here's a step-by-step guide on how to achieve this:

1. Import the necessary modules.
2. Define the shift in seconds from UTC.
3. Convert the shift to a timezone offset.

Here's an example:

```
““python
from datetime import datetime, timedelta
import pytz
```

```
# Define the shift in seconds from UTC
shift_seconds = 3600 Example: +1 hour
```

```
# Convert the shift to a timezone offset
timezone_offset = timedelta(seconds=shift_seconds)
```

```
# Print the timezone offset
print(timezone_offset)
““
```

In this example, 'timedelta(seconds=shift_seconds)' creates a 'timedelta' object representing the shift in seconds from UTC.

If you want to get the timezone offset in a more readable format, you can use the ‘pytz’ library to convert the ‘timedelta’ object to a string:

```
““python
from datetime import datetime, timedelta
import pytz

# Define the shift in seconds from UTC
shift_seconds = 3600 Example: +1 hour

# Convert the shift to a timezone offset
timezone_offset = timedelta(seconds=shift_seconds)

# Print the timezone offset in a readable format
print(pytz.timezone(timezone_offset))
““
```

This will output the timezone offset in a more readable format, such as ‘UTC+01:00’.

Prompt: What is belly inflation in Chinese Simplified?

Correct length: 50 Predicted length: 160 Actual answer length: 55

Answer: Belly inflation in Chinese Simplified is translated as " [OMITTED]" (fù shuǐ). This term refers to the accumulation of fluid in the abdomen, which can occur due to various medical conditions such as liver disease, heart failure, or other health issues.

Prompt: what is the fastest way to create a modern and secure web API?

Correct length: 255 Predicted length: 320 Actual answer length: 319

Answer: There are several ways to create a modern and secure web API, but one of the fastest and most efficient ways is to use a popular web framework like Django or Flask. These frameworks have pre-built security features and libraries that can help you create a secure API quickly.

Here are some steps you can follow to create a modern and secure web API using a web framework:

1. Choose a web framework that suits your needs. Django and Flask are both popular choices, and they have extensive documentation and support.
2. Set up your development environment. Install the web framework and any necessary dependencies.
3. Define your API endpoints and routes. Use the web framework’s built-in routing system to define your API endpoints and routes.
4. Implement authentication and authorization. Use the web framework’s built-in authentication and authorization system to ensure that only authorized users can access your API.
5. Implement encryption. Use the web framework’s built-in encryption system to encrypt sensitive data at rest and in transit.
6. Implement input validation. Use the web framework’s built-in input validation system to ensure that all input is properly validated and sanitized.
7. Implement error handling. Use the web framework’s built-in error handling system to handle errors and exceptions gracefully.
8. Test your API. Use the web framework’s built-in testing tools to test your API and ensure that it is secure and reliable.

By following these steps and using a web framework, you can create a modern and secure web API quickly and efficiently.

Prompt: are you able to summarize conversations?

Correct length: 29 Predicted length: 96 Actual answer length: 39

Answer: Yes, I can help summarize conversations. If you’d like me to summarize a specific conversation, please provide the details, and I’ll do my best to provide a concise and accurate summary.