A* decoding - Fast, precise and diverse decoding for LLMs Samsung R&D Institute Poland participation in WMT2025

Team Name: SRPOL

Adam Dobrowolski, Paweł Przewłocki, Dawid Siwicki
Samsung R&D Institute, Warsaw, Poland
{a.dobrowols2@samsung.com, p.przewlocki@partner.samsung.com, d.siwicki@samsung.com}

Abstract

This work presents an innovative decoding approach utilizing the A* algorithm, which generates a diverse and precise set of translation hypotheses. Subsequent reranking through the Noisy Channel Model Reranking and Quality Estimation selects the best among these diverse hypotheses, leading to a significant improvement in translation quality. This approach achieves up to a 0.5-point reduction in the MetricX-24 score and a 1.5-point increase in the COMET score.

The A* decoding algorithm is model-agnostic and could be applied to decoding in LLMs as well as classic transformer architectures. The experiment shows that by using freely available open source MT models, it is possible to achieve translation quality comparable to the best online translators and LLMs using a single 32GB GPU card.

1 Introduction

The final stage of inference in language models is decoding, which involves generating text based on calculated token probabilities. The most commonly used approach is autoregressive decoding, which generates tokens sequentially based on the source text and the text produced thus far. There are three primary strategies for autoregressive decoding: greedy, beam, and sampling. Each of them has its flaws that A* resolves.

2 Decoding Strategies

Greedy Decoding selects the most probable token at each step, prioritizing speed and simplicity but often producing suboptimal results due to its limited exploration of alternative paths.

Sampling introduces randomness by selecting tokens based on their probability distribution, but the generated hypotheses can be overly random, resulting in translations that are not always optimal.

Beam Search maintains multiple high-probability sequences (beams) simultaneously, enhancing output quality at a modest computational cost proportional to the beam width. This algorithm has been widely used in machine translation.

Recent advances in large language models (LLMs) have demonstrated their superiority over traditional encoder-decoder transformers (Vaswani et al., 2017). However, because of their primary purpose, LLMs are designed for rapid text generation and not for quality-oriented beam search. Beam search in LLMs takes much more time than in classic transformers. For example, from our experiments, the beam search using vLLM[1] on EuroLLM-9B (Martins et al., 2025) takes about 30 seconds to decode with beam of width 10 for just one sentence. Within the same time, we can use sampling to generate as many as 800 diverse hypotheses, which later assessed by QE give great improvement over greedy decoding or beam search. In the following, we propose a novel A* decoding algorithm for LLMs that combines the speed of beam search in classic transformers with the diversity of sampling.

3 A* Decoding

Let's consider decoding as the process of searching for the optimal path within a tree of all possible sentences generated by a model, where the chosen path represents the best translation of the source sentence. Typically, there are no clear criteria for evaluating the quality of the generated sequence, so we must rely on proxy metrics derived from available data to assess its quality. One of the most effective and straightforward proxy metrics is the average probability of all tokens generated. This metric guides the algorithm to select the most probable token at each step of the output generation

[1] https://docs.vllm.ai/

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 219–223

```
def a_star_decoding(source, max_cands, hope_level):
   queue = ReversedPriorityQueue()
   queue.put((0, []))
   results = []
   for iteration in range(max_cands):
      if queue.empty(): break
      old_f_score, prefix = queue.pop()

      new_trn = LLM.generate(source, prefix)
      results.append(new_trn)
      if iteration == 0:
         def_f_score  = sum(new_trn.tokens) / len(new_trn.tokens)

      for idx, token in enumerate(new_trn.tokens[len(prefix):]):
         idx += len(prefix)
         for alt_token in token.alternative_tokens:
            g_score = sum(new_trn.tokens[:idx]) + alt_token.logprob
            h_score = sum(new_trn.tokens[idx+1:])
            f_score = (g_score + h_score) / len(new_trn.tokens) + hope_level
            if f_score > def_f_score:
               queue.put((f_score, new_trn.tokens[:idx] + [alt_token]))
   return results
```

Listing 1: Python-like pseudocode of the A* decoding

process. The simplest approach, known as greedy decoding, follows this principle. However, the results of greedy decoding may be suboptimal, as it overlooks many potential paths. In contrast, A* decoding allows the algorithm to explore beyond just the most probable next token, selecting tokens that offer the potential for discovering a more probable path by the end of the process. A similar concept has already been explored for Statistical Machine Translation (SMT) with long paragraphs (Och et al., 2001).

3.1 A* algorithm

A* algorithm uses priority queue to explore graph nodes, prioritizing those with the lowest estimated total cost.

For each node n, it computes:

• $g(n)$: exact cost from the start node to n.

• $h(n)$: heuristic estimate of the cost from n to the end of sentence.

• $f(n) = g(n) + h(n)$: estimated total cost of the path through n.

3.2 A* decoding description

We propose the following adaptation of the A* algorithm to decoding in LLMs. The probability of the remaining sequence is the probability of the generated token (as in beam-search) plus probability of the sequence following the new token. Assuming that the remaining sequence could have a slightly higher probability than the base sequence, we can estimate the total probability of a new path. This is done by adding the probability of the new token, adjusted by a small constant to account for the potential increase of total probability.

The initial step of the algorithm is generation of the default greedy translation for the source sentence. We then enhance this baseline by employing the A* algorithm for further exploration. For each token generated, we select alternative tokens at its position and calculate the estimated total cost, f(n), as outlined below:

• $g(n)$ (actual cost) - normalized logarithm probability of a prefix with alternative token.

• $h(n)$ (estimate cost) - normalized logarithm probability of the default suffix plus some constant "hope_level" that assumes that the actual probability of the rest of sentence may be bigger than the default. The "hope_level" should be chosen experimentally. High enough to ensure that $h(n)$ remains admissible (does not overestimate the actual cost), but not too high to avoid considering highly improbable alternative tokens.

The pseudocode for the algorithm is presented in Listing 1.

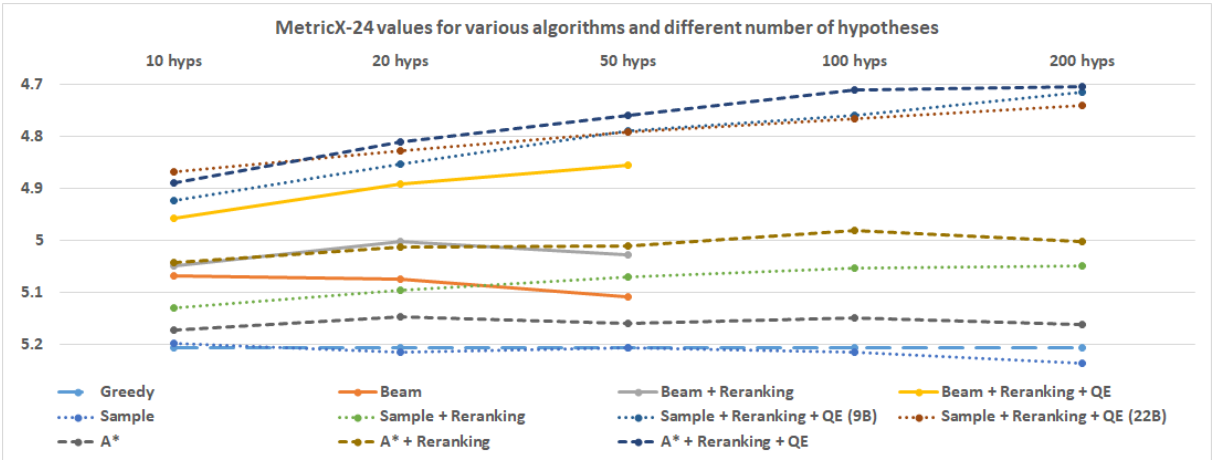

Figure 1: Comparison of A* with other algorithms for the English-to-Czech direction.

Method	10 hyps	20 hyps	50 hyps	100 hyps	200 hyps	time/200hyps
Greedy	5.2064	5.2064	5.2064	5.2064	5.2064	0.031
Beam	5.0677	5.0740	5.1088			3.330
Beam + Reranking	5.0492	5.0021	5.0279			3.330
Beam + Reranking + QE	4.9586	4.8920	4.8555			3.330
Sample	5.1992	5.2149	5.2077	5.2146	5.2363	0.040
Sample + Reranking	5.1303	5.0962	5.0709	5.0539	5.0493	0.042
Sample + Reranking + QE (9B)	4.9233	4.8545	4.7894	4.7604	4.7151	0.047
Sample + Reranking + QE (22B)	**4.8678**	4.8288	4.7919	4.7660	4.7410	0.047
A*	5.1722	5.1468	5.1590	5.1494	5.1612	0.024
A* + Reranking	5.0439	5.0132	5.0111	4.9811	5.0028	0.025
A* + Reranking + QE	4.8889	**4.8106**	**4.7601**	**4.7110**	**4.7049**	0.029

Table 1: Comparison of MetricX-24 scores on the WMT24++ test set for the English-to-Czech translation direction.

4 Reranking

Following the generation of a set of translation hypotheses using the above described algorithm, we select the optimal hypothesis through a modified Noisy Channel Model Reranking approach, as described by (Yee et al., 2019). This reranking method enhances machine translation by reordering candidate translations during the decoding process. It integrates three components: a direct translation model, $P(T|S)$, which predicts the target sentence T given the source sentence S; a channel model, $P(S|T)$, which assesses the likelihood of the source sentence given the target; and a language model, $P(T)$, which evaluates the fluency of the target sentence. The algorithm ranks candidates by computing a weighted combination of these probabilities, ensuring the selection of the most accurate and fluent translation. The algorithm scores candidates using a weighted combination of these probabilities:

• $S(T|S)$=$P(T|S)$+$\lambda_1$log $P(S|T)$+$\lambda_2$log $P(T)$

where λ_1 and λ_2 are tunable weights. Top candidates from the direct model are reranked based on this score, prioritizing translations that balance fidelity to the source and fluency in the target. For our submission in WMT2025 we applied the following scoring:

• $P(T|S)$ - calculated by weighted sum of probabilities of EuroLLM-9B, NLLB-3.3 (Team et al., 2022) combined with Unbabel/comet wmt23-cometkiwi-da-xl score. (Rei et al., 2020)

• $P(S|T)$ - calculated by NLLB-3.3.

• $P(T)$ - not used - for compliance with constrained path.

4.1 Hallucination detection

Before reranking, we remove hypotheses that appear to be outliers or hallucinations. For each source sentence, we calculate the standard deviation of the probability scores across all hypotheses for each scoring model. Hypotheses with scores below the mean minus one standard deviation are filtered out.

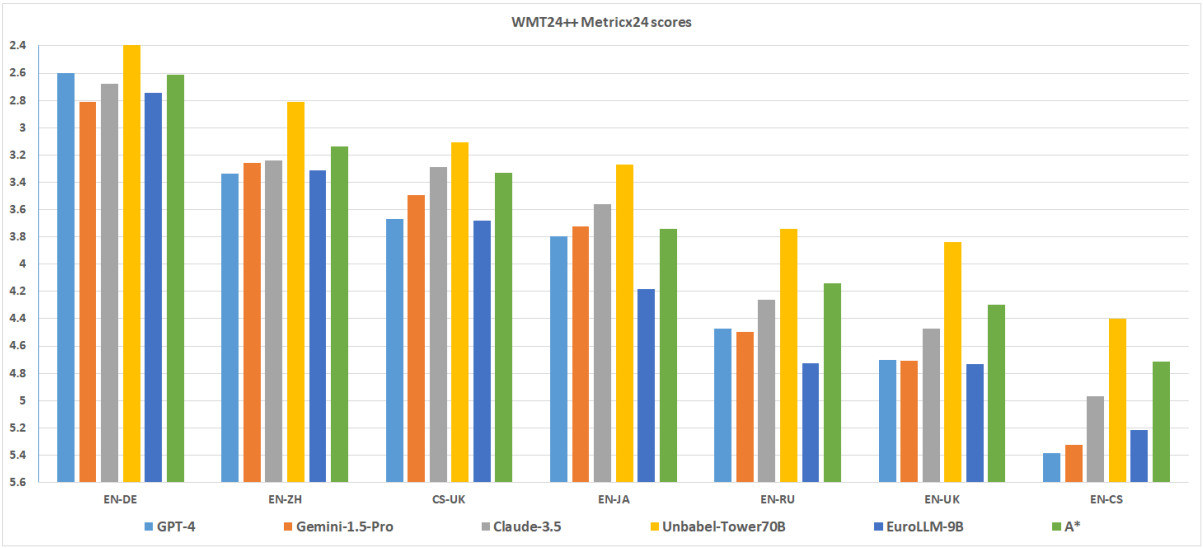

Figure 2: Comparison of presented solution (green) to leading LLMs.

5 Results

5.1 Comparison with different decodings

The chart on Figure 1 presents the quality of various algorithms across different number of generted hypotheses. The Y-axis represents reference-based MetricX-24 scores (Juraska et al., 2024) on wmt24++ testset (Deutsch et al., 2025) for the English-to-Czech translation direction. The X-axis indicates the number of hypotheses generated per source sentence. Table 1 presents exact values for the chart. The values for sampling vary with each run. The values in the table represent the mean of multiple runs. Performance times were measured in seconds on a single A100-80GB GPU using the vLLM framework on EuroLLM-9B. 80GB of GPU memory is not necessary. We have successfully run the above tests on A100-40GB and V100-32GB with slower decoding time, but the same quality.

The baseline performance, established through greedy decoding, achieves a score of 5.2064. Pure beam search, utilizing a beam width of only 10 hypotheses, yields a better score of 5.0667, though it requires approximately 30 seconds to decode a single sentence. Despite its computational complexity, beam search remains the best algorithm for decoding without reranking. The introduction of reranking shifts the advantage to the A* algorithm, which delivers an improved score of 5.0062, representing a 0.2-point improvement over the baseline. Incorporating Quality Estimation (QE) further enhances performance, boosting results by approximately 0.2 to 0.3 points above the reranked results. In this scenario, A* remains the top-performing algorithm, achieving the best score of 4.7049. When all techniques are combined, the overall improvement ranges from approximately 0.2 to 0.5 points on the reference-based MetricX-24 compared to the baseline.

It may seem surprising that A* decoding requires less time than sampling for a single source sentence, but this number comes from total time divided by maximum number of hypotheses - 200. Sampling always generates a fixed number of hypotheses, often including duplicates. In contrast, A* decoding halts once it can no longer identify distinct hypotheses, especially for short segments, resulting in a shorter total processing time compared to sampling. E.g. for beam of 200 A* decoding on WMT24++ generates only about 100 hypotheses on average.

5.2 Comparison with other LLMs

Figure 2 presents automatic scores for translations for different directions, generated using several leading LLMs: Claude[2], Gemini[3], GPT[4], Unbabel-Tower (Alves et al., 2024). Values are reference-based scores of MetricX-24. While these scores do not perfectly reflect translation quality, they offer a general indication of performance. The table illustrates that the translation quality achieved by the method described in this paper is comparable to that of the leading LLMs.

[2]https://claude.ai/
[3]https://gemini.google.com/
[4]https://chatgpt.com/

6 Conclusions and future work

We introduced a novel, high-speed decoding algorithm (A*) that generates hypotheses significantly faster than beam search in large language models (LLMs). This algorithm is adaptable to any language model. The application of Noisy Channel Reranking enhances the quality of diverse generated candidates by up to 0.2 points on the MetricX-24 scale. Further application of Quality Estimation (QE) reranking yields an additional improvement of another 0.2 to 0.3 points.

A* decoding algorithm flexibility enables a balance between quality and speed. Due to its efficiency and adaptability for any language model, this method has potential for numerous practical applications.

The proposed solution demonstrates high quality of EuroLLM models. The final results reflect a significant improvement over EuroLLM-9B translations, with a reduction of up to 0.5 points in MetricX-24 and an increase of 1.5 points in COMET, achieving quality comparable to leading large language models, even for non-European languages.

Due to time constraints prior to the WMT25 workshop, we were unable to explore post-processing techniques or context-aware multi-sentence decoding. The results submitted represent the unrefined output of the method described in this paper. A* decoding is a new idea, and we intend to refine it through further research. The results presented at WMT25 could be enhanced by leveraging other LLMs models as a base model, incorporating alternative Quality Estimation methods, or utilizing improved language models for reranking.

7 Acknowledgements

We would like to express our gratitude to the entire Machine Translation team at Samsung R&D Poland for their support. Special thanks go to Marcin Szymański for valuable reviews. We also acknowledge the management of Samsung Poland for their support in preparing this work.

References

Duarte M. Alves, José Pombal, Nuno M. Guerreiro, Pedro H. Martins, João Alves, Amin Farajian, Ben Peters, Ricardo Rei, Patrick Fernandes, Sweta Agrawal, Pierre Colombo, José G. C. de Souza, and André F. T. Martins. 2024. Tower: An open multilingual large language model for translation-related tasks.

Daniel Deutsch, Eleftheria Briakou, Isaac Caswell, Mara Finkelstein, Rebecca Galor, Juraj Juraska, Geza Kovacs, Alison Lui, Ricardo Rei, Jason Riesa, Shruti Rijhwani, Parker Riley, Elizabeth Salesky, Firas Trabelsi, Stephanie Winkler, Biao Zhang, and Markus Freitag. 2025. Wmt24++: Expanding the language coverage of wmt24 to 55 languages dialects.

Juraj Juraska, Daniel Deutsch, Mara Finkelstein, and Markus Freitag. 2024. MetricX-24: The Google submission to the WMT 2024 metrics shared task. In *Proceedings of the Ninth Conference on Machine Translation*, pages 492–504, Miami, Florida, USA. Association for Computational Linguistics.

Pedro Henrique Martins, João Alves, Patrick Fernandes, Nuno M. Guerreiro, Ricardo Rei, Amin Farajian, Mateusz Klimaszewski, Duarte M. Alves, José Pombal, Nicolas Boizard, Manuel Faysse, Pierre Colombo, François Yvon, Barry Haddow, José G. C. de Souza, Alexandra Birch, and André F. T. Martins. 2025. Eurollm-9b: Technical report.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Kyra Yee, Nathan Ng, Yann N. Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation.