# A Token-Level Decoding Algorithm for Ensembling Models Across Vocabulary Sizes

**Anonymous ACL submission**

## Abstract

Model ensembling is a technique to combine the predicted distributions of two or more models, often leading to improved robustness and performance. For ensembling in text generation, the next token's probability distribution is derived from a weighted sum of the distributions of each individual model. This requires the underlying models to share the same subword vocabulary, limiting the applicability of ensembling, as many open-sourced models have distinct vocabularies. This paper proposes an inference-time-only algorithm for ensembling models with different vocabularies without the need to learn additional parameters or alter the underlying models. Instead, the algorithm ensures that tokens generated by the ensembled models *agree* in their surface form. We apply this technique to combinations of traditional encoder-decoder models and decoder-only LLMs and evaluate on machine translation. In addition to expanding to model pairs previously incapable of token-level ensembling, our algorithm frequently improves translation performance over either model individually.

## 1 Introduction

Text generation takes place as a sequence of token predictions. At each time step, the model, conditioned on some input, produces a probability distribution over the vocabulary. From this distribution, the next token is selected to extend the hypothesis—the text generated thus far. Individual models may be sensitive to noise or lack coverage in certain domains. Model ensembling is a method for combining outputs from multiple models, which often produces more robust outputs and increases performance. The traditional model ensembling approach assumes a shared vocabulary and computes a new distribution as a weighted sum of its component vocabularies:

$$p(x_t) = \sum_i \lambda_i p_{m_i}(x_t \mid x_{1..t-1}) \quad (1)$$

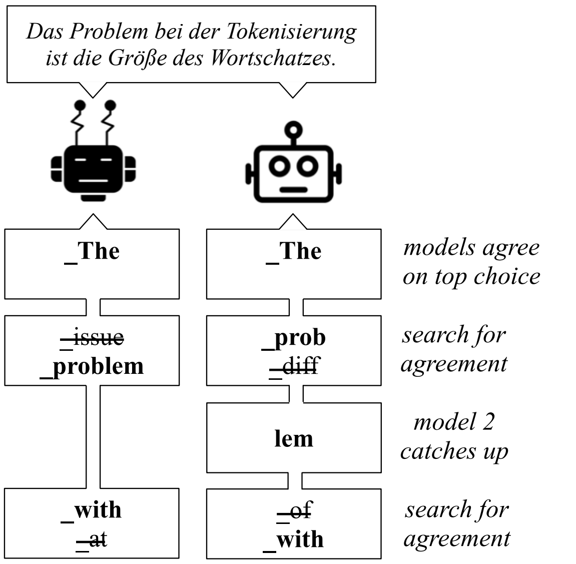



Figure 1: Agreement-Based Ensembling (ABE) enables ensembling among models with different vocabularies. Token generation for each beam item is constrained to tokens with agreeing detokenized forms.

where all interpolation weights, $\lambda_i$, sum to 1. The new ensembled distribution functions the same as if it originated from a single model, and the next token prediction proceeds as usual.

In practice, most models do *not* share vocabularies. When the vocabularies differ, the resulting probability distributions are no longer comparable. Then, it is no longer straightforward to ensemble these outputs. To address this, we introduce Agreement-Based Ensembling (ABE), an inference-time ensembling algorithm that requires no new parameters or model adaptation, but instead works by coordinating token selection across models under the notion of *agreement* (§ 3.1). At each decoding timestep, each model produces its distribution over the next token; our method efficiently searches over their cross-product for tokens that are contextually compatible with the currently generated surface string (§ 3.2). When the tokens are different (but agreeing), the longer token constrains future search (§ 3.3). This is caricatured in Figure 1). Our approach easily extends to other inference algorithms such as beam search (§ 3.4).

Our contributions are as follows:

- introduce an inference-time algorithm for ensembling models with different vocabularies[1],
- demonstrate the ability to ensemble across varying architectures (encoder-decoder, LLMs, or both), and
- show improved results in machine translation (MT) across a range of models.

Our code is implemented in Python using the Huggingface transformers library (Wolf et al., 2019) and is open-source.[2]

## 2 Related Work

Ensembling is a generally reliable technique for increasing the quality of model outputs that goes back at least as far as Hansen and Salamon (1990). Although it is more expensive and, therefore, often prohibitive in production inference settings, it is useful in competitions or for production training scenarios, such as distillation. In such settings, the user typically has control over model training; ensembled models can be taken from different checkpoints (Sennrich et al., 2016) or from completely different training runs initialized from different random checkpoints, and therefore all have the same vocabularies. Hoang et al. (2024) move a step beyond this by ensembling models with divergent architectures (an MT system and an LLM) and across contexts longer than are supported by all models, but the models still share the same vocabulary.

The situation becomes more difficult when the vocabularies are not shared. One way to address this is to work at the sequence level instead of the token level. One such approach is that of Jiang et al. (2023), who propose LLM-Blender. It comprises a ranking function that computes pairwise comparisons of complete model outputs and then selects from among them; this approach completely avoids the need for token-level ensembling. Farinhas et al. (2023) generate multiple translation hypotheses and then explore selecting from among them using voting, minimum Bayes risk, and LLM-based selection. Sequence-level ensembling has limitations, and the reality of disjoint vocabularies has motivated prior work in token-level ensembling even across different vocabularies. Xu et al. (2024) learned mappings across vocabularies through extra model training that map token representations into a joint space, and employs various filtering methods for efficiency. Shen et al. (2024) presents a "collaborative decoding" framework between a lead and assistant model where a classifier dynamically selects which will produce the next token at each generation step; their approach also appears to require a shared vocabulary.

Union-based ensembling methods combine models with different vocabulary by constructing a shared token space. GAC by Yu et al. (2024) creates a complete union of vocabularies of individual models and averages token probabilities. Still, it assumes high overlap between tokenizers, leading to misalignment issues when semantically similar tokens (e.g., 'James' vs. 'Jam' + 'es') are split differently. Top-k UNITE (Yao et al., 2024) builds on this by forming a union over each model's top-k tokens and augmenting the list to include full or partial tokens as needed, mitigating tokenizer mismatch. However, it requires a designated primary model and retains dependence on top-k filtering. DeepEn (Huang et al., 2024) takes a different approach by mapping each model's output distribution into a shared relative representation space, constructed using common tokens across models, and inverse mapping, via gradient descent, back to the primary model's token space to select the next token. Our work is distinct from prior methods as it does not require further training or parameters, does not rely on vocabulary overlap or a primary model, and is evaluated on an open-ended generation (MT) task rather than classification benchmarks. We achieve this by managing token-level ensembling across different vocabularies by ensuring that all models in the ensemble agree on the *string* being generated, and interleaving model steps for models that fall behind.

## 3 Agreement-Based Ensembling

Autoregressive models produce distributions over their vocabularies at each decoding time step. This process generally continues until the end-of-sequence token is produced or some maximum length is reached. Greedy decoding, beam

[1] Our sole requirement is that models are open-vocabulary so that they can generate any string the other model can.

[2] Outputs and code available anonymously at https://anonymous.4open.science/r/anon-abe-073B. It will be released as Apache 2.0.

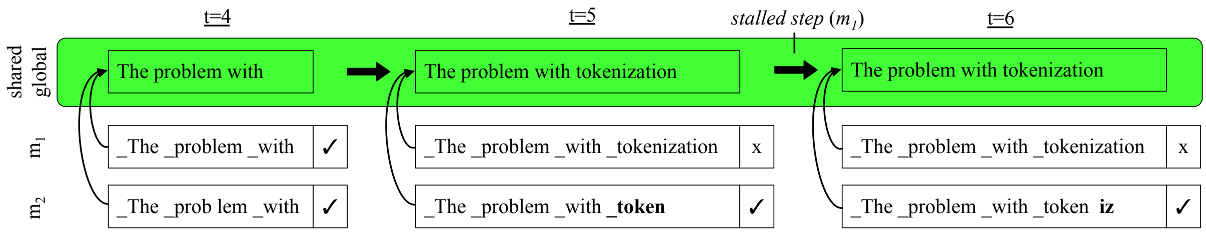



Figure 2: A global state maintains the shared detokenized string, which is determined by the local hypotheses. A flag denotes whether a model is stalled (×) or can generate (✓). In stalled steps (§ 3.3), only the trailing model(s) generate(s) a token, catching up with the shared string, and the stalled model is prevented from generating content.

search, and sampling are all search algorithms that change how the next token is selected. The traditional model ensembling approach (also called here *interpolation-based ensembling*) fits nicely within any of these frameworks, but requires the models to share the same vocabulary. This approach alters the probability distribution to be a weighted sum of the distributions from each model. Any search algorithm proceeds as before, selecting a token from this new distribution.

When vocabularies differ, the distributions do not match, and we cannot factor in the probability computation from the algorithm as nicely. In Agreement-Based Ensembling, each model produces its distribution over its target vocabulary as usual, but algorithmic changes are required to coordinate the selection of the next token to ensure they agree on the detokenized surface string.

In this section, we will describe these changes. At a high level, this requires maintaining a shared global agreement state (§ 3.1), efficiently searching the cross-product of the models' vocabularies (§ 3.2), and handling the varying token lengths of the models' differing vocabularies (§ 3.3). For ease of presentation, we will describe the algorithm using two models in a greedy decoding setting; this allows us to focus on these new ideas, without the complexity of beam search. However, the algorithm works with any number of models as long as they all have open vocabularies, and the extensions to beam search (which we used for all our experiments) are straightforward.

### 3.1 Agreement

The fundamental difficulty when ensembling models with different vocabularies is ensuring that they reach consensus on the shared output string, despite the string being generated via different tokenizations. In Agreement-Based Ensembling, we maintain a shared string—the global hypothesis—which is updated at each time step by the predicted tokens. It is vital to store and compare against this string in *detokenized* form[3] for precise comparison. Each model separately maintains its own local hypothesis under its own tokenization, which is a substring of this global hypothesis. This is visualized in Figure 2.

We define the notion of *agreement*. Consider a set of strings $S$. The global hypothesis, $g$, of this set is defined by (1) the shortest terminated string (ends with end-of-sequence token) or (2) the longest unterminated sequence—whichever is satisfied first. A set of strings $S$ is in agreement $\Leftrightarrow$ all $s_i \in S$ are substrings of $g$. Note that *agreement* does not mean the models have produced the same string, only that their strings do not *disagree*. The algorithm provides a core inductive guarantee that the detokenized string for every model will always agree with the shared global hypothesis.

### 3.2 Efficient Search

At each decoding timestep, each model takes its forward step from its current state and produces a distribution over its vocabulary. We need to efficiently search the intersection of their vocabularies for extensions to the current shared hypothesis that agree. This space has dimensions $V_1 \times V_2$ and is too large to search completely. We therefore apply a variant of cube pruning (Chiang, 2007; Huang and Chiang, 2007) with an "agreement filter" to search this space efficiently. The distributions from each model are sorted, per usual, and arranged into a two-dimensional grid (Refer to Figure 3).

Each box in the grid denotes the selection of a token from each vocabulary, each associated with a score, computed as the weighted sum of the length-normalized model scores for each local hypothesis.[4] Normalization is essential as model hypotheses are not guaranteed to be the same length.

[3] We store byte-strings so byte fall-back tokenization and non-Latin scripts to work.

[4] In all experiments, models are equally weighted.

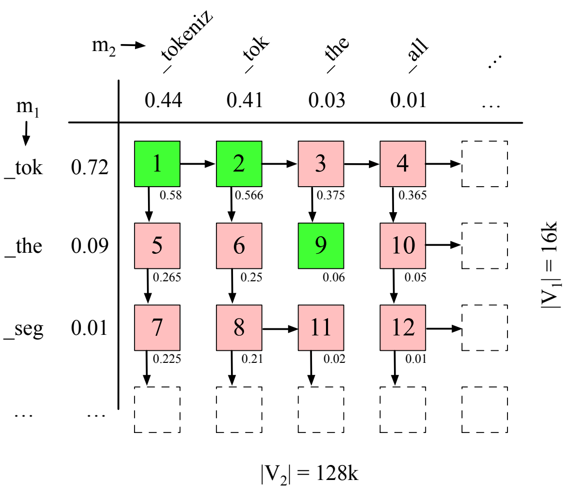



Figure 3: The first 12 candidates in the ABE search space for unstalled $m_1$, $m_2$. Each model's vocabulary is sorted by score. The top left corner is pushed onto a heap with a weighted score of 0.58. We present probabilities here for simplicity. In practice, each token score is the cumulative log prob of the local hypothesis with this token as the extension. The loop then pops from the heap, checks for agreement, and adds unvisited neighbors onto the heap. Numbers denote visitation order.

To enumerate these items, we maintain a heap, which stores tuple items $(i, j, s)$, where $i$ and $j$ index the candidate vocabulary items, and $s$ records their weighted score. The heap is seeded with the tuple $(1, 1, s)$, which denotes the top left corner of this grid and represents the most probable token extension from each model. We now iterate as follows:

**while** True **do**
    Pop *item* from *heap*
    Compute strings $s_1$ and $s_2$
    **if** agrees($s_1, s_2$) **then**
        return *item*
    **end if**
    Add *neighbors* of *item* to *heap*
**end while**

Although we need only one valid item for our greedy search example, Figure 3 depicts the first twelve loop iterations for illustrative purposes. The current item is popped from the heap at each step and checked for agreement. This item is checked to determine whether the set of proposed local hypotheses is in agreement. Arrows denote "neighbor" items (the next vocabulary extension in each dimension), which are used to create updated tuples that are then added to the heap. The algorithm can be extended to an arbitrary number of ensembled models using an $n$-dimensional hypercube,[5] and extending the tuples to include $n$ vocabulary position indices.

### 3.3 Stalled steps

Models with larger vocabularies are likely to generate longer subwords at each timestep. This means that one model may be ahead of the rest and needs to be *stalled*. We define *stalling*. Consider a set of models, $M$. The set of local hypotheses generated by $M$ is $S$, where $s_i$ was generated by $m_i$. Recall that the global hypothesis is represented by $g$. A model, $m_i$, is stalled when $s_i = g$ and at least one other model is not stalled: $\exists (m_j, s_j) \ni s_j \neq g$. An example of when a model becomes stalled is illustrated in time steps $t = 5$ and $t = 6$ in Figure 2.

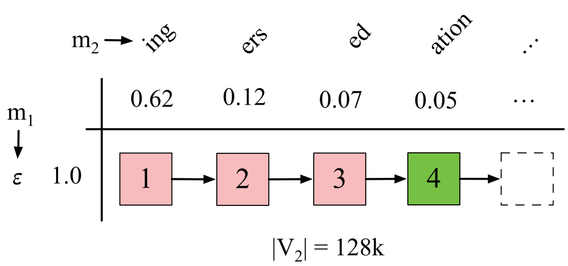



Figure 4: Search space when $m_1$ is stalled. $m_1$ has generated *tokenization* while $m_2$ has only generated *_token iz*. Similar to Figure 3, we present probabilities instead of the cumulative log-prob of the local hypothesis with this token as the extension for simplicity.

Stalled steps aim to restore this imbalance by allowing the unstalled models to generate without the stalled models in order to catch up. Conceptually, stalling a model is simple. We prevent the model from being able to generate a token by replacing $V$ (its vocabulary) with $\{\epsilon\}$—an empty transition. We illustrate the reduction in search space in Figure 4. Note that for each stalled model, the dimensionality of the search space is effectively reduced by one.

### 3.4 Beam Search

Greedy decoding is a special case of beam search where the beam size is 1. Extending ABE to handle larger beams is simple. The main conceptual difference is that the search space includes an additional dimension, the beam index. For a beam size of $k$, the search space is $k \times V_1 \times V_2$. Similar to the extension beyond two models (end of Section 3.2), we add an index to denote which beam item each

[5]For simplicity, we use the term hypercube, though not all dimensions are equal.

vocabulary pair comes from. Then, instead of terminating after the first valid item, we iterate until we have encountered $k$ of them. For instance, three models with a beam would have a 4-dimensional search space of $\{k \times V_1 \times V_2 \times V_3\}$. The $k$ items become the beam at the next time step. Note that *neighbors* of a given candidate must come from the same beam item; beam number 2 cannot have neighbors in beam number 3. Beam lengths may be ragged due to stalling, but this is handled with padding, normalization, and selecting hidden states based on hypothesis length.

## 4 Experiments

Agreement-Based Ensembling constrains the output of each model by the output of all models. We therefore choose to evaluate against machine translation (MT) due to its constrained nature. We primarily evaluate on the WMT24 test set (Kocmi et al., 2024) `en-de` but extend to several other out-of-English directions (`cs`, `es`, `uk`) from the same test set. For evaluation, we consider both COMET (Rei et al., 2022) and BLEU (Papineni et al., 2002). We computed COMET scores with with pymarian[6] (Gowda et al., 2024), and BLEU scores with sacrebleu[7] (Post, 2018).

We examine ensembling within and between different classes of models: 1) **Custom MT**– Our own Encoder-decoder models trained on the same pool of data with different vocabulary sizes, 2) **Public MT**– Large-scale, multilingual, publicly-available MT models, and 3) **LLMs**– Decoder-only LLMs with demonstrated capabilities in MT.

### 4.1 Models

For preliminary experiments, we ensemble our own trained MT models. This allows us to control the vocabulary while also guaranteeing that the models are reasonably similar and will frequently agree during generation. We then extend to off-the-shelf models, covering both encoder-decoder and decoder-only architectures.

**Custom MT** We train transformer base models using Marian (Junczys-Dowmunt et al., 2018) on approximately 600m lines of filtered English–German (en-de) data downloaded using `mtdata` (Gowda et al., 2021) (details in Appendix A). We perform standard data filtering to include deduplication, language identification, length ratios, and margin-scoring. We train four unigram-based `sentencepiece` tokenization models (Kudo, 2018; Kudo and Richardson, 2018) with sizes of 8k, 16k, 32k, and 64k. Using these four tokenizers, we train four associated machine translation models.

Each model is a standard transformer base model (Vaswani et al., 2023) with 6/6 layers, embedding size 1024, and hidden sizes of 8192. The entire configuration can be found in Table 5 in the Appendix. The data is randomly shuffled for infinite streaming via `sotastream` (Post et al., 2023), so we use logical epochs (1b tokens) rather than exact passes over the training set. We train for 25 logical epochs on one 24GB Titan RTX. In our experiments, we use various checkpoints of these models.[8]

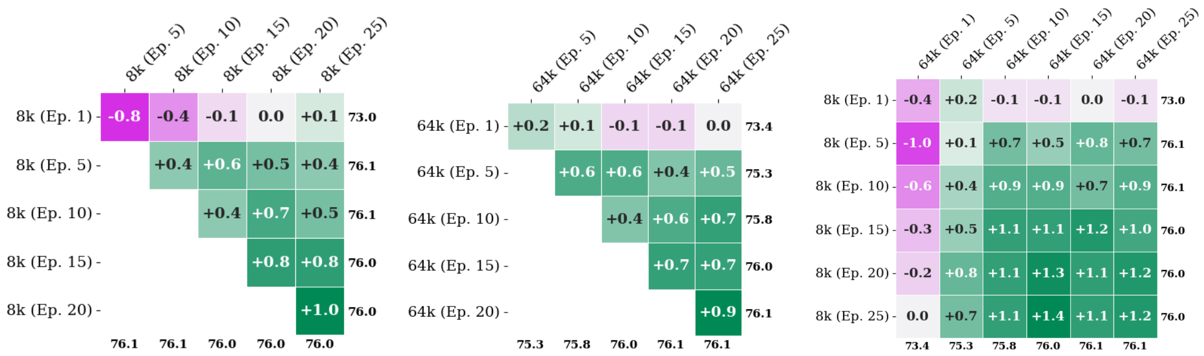
**Public MT** In addition to custom models that only support English and German, we also consider two widely used multilingual MT models, M2M (Fan et al., 2020) and NLLB (Team et al., 2022) in multiple sizes and distillation variants. The former covers 100 languages with a 128k multilingual vocabulary, while the latter covers 202 languages with a 256k multilingual vocabulary. The Huggingface repository IDs for all off-the-shelf models are in the Appendix in Table 6.

**LLMs** We consider TOWER (Alves et al., 2024) and LLaMa 3.x (Grattafiori et al., 2024). TOWER is an LLM specifically fine-tuned for the translation task, whereas LLaMa is general-purpose. LLaMa models use a vocab of 128k while TOWER uses 32k. TOWER was fine-tuned with the following prompt:

> **Translate the following text from English into German. \n English: {source sentence} \n German:**

For LLaMa models, we use both 0-shot prompts and 3-shot prompts derived from the WMT24 baseline evaluation scripts.[9] The exact verbiage of prompts is in the Appendix in Table 7. LLMs differ in architecture from the previous settings as they lack an encoder, demonstrating that ABE is architecture-agnostic. Ensembling two large LLMs with 3-shot prompts requires an additional memory footprint. These experiments were run on a single 80GB A100, though they can be managed with approximately 48 GB.

[6] Version v1.12.31, wmt22-comet-da model
[7] Version 2.5.1, standard params.
[8] Namely epochs {1, 5, 10, 15, 20, 25}
[9] https://github.com/wmt-conference/wmt-collect-translations




(a) Same Vocabulary (Small) (b) Same Vocabulary (Large) (c) Different Vocabularies

Figure 5: ΔCOMET results on our custom MT models using ABE. ΔCOMET is the improvement of ensembling two models via ABE over the best individual model. Individual COMET scores are displayed on axes. Labeling indicates the vocab size, followed by the epoch checkpoint. All results on `en-de` WMT24.

### 4.2 Baselines

We have two baseline generation algorithms to compare the results of our ensembling. The first is vanilla translation: using the model as intended. The MT models only pass the source input (with some language id tags for the multilingual models) to the huggingface `generate` function. For TOWER and LLaMa, we use the huggingface `pipeline` function with the prompts above (explicitly listed in Table 7).

We additionally consider linear interpolation as an ensembling baseline. In this traditional setting, the two models' output distributions can only be interpolated when they are over the same event space (i.e., have the same vocabulary). We only run this baseline over our custom MT models, using different checkpoints along the training trajectories of the different models. We use a beam size of 5 for both baselines and ABE for all models. We generate with a maximum length of 256 tokens.[10]

## 5 Results

We demonstrate the effectiveness of our ensembling algorithm by comparing the sequences generated by ABE with the performance of the best individual model. Given two models $m_i$, $m_j$, the translations produced by either model alone are $\texttt{T}_i$ and $\texttt{T}_j$, respectively. The translations produced by ensembling these two models with ABE are denoted as $\texttt{ABE}_{i,j}$. We define the delta as:

$$\Delta\mathcal{S} = \mathcal{S}(\texttt{ABE}_{i,j}) - \max(\mathcal{S}(\texttt{T}_i), \mathcal{S}(\texttt{T}_j)) \quad (2)$$

where $\mathcal{S}$ may refer to BLEU or COMET scores.

[10]If a model is stalled at this length, there is no agreed-upon hypothesis, and we return an empty string.

### 5.1 Custom MT Models

In Figure 5, we display the ΔCOMET scores across various combinations of custom MT models (Refer to Figure A in Appendix for the ΔBLEU scores). We see consistent positive improvements across many checkpoints. In Figure 5(a) and Figure 5(b), we ensemble the smallest and largest custom MT models with vocabulary sizes of 8k and 64k, respectively, across various checkpoints. Further, we successfully do token-level ensembling of models with differing vocabularies (Figure 5(c))—a previously impossible task.

A persistent trend we find is that under-fitted models (e.g., Ep. 1) do not ensemble well. This is evidenced by negative ΔCOMET scores across the first row. In all other combinations, we see improvement, thus demonstrating the power of ensembling via ABE over using individual models.

| BLEU | ΔInterpolation | ΔABE |
|---|---|---|
| 27.7 | 0.16 | 1.07 |

Table 1: We report the average Δ BLEU for all model pairs using interpolation or ABE over the average maximum individual score.

We also seek to demonstrate that these ensembling results are at least as good as a naive interpolation-based ensembling baseline. To do this, we compare the relative improvement using interpolation-based ensembling to the improvement gained from ABE. Note that this restricts the ensemble setting as the vocabularies *must* match. In Table 1, we display the relative ΔBLEU improvements and see that ABE is often a bigger improvement in these models.

### 5.2 Public MT Models

Our custom models are well-suited for ABE, since they were trained on the same data and potentially have related vocabulary distributions even when their vocabularies differ. We next consider models over which we have less control. As large multilingual models, M2M and NLLB are pretty different from our custom ones. Figure 6 shows ΔBLEU from ABE, with improvements in many cases, though not universal, as seen in Custom MT.

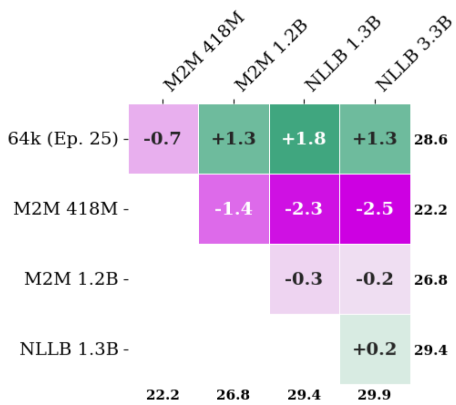



Figure 6: ΔBLEU of ensembling different MT model pairs using ABE. This includes our largest custom model (bilingual) and publicly available multilingual models. Individual BLEU scores displayed on axes.

We see an improvement when ensembling our largest custom model (64k) with larger multilingual MT models. We suspect the smaller multilingual model (M2M 418M) performs less well than a bilingual model or a larger multilingual model, and these negative trends may be further examples of underfit models. The ΔCOMET scores (displayed in the Figure 10) with ABE are more negative than their BLEU equivalents. This may indicate that ABE does better at surfacing particular $n$-grams but may affect other aspects, such as fluency, that COMET or other neural metrics may penalize.

### 5.3 Off-the-Shelf LLMs

We demonstrate the algorithm's flexibility by extending our ensembling results to distinct architectures—encoder-decoder with decoder-only LLMs. In Figure 7, we display ΔBLEU improvements. In this section, we only display 3-shot experiments with LLaMa, but a more comprehensive results table is available in Figure 9 in the Appendix.

We still see consistent positive gains from ensembling models—particularly when ensembling


Figure 7: ΔBLEU of ensembling various encoder-decoder models with LLMs using ABE. Individual BLEU scores displayed on axes.

the bilingual models with the larger multilingual models. One crucial trend we notice is that poorer performing models, such as the smaller instances of M2M or LLaMa get consistent *negative* results. This indicates that poorer-performing models will only deteriorate the performance of the better model, which is also typical of other ensembling approaches. However, we see improvements when ensembling across architectures: +2.7 BLEU when ensembling a small bilingual model with Tower or LLaMa. We further see improvements when ensembling two LLMs (+1.4 with Tower and LLaMa8b). As before, we observe more negative results when using COMET (Figure 10 in the Appendix).

### 5.4 Additional languages

To further demonstrate the generalizability of our results beyond en-de translation, we evaluate ABE on additional target languages (`cs`, `es`, `uk`). We ensemble NLLB, Tower, and LLaMa, with results presented in Appendix B.

## 6 Analysis

We investigate why our ensembling succeeds in some cases but not others through quantitative and qualitative analysis.

### 6.1 Model Preferences

One effect we wish to disentangle is whether ABE improves the search space or the modeling. Interpolation-based ensembling only affects the intermediate token probabilities (a modeling change) and does not change the search procedure. ABE does both by severely altering the search and mildly altering the modeling (scoring by the weighted sum of two models instead of one).

To investigate, we quantify each model's preference for the generated translations. Given $m_1$, $m_2$, we get the associated individual translations ($\mathtt{T}_1$, $\mathtt{T}_2$) and the ensembled ABE translation ($\mathtt{ABE}_{m_1}$, $\mathtt{ABE}_{m_2}$), respectively. We rank these three translations by likelihood under $m_1$ and $m_2$. Table 2 shows models that ensemble well (custom MT models) consistently prefer the ABE output, agreeing 86% of the time. In contrast, mixed preferences appear with M2M and NLLB ($\Delta$BLEU = -0.2), suggesting that ABE cannot overcome underlying modeling disagreements, and is more effective in exploring the search space when models agree.

| | | $\mathtt{T}_1$ | $\mathtt{T}_2$ | ABE | Same % |
|---|---|---|---|---|---|
| 8k+64k | $m_1$ | 102 | 106 | 2207 | 86.0 |
| | $m_2$ | 198 | 223 | 2028 | |
| M2M+NLLB | $m_1$ | 1002 | 1012 | 1092 | 54.5 |
| | $m_2$ | 840 | 809 | 1096 | |

Table 2: Preference. Top: $m_1$ and $m_2$ are our bilingual 8k and 64k models (+$\Delta$ under ABE). Bottom: $m_1$ and $m_2$ are M2M1.2B and NLLB3.3B (-$\Delta$ with ABE). $\mathtt{T}_i$ shows counts when outputs of $m_i$ were ranked highest (or tied). ABE shows counts when the outputs of the ensemble were ranked highest. "Same %" designates when models had the same ranking.

### 6.2 Constraining Hallucinations

Standard (same-vocabulary) ensembling can normalize models, helping increase their robustness to noise. Upon examining outputs, we found a recurring trend that ABE also helps prevent models that have begun to hallucinate. An example is shown in Table 3. Here, noisy inputs included by design in the WMT24 test sets occasionally trip up individual models, including Llama-3.2 (3B-Instruct-3-SHOT). Using ABE on all pairs of these models yields the correct output.

| source | lfg $sqqq |
|---|---|
| 16k | lfg $sqqq {m} {m} {m} {m} {m} ... |
| 64k | lfg $qqqq$qqqqqqqqqqqqqqqqqq... |
| Llama | Es scheint, dass das ursprüngliche Textstück fehlt oder nicht verfügbar ist. Die gegebene Zeichenkombination "lfg $sqqq" ist nicht... |
| ABE | lfg $sqqq |

Table 3: (Truncated) examples of individual models hallucinating or becoming overly verbose on noisy input, but in different ways. Any ABE pairing of these models produces the correct output.

## 7 Conclusion

We have presented an algorithm enabling token-level ensembling models with distinct vocabularies. In contrast to prior work, our approach requires no learned mappings of token representations (Xu et al., 2024), high-vocabulary overlap (Yu et al., 2024), a base model (Yao et al., 2024), or other model fine-tuning. Instead, we run models in parallel, using a classical approach from parsing and statistical machine translation to select tokens whose surface representation all models agree on. We believe the algorithm itself is an interesting contribution to the literature, since it enables (and makes easy) a previously impossible task. Traditional ensembling is a technique that introduces improvements in some, but not all, settings. It is therefore interesting that our approach also (a) produces gains in various MT settings and (b) also often improves over standard ensembling.

Prior ensembling methods have primarily targeted tasks with multiple-choice or span evaluation of the answer. In contrast, our work focuses on open-ended generation, specifically MT – a setting with far greater flexibility in output space. MT also serves as a compelling testbed for our method. For one, ensembling is often used to produce higher-quality distilled results. Second, the translation task helps constrain the generative output to a subset of tokens, meaningfully capturing the source semantics. That being said, our agreement-based approach might falter in less constrained tasks. The implementation is conceptually simple and factored, and allows for easy experimentation with different methods for agreement-based search. We therefore view this as a fruitful topic for future research.

## Limitations and Ethics

We note a few limitations with our work. The first is our focus on one task of machine translation, which is heavily conditioned on the input, and the accepted translation set is relatively small compared to other tasks. Though this approach works on Large Language Models, it may not easily extend to other more diverse tasks such as summarization.

We also acknowledge that machine translation is still a generation task, and is prone to the typical generation pitfalls of hallucinations, or erroneous translations—particularly when using LLMs. Overly relying on error-prone automated translation without a human review can have unintended consequences when used as a means of distributing information.

The authors also acknowledge the assistance of LLMs in the work in this paper—in particular using AI agents like CoPilot and ChatGPT to write code and edit plots.

## 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. *Preprint*, arXiv:2402.17733.

Mikel Artetxe and Holger Schwenk. 2019. Margin-based parallel corpus mining with multilingual sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 3197–3203. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2013. Report on the 10th IWSLT evaluation campaign. In *Proceedings of the 10th International Workshop on Spoken Language Translation: Evaluation Campaign*, Heidelberg, Germany.

Yu Chen and Andreas Eisele. 2012. MultiUN v2: UN documents with multilingual alignments. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2500–2504, Istanbul, Turkey. European Language Resources Association (ELRA).

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzman, and Philipp Koehn. 2020. Ccaligned: A massive collection of cross-lingual web-document pairs. *Preprint*, arXiv:1911.06154.

Ahmed El-Kishky, Adithya Renduchintala, James Cross, Francisco Guzmán, and Philipp Koehn. 2021. XLEnt: Mining a large cross-lingual entity dataset with lexical-semantic-phonetic word alignment. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10424–10430, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Miquel Esplà, Mikel Forcada, Gema Ramírez-Sánchez, and Hieu Hoang. 2019. ParaCrawl: Web-scale parallel corpora for the languages of the EU. In *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, pages 118–119, Dublin, Ireland. European Association for Machine Translation.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *CoRR*, abs/2010.11125.

António Farinhas, José de Souza, and Andre Martins. 2023. An empirical study of translation hypothesis ensembling with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11956–11970, Singapore. Association for Computational Linguistics.

Thamme Gowda, Roman Grundkiewicz, Elijah Rippeth, Matt Post, and Marcin Junczys-Dowmunt. 2024. PyMarian: Fast neural machine translation and evaluation in python. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 328–335, Miami, Florida, USA. Association for Computational Linguistics.

Thamme Gowda, Zhao Zhang, Chris Mattmann, and Jonathan May. 2021. Many-to-English machine translation tools, data, and pretrained models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 306–316, Online. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

L.K. Hansen and P. Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

Anna Hätty, Simon Tannert, and Ulrich Heid. 2017. Creating a gold standard corpus for terminological annotation from online forum data. In *Proceedings of Language, Ontology, Terminology and Knowledge Structures Workshop (LOTKS 2017)*, Montpellier, France. Association for Computational Linguistics.

Kenneth Heafield, Elaine Farrow, Jelmer van der Linde, Gema Ramírez-Sánchez, and Dion Wiggins. 2022. The EuroPat corpus: A parallel corpus of European patent data. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 732–740, Marseille, France. European Language Resources Association.

Hieu Hoang, Huda Khayrallah, and Marcin Junczys-Dowmunt. 2024. On-the-fly fusion of large language models and machine translation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 520–532, Mexico City, Mexico. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic. Association for Computational Linguistics.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. 2024. Enabling ensemble learning for heterogeneous large language models with deep parallel collaboration. *Preprint*, arXiv:2404.12715.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin Popel, Maja Popović, and 3 others. 2024. Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46, Miami, Florida, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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

Khanh Nguyen and Hal Daumé III. 2019. Global Voices: Crossing borders in automatic news summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 90–97, Hong Kong, China. Association for Computational Linguistics.

Byung-Doh Oh and William Schuler. 2024. Leading whitespaces of language models' subword vocabulary pose a confound for calculating word probabilities. *Preprint*, arXiv:2406.10851.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Matt Post, Thamme Gowda, Roman Grundkiewicz, Huda Khayrallah, Rohit Jain, and Marcin Junczys-Dowmunt. 2023. Sotastream: A streaming approach to machine translation training. *Preprint*, arXiv:2308.07489.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Roberts Rozis and Raivis Skadiņš. 2017. Tilde MODEL - multilingual open data for EU languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 263–265, Gothenburg, Sweden. Association for Computational Linguistics.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021a. WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia. In *Proceedings of the*

*16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361, Online. Association for Computational Linguistics.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, Armand Joulin, and Angela Fan. 2021b. CCMatrix: Mining billions of high-quality parallel sentences on the web. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. 2024. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12974–12990, Bangkok, Thailand. Association for Computational Linguistics.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, and Dániel Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

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, and 20 others. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Jörg Tiedemann. 2020. The tatoeba translation challenge – realistic data sets for low resource and multilingual mt. *Preprint*, arXiv:2010.06354.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. 2024. Bridging the gap between different vocabularies for LLM ensemble. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7140–7152, Mexico City, Mexico. Association for Computational Linguistics.

Yuxuan Yao, Han Wu, Mingyang Liu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. 2024. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. *Preprint*, arXiv:2410.03777.

Yao-Ching Yu, Chun Chih Kuo, Ye Ziqi, Chang Yucheng, and Yueh-Se Li. 2024. Breaking the ceiling of the LLM community by treating token generation as a classification for ensembling. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1826–1839, Miami, Florida, USA. Association for Computational Linguistics.

# A Appendix

Below we describe each step of our filtering pipeline:

1. Remove items when equal to source and target pair in our validation set.
2. Remove lines without both source and target.
3. Remove lines where langid (Lui and Baldwin, 2012) on source is < 0.5 for English and on target is < 0.5 for German.
4. Remove lines when more than half of the line is punctuation.
5. Remove lines that have too many characters with frequencies outside of the expected language set (Fan et al., 2020).[11]
6. LASER based Margin-scoring (Artetxe and Schwenk, 2019) (done in 2.5M line chunks for computation).
7. Deduplicate all training data.

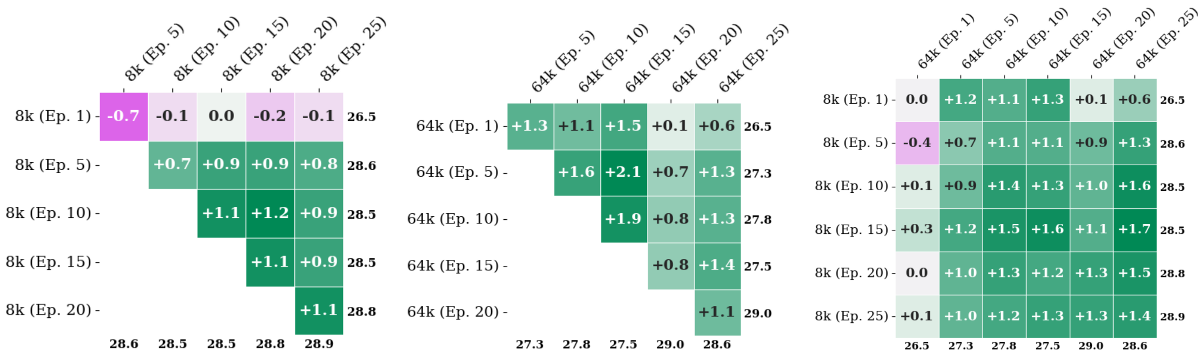
[11] https://github.com/facebookresearch/fairseq/blob/main/examples/m2m_100/README.md

| Data Name | Filtered Size | Paper (if applicable) |
|---|---|---|
| ELRC | 6.5M | |
| ELRA | 66k | |
| EU (dcep, eac, ecdc) | 1.8M | |
| Wikimatrix | 5.6M | Schwenk et al. (2021a) |
| WikiTitles | 2.9M | |
| TedTalks | 166k | |
| Bible | 35k | |
| OPUS Books | 43k | Tiedemann (2012) |
| CC-Aligned | 12M | El-Kishky et al. (2020) |
| CC-Matrix | 244M | Schwenk et al. (2021b) |
| DGT | 4M | |
| European Central Book (ECB) | 83k | |
| ELITR | 232k | |
| EMEA | 233k | |
| EU Bookshop | 5.1M | |
| EU Const. | 4k | |
| Europarl (v3,7,8,10) | 6.3M | Koehn (2005) |
| EuroPat (v1-3) | 47M | Heafield et al. (2022) |
| Global Voices | 174k | Nguyen and Daumé III (2019) |
| JRC | 457k | Steinberger et al. (2006) |
| KDE/GNome | 110k | Hätty et al. (2017) |
| MultiUN | 118k | Chen and Eisele (2012) |
| MultiCCAligned | 60M | |
| MultiParaCrawl | 70M | |
| News Commentary (v9,14,16) | 937k | |
| OPUS Train | 580k | Tiedemann (2012) |
| ParaCrawl (v9) | 242M | Esplà et al. (2019) |
| PHP | 7k | |
| QED | 400k | |
| Tanzil | 476k | |
| Tatoeba | 1.8M | Tiedemann (2020) |
| TED (2013) | 403k | Cettolo et al. (2013) |
| XLEnt | 1.4M | El-Kishky et al. (2021) |
| Tilde | 4.8M | Rozis and Skadiņš (2017) |
| StatMT 13 (CommonCrawl) | 1.8M | |
| Deduplicated | 618M | |

Table 4: We aggregate most English–German bitext listed on `mtdata` (available at https://github.com/thammegowda/mtdata). The above is the filtered text sizes.

**Algorithm 1** Agreement-Based Decoding Using Beam Search (One Time-Step)

```
"""
scores is a BEAM_SIZE x MODEL_NUMBER x VOCABULARY_SIZE list.
Section 3.2: scores are sorted so we can enumerate as shown in Figure 2
Section 3.3: if models are stalled, we only consider epsilon transitions
"""
scores = [[model.step(j) for model in models] for j in range(beam_size)]
scores = [torch.sort(beam_score) for beam_score in scores]
scores = mask_stalled_beams(scores)

class State:
  def __init__(self, beam_index, grid_indices, token_id, score):
    # set values ...
  def find_neighbors(self):
    # enumerate neighbors ...
  def score(self):
    # score is the weighted sum of model's beam scores

# now we search the cross-product of vocabulary items
next_beam = []
heap = heap()
for j in range(beam_size):
  """
  We seed (0 index for all model) our heap to search our grid (Figure 2).
  For stalled models, this is the epsilon transition
  The token_ids is the list of tokens belonging to each model's vocabulary
  The token_scores is the associated score of these tokens
  """
  token_ids = [scores[j][i].idx[0] for i in range(len(models))]
  token_scores = [scores[j][i].value[0] for i in range(len(models))]
  state = State(
              beam_index = j,
              grid_indices = [0 for _ in models],
              token_ids = token_ids,
              token_scores = token_scores)
  heap.push(state)

# now we expand the search space until we find beam_size agreeing extensions
while len(next_beam) < beam_size:
  item = heap.pop()

  # Each model has a local hypothesis (specific to internal state)
  local_hypotheses = [model.extend_beam(item) for model in models]

  # global hypothesis will define agreement
  global_hypothesis = determine_global(local_hypothesis)

  if agreement(local_hypotheses, global_hypothesis):
    next_beam.append(item)

  for neighbor in find_neighbors(item):
    heap.push(neighbor)

return next_beam
```



(a) Same Vocabulary (Small)


(b) Same Vocabulary (Large)


(c) Different Vocabularies

Figure 8: BLEU results on our custom English–German models using Agreement-Based Ensembling. These charts show the $\Delta$ BLEU improvement of ensembling two models via ABE over the best individual model. Labeling indicates vocab size followed by epoch checkpoint.

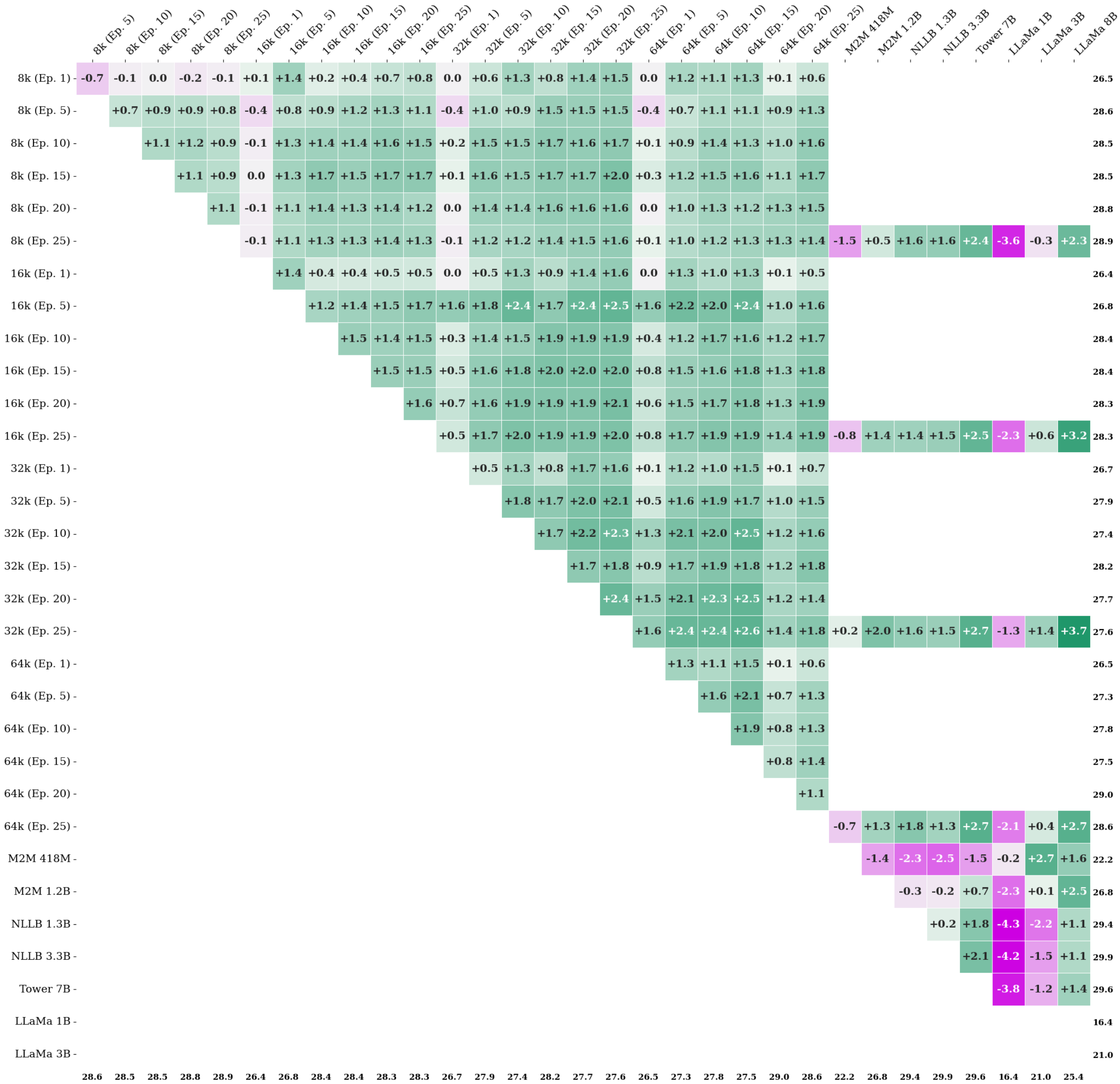



Figure 9: The Δ BLEU scores for all model pairs.

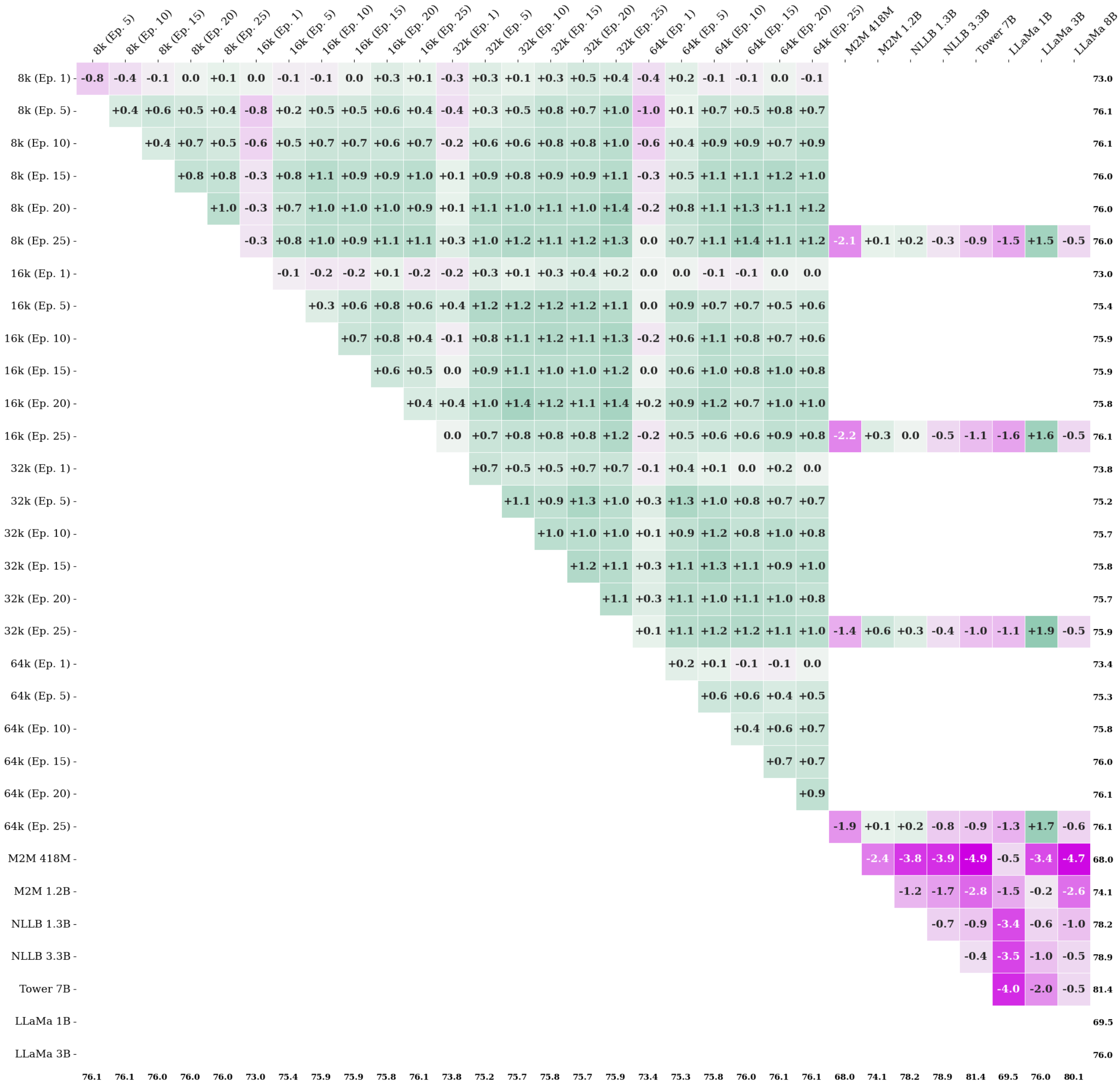



Figure 10: The Δ COMET scores for all model pairs.

| Hyper-Parameter | Value |
|---|---|
| label smoothing | 0.1 |
| learning rate | 0.0005 |
| lr warmup | 4000 |
| lr decay inv sqrt | 4000 |
| mini batch warmup | 4000 |
| mini batch | 1000 |
| mini batch words | 500000 |
| max length | 256 |
| mini batch fit | true |
| early stopping | 40 |
| logical epoch | 1Gt |
| shuffle | batches |
| fp16 | false |
| tied embeddings | true |
| tied embeddings all | true |
| dim emb | 1024 |
| enc depth | 6 |
| dec depth | 6 |
| transformer dim ffn | 8192 |
| transformer decoder dim ffn | 8192 |
| transformer depth scaling | true |
| lemma dim emb | 0 |
| transformer ffn activation | relu |
| transformer-heads | 8 |
| transformer dropout | 0.1 |
| transformer dropout attention | 0 |
| transformer dropout ffn | 0.1 |

Table 5: The above enumerate the Marian hyperparameters used for all of our custom models.

| Model Type | Repo ID/URL | $m$ Size | $V$ Size | Languages | License |
|---|---|---|---|---|---|
| LLM | meta-llama/Llama-3.1-8B-Instruct | 8B | 128k | de,es | LLaMa3 |
| | meta-llama/Llama-3.2-1B-Instruct | 1B | 128k | de, es | LLaMa3 |
| | meta-llama/Llama-3.2-3B-Instruct | 3B | 128k | de, es | LLaMa3 |
| | Unbabel/TowerInstruct-7B-v0.2 | 7B | 32k | de, es | CC-BY-NC-4.0, LLaMa2 |
| | Unbabel/TowerInstruct-Mistral-7B-v0.2 | 7B | 32k | de, es | CC-BY-NC-4.0, LLaMa2 |
| Public MT | facebook/m2m100_1.2B | 1.2B | 128k | de, es, cs, uk | MIT |
| | facebook/m2m100_418M | 418M | 128k | de, es, cs, uk | MIT |
| | facebook/nllb-200-1.3B | 1.3B | 256k | de, es, cs, uk | CC-BY-NC |
| | facebook/nllb-200-3.3B | 3.3B | 256k | de, es, cs, uk | CC-BY-NC |
| | facebook/nllb-200-distilled-1.3B | 1.3B | 256k | de, es, cs, uk | CC-BY-NC |
| | Facebook/nllb-200-distilled-600M | 600M | 256k | de, es, cs, uk | CC-BY-NC |
| Custom MT† | rewicks/baseline_en-de_8k_ep* | 286M | 8k | de | Apache 2.0 |
| | rewicks/baseline_en-de_16k_ep* | 294M | 16k | de | Apache 2.0 |
| | rewicks/baseline_en-de_32k_ep* | 310M | 32k | de | Apache 2.0 |
| | rewicks/baseline_en-de_64k_ep* | 343M | 64k | de | Apache 2.0 |

Table 6: Huggingface Repo Ids for our publicly available models. LLaMa3 license refers to https://www.llama.com/llama3/license/. LLaMa2 refers to https://ai.meta.com/llama/license/. Tower also states the LLaMa license as it uses the LLaMa 2 pretraining weights. Language set only covers those addressed in this paper.

† Each en-de custom MT model has 40 (epoch 1 to epoch 40) checkpoints, all of which are available in the above-mentioned URL-s.

| Shot | Prompt |
|---|---|
| 0 | [<br>`{"role": "system", "content": "Cutting Knowledge Date: December 2023\nToday Date: 26 Jul 2024"}`<br>`{"role": "user", "content": "Translate the following segment into XX. Do not add any additional content. Do not add parentheticals. Only provide the translation. The English segment:"}`<br>] |
| 3 | The example translations are identical to the WMT24 evaluation scripts specific to the target language. The examples can be found at `https://github.com/wmt-conference/wmt-collect-translations/tree/main/few_shots`. Each example is put in the same format. Language names exchanged when necessary:<br><br>[<br>`{"role": "user", "content": "Translate the following text from English into German. The English Segment: example source}`<br>`{"role": "assistant", "content": "{example translation}"}`<br>] |

Table 7: LLaMa prompting messages.

## B Additional Languages

We additionally study the ensembling of these models with ABE by comparing the performance in other languages (`cs`, `es`, `uk`). We compare NLLB, Tower, and LLaMa and display the results in Table 8. Similar to before, we notice mixed performance across model pairs and target languages. We suspect this is due to underlying model differences.

| | | $m_1$ | $m_2$ | ABE | $\Delta$ |
|---|---|---|---|---|---|
| NLLB + Tower | cs | 26.8 | 14.1 | 24.0 | -2.8 |
| | es | 43.2 | 41.0 | 44.4 | +1.2 |
| | uk | 26.3 | 6.1 | 24.1 | -2.2 |
| NLLB + LLaMa | cs | 26.8 | 19.6 | 25.3 | -1.5 |
| | es | 43.2 | 37.1 | 42.7 | -0.5 |
| | uk | 26.3 | 20.3 | 26.2 | -0.1 |
| Tower + LLaMa | cs | 14.1 | 19.6 | 21.7 | +2.1 |
| | es | 41.0 | 37.1 | 42.0 | +1.0 |
| | uk | 6.1 | 20.3 | 22.4 | +2.1 |

Table 8: BLEU scores for different ensembling pairs and their individual models. $m_1$ and $m_2$ denote the individual model score while ABE denotes the ensembled score. $\Delta$ is the difference between ABE and max($m_1$, $m_2$). The model versions are M2M 1.2B, NLLB 3.3B, Tower v0.2 7B, LLaMa 3.1 8B 3-shot.

Tower and LLaMa, a consistently successful ensembling pair, see improvements in all three languages. According to their respective documentation, neither model explicitly supports `cs` or `uk`. Still, there were likely substantial amounts of these languages in the pretraining data. We see improvements in the BLEU score in both models using ABE.

## C Sampling

One common use case with autoregressive models is sampling. As with other search procedures, standard ensembling works transparently with sampling. As a procedure, sampling is easy to implement with ABE. Instead of searching over the grid, we sample from each model consecutively (skipping over stalled models). The vocabulary which we sample from is renormalized to only allow for *agreeing* tokens.

We experimented with adding sampling to Agreement-Based Ensembling but found that it did not work well. We hypothesize the instability of sampling with this method stems in some part from the underlying idea that most tokenizers denote whitespace as *leading* (designating word beginnings) and not as *trailing* (designating word endings). This idea has been shown to have interesting effects on probability distributions (Oh and Schuler, 2024).

As an illustrative example, consider the following German indefinite articles: "ein" and "eine." The key difference being that "eine" is feminine. Both of these words are short and fundamental to the German vocabulary, so it is almost guaranteed that both words in their full form are in the model vocabulary. We further suspect that models with both of these words in their vocabulary have *never* seen "eine" tokenized as "_ein" + "e" in their training data.

Now consider our previously stated sampling procedure. Assume from $m_1$, we sample "_Eine." When conditioned on this decision, we are likely to see *both* "_Ein" and "_Eine" holding most of the probability mass of $m_2$. Let's assume we sample "_Ein" from $m_2$. Since the local hypothesis of $m_1$ ("_Eine") and the local hypothesis of $m_2$ ("_Ein") are in agreement, this is a valid state to be in. However, when we next sample from $m_2$ to catch up to $m_1$ it is *not* going to have a high probability on "e" because it has never seen "Eine" tokenized that way during training.

We understand that $m_1$ has implicitly decided to generate the entire word "Ein", but it was unable to convey that it was *also* modeling the end of that word due to the tokenization scheme.

Now consider a word-ending tokenization scheme. Now, $m_1$ samples "Eine_" signifying that it is *done* with this word. When we constrain the output of $m_2$ on this hypothesis, "Ein_" is *not* going to be sampled because it does not agree. In order to get into the same predicament, it would need to place high probability on "Ein", specifically *not* ending the word which is unlikely if both models wish to generate some version of the word "a."