Faster Machine Translation Ensembling with Reinforcement Learning and Competitive Correction

Kritarth Prasad*, **Mohammadi Zaki***, **Pratik Rakesh Singh** and **Pankaj Wasnik**
Media Analysis Group, Sony Research India
{kritarth.prasad, mohammadi.zaki, pratik.singh, pankaj.wasnik}@sony.com

Abstract

Ensembling neural machine translation (NMT) models to produce higher-quality translations than the L individual models has been extensively studied. Recent methods typically employ a candidate selection block (CSB) and an encoder-decoder fusion block (FB), requiring inference across *all* candidate models, leading to significant computational overhead, generally $\Omega(L)$. This paper introduces **SmartGen**, a reinforcement learning (RL)-based strategy that improves the CSB by selecting a small, fixed number of candidates and identifying optimal groups to pass to the fusion block for each input sentence. Furthermore, previously, the CSB and FB were trained independently, leading to suboptimal NMT performance. Our DQN-based **SmartGen** addresses this by using feedback from the FB block as a reward during training. We also resolve a key issue in earlier methods, where candidates were passed to the FB without modification, by introducing a Competitive Correction Block (CCB). Finally, we validate our approach with extensive experiments on English-Hindi translation tasks in both directions as well as English to Chinese and English to German.

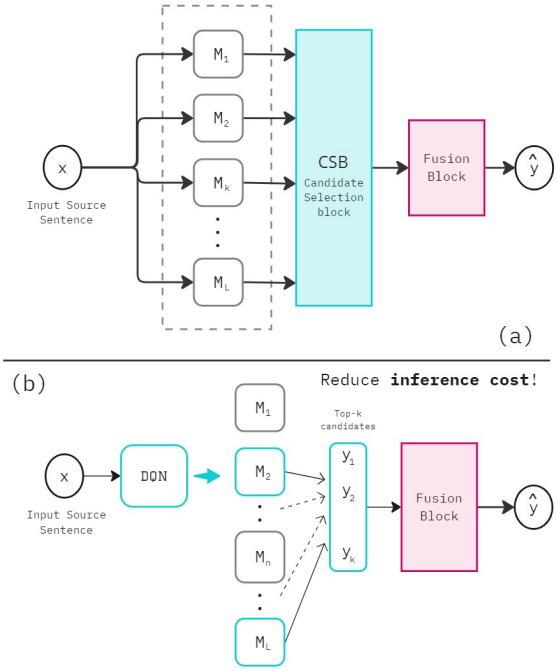

Figure 1: (a)The general strategy of ensembling MT systems, comprising of a candidate selection block followed by a fusion block, (b) Modified approach in this work.

1 Introduction

Recent advancements in machine translation have led to the availability of various open-source neural machine translation (NMT) models and general-purpose Large Language Models (LLMs) capable of translation tasks. However, the performance of these models depends on the model size and the quality of the pre-training data. As a result, ensembling methods have become essential for achieving state-of-the-art performance when low-parameter models are the only viable option due to resource constraints.

While traditional ensembling methods often rely on static weight sharing between models, these methods are now replaced by more dynamic approaches that adjust the weights of the component models depending on the specific source sentence and the quality of each candidate translation produced. A common approach to ensemble LLMs/MT models is depicted in Figure 1; it comprises a candidate selection block (CSB) that selects a subset of candidate translations, potentially followed by a fusion block (FB), which is commonly an encoder-decoder network that combines the input to produce a (usually) better output translation.

These approaches, however, require the source sentence $\mathbf{x}$ to be translated by each of the L candidate models in the first stage, which the CSB then takes as input. This significantly increases the time

*Equal contribution

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 8337–8350
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

complexity of the ensembling process, making it a major drawback, particularly when the inference cost of LLMs is a fundamental challenge (Li et al., 2024; Yuan et al., 2024; Cai et al., 2024). Additionally, our extensive experiments reveal that current CSB strategies are suboptimal in selecting the best candidates to pass to the FB (discussed in detail in Sec.2.2). This motivates us to address the need for an ensembling method that improves the quality of translations and deals with the high computational cost associated with traditional approaches. Specifically, we aim to design a strategy that delivers superior performance compared to state-of-the-art methods while reducing the inference time.

Thus, in this paper, we frame the candidate selection task as a reinforcement learning problem and employ a Deep Q-Network (DQN) block that efficiently identifies the group of optimal candidate translations (actions) for every source input sentence (state). Further, through experimental analysis, we identify that the overall performance of recent ensembling approaches is often limited by the worst-performing candidate in the system. To address this, we introduce a strategic candidate improvement method, which significantly enhances the performance of the fusion block.

Furthermore, the recent work of Lu et al. (2023) addresses the inference cost issue in current ensembling methods. However, their approach is a *selection-based* ensembling, while we focus on *generation-based* ensembling with the anticipation that the selected candidates can be further modified to produce potentially better translations. Our key contributions can be summarized as follows:

- We introduce **SmartGen**, a novel strategy that leverages DQN to dynamically select a subset of candidate MT models based on the source sentence, which are then fused in the next step. This approach significantly reduces the inference time of the ensembling method by only inferring from a subset of MT models (Figure 4).

- To enhance translation quality further, we propose a correction strategy (Sec. 3.2) that selectively improves candidates using rejected ones. This combined methodology, referred to as **SmartGen++**, achieves significant performance gains with a modest increase in inference cost.

- We demonstrate the effectiveness of the proposed strategy through extensive experiments on English-to-Hindi and Hindi-to-English translation tasks, using four publicly available datasets and a private datasets, evaluated across multiple metrics.

2 Background and Motivation

2.1 Previous Work

Ensemble Learning and Ranking: Ensemble learning is a widely used technique that combines the strengths of individual models to produce improved outputs (Sagi and Rokach, 2018; Jia et al., 2024). With the recent advancements in large language models, ensemble learning has become instrumental in employing multiple experts to tackle complex tasks based on their respective expertise (Guo et al., 2024). Studies have shown that ensembling can be achieved through model weights (Wan et al., 2024; Goddard et al., 2024) or by combining model outputs. In this context, the recent mixture of experts (MoE) approach has emerged as an effective technique for significantly boosting performance by merging outputs from multiple expert models (Artetxe et al., 2022).

Recent research introduces various selection methods prior to the fusion step, selecting only the best candidates from the experts for fusion. Significant work has been carried out in the field of summarization using different techniques such as training reranking models based on metrics (Ravaut et al., 2023), employing contrastive learning for effective ranking (Liu and Liu, 2021), and using pairwise ranking to compare candidates (Jiang et al., 2023). However, there has been limited exploration of such methods in the task of machine translation (Hoang et al., 2024a). Moreover, existing work does not establish a clear connection between the top-ranked candidates and the final fusion output, as discussed in detail in Sec. 2.2.

LLM Refinement: As a result of continuous advancements in LLMs, their reasoning capabilities have significantly improved. This has prompted studies on enhancing performance through various strategic self-correction methods (Pan et al., 2023). Previous research has investigated self-scoring and self-correction techniques across various domains (Liu et al., 2023; Madaan et al., 2023), along with the development of automated feedback systems (Xu et al., 2023). In the context of NMT systems, common approaches include improvements through several iterations (Chen et al., 2024) and post-editing strategies for correcting translations

(Raunak et al., 2023). However, self-improvements in selected candidates in ensembling are yet to be explored, as discussed in detail in 3.2.

RL for Ensembling and Model Selection: Several notable works have employed reinforcement learning (RL), particularly multi-armed bandits, for model selection across various domains. Applications include load forecasting (Feng and Zhang, 2018), wind-speed prediction (Kosana et al., 2022), airline pricing (Shukla et al., 2019), and the development of online and scalable model selection strategies (Xie et al., 2021), among others.

2.2 Motivation

Motivation for DQN Block: As discussed, the tried and tested approach to combine an ensemble of candidate translations to produce higher quality translations is by first comparing the candidate translations and scoring them, before passing the top K candidates to an encoder-decoder block for fusion. However, this approach to handling output translations $(y_1, y_2, \ldots, y_L)$ is suboptimal, as the selection module is trained independently of the fusion block (Hoang et al., 2024b). Specifically, the chosen candidates may perform very well individually, however, they may not be the best choice for the fusion block to produce the best results effectively.

To test our assumption, we conducted an experiment using a pool of 8 MT systems for English-to-Hindi translation. In the set of $\{1, 2, ..., L\}$ candidates (MT systems), the optimal candidates which provide the best BLEU score can be any K-subset of the possible $\binom{L}{K}$ combinations of $\{1, 2, ..., L\}$ MT systems. Let the translations generated by these K candidates are denoted as $\{y_{n_1}, y_{n_2}, ...y_{n_K}\}$. For every sentence in a chosen test set, we observe by brute force the optimal triplet $(y_{n_1}, y_{n_2}, y_{n_3})$ of candidate translations (we choose $K = 3$ here, in line with the choice in Jiang et al. (2023), also see Sec. 4.6) which leads to achieving the best performance (in terms of BLEU score) when passed through the fusion block. In Figure 2, we compare the number of times various SOTA ensembling methods select a specific triplet with the brute-force strategy described earlier, which determines the optimal triplet every time. The plot demonstrates that for different sentences in the test sets, different triplets yield optimal performance, with no clear preference for any particular subset across all source sentences. In contrast, SOTA algorithms tend to favor certain triplets, leading to a gap between the achieved and the achievable performance.

We instead opt to choose a dynamic strategy, in that it explores the available action space for possibly better candidate groups. We, therefore, choose a DQN for action selection strategy and train it with the help of the reward obtained from the *final* translation obtained from the fusion block, instead of treating it independently.

Motivation for the Competitive Correction Block: Another crucial observation that we make is that the performance of the fusion block is limited by and degrades with the 'worst' input candidate translation. To verify this, we perform an experiment where 1) we pass the 'reference' (actual translation) sentence to the fusion block repeatedly K times, and compare it with 2) reference translation + top $(K-1)$ translations from the candidate pool. We report the result in Table 1. This simple experiment demonstrates that even though the selection block successfully manages to pick the best candidates from the pool, the performance of the final translation severely degrades (BLEU decreased by more than 17 points) depending on the 'other' chosen translations.

Candidates	BLEU
Reference $\times K$	85.83
Reference + Top $K-1$ BLEU candidates	68.38

Table 1: A comparison of the performance of the reference sentence repeated K times vs. reference appended with $K-1$ other candidates. The task is carried out for English to Hindi on a subset of human translated test set.

This limitation restricts all methods following the usual process of ensembling (Figure 1), ie., select and fuse. To mitigate this issue, we introduce a candidate improvement strategy, which we explain in detail in Sec. 3.2, which aims to identify and subsequently improve the weak candidate translations from the set of selected candidates before passing to the fusion block.

3 Proposed Methodology

In this section, we first present the problem statement and define the necessary notations, followed by the details of the proposed methodology.

Problem Statement: Given a set of L machine translation (MT) systems (e.g., NMT systems, LLMs), denoted as $M_1, M_2, \ldots, M_L$, the objective is to translate a source sentence $\mathbf{x}$ into a tar-

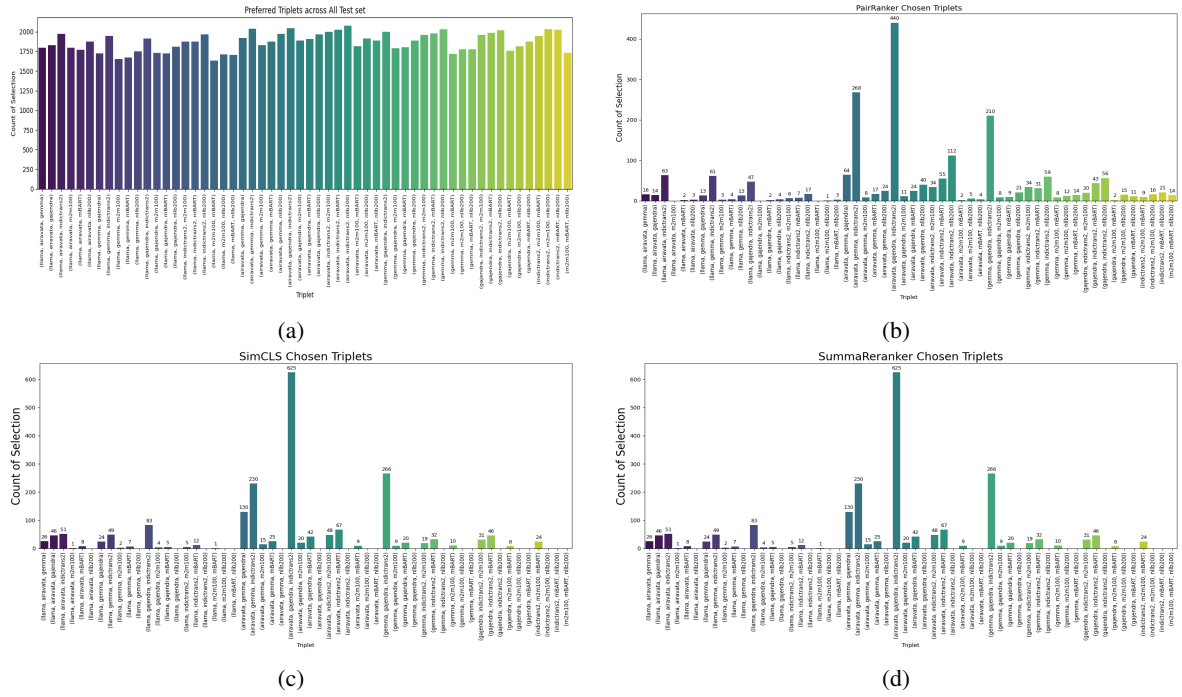

Figure 2: A comparative analysis of the distribution of the number of times candidate triplets are chosen in a) brute-force strategy, b) Pair-Ranker (Jiang et al., 2023), (c) SimCLS (Liu and Liu, 2021), and (d) SummaRanker (Ravaut et al., 2023)

get language such that the output sentence $\widehat{y}$ has higher quality than any of the individual candidate translations $(y_1, y_2, \ldots, y_L)$, while minimizing the computational overhead during inference.

The proposed methodology proceeds through three main phases: 1) candidate translation selection via the DQN network, 2) improvement of the selected translations with the Competitive Candidate Improvement (CCI) block, and 3) Fusion of the selected candidates with standard Encoder-Decoder (Jiang et al., 2023) block. A block diagram of SmartGen(++) is shown in Figure 3. We present details of these modules in the subsequent sections.

3.1 Candidate Translation Selection using DQN Block

Here, we employ a Deep Q-Network (DQN) (Mnih et al., 2013) to select the group of translations that we pass to the fusion block. Formally, we model the candidate (*action*) selection process as a discounted Markov Decision Process (MDP) $\mathcal{M}(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ represents the state space, consisting of all possible embedding vectors of the input sentence $\mathbf{x}$. Let $\mathcal{E}(.)$ be the encoder model of choice, then $\mathcal{S} \equiv \{s \in \mathbb{R}^d : s = \mathcal{E}(x), x$ is a sentence in the source language$\}$. The action space $\mathcal{A}$ is defined as the set $1, 2, \ldots, L$, denoting the index of a candidate MT system. The DQN outputs an index (action) a on being fed the state s corresponding to the current sentence x and we obtain the translation y from that candidate MT system. In this way, we select the candidate translations $\widehat{T} := y_{n_1}, y_{n_2}, \ldots, y_{n_K}$ to pass to the fusion block by choosing $\arg TopK_a, Q(s, a)$, where $Q(.,.)$ represents the DQN block. Further, γ is a discount factor whose value is in the range $[0, 1)$. The $\mathcal{P}$ is a probability transition kernel, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$, which gives a distribution over the next state, given the current state and the action. In our setting $\mathcal{P}$ is a deterministic function.

We use a standard experience replay buffer (Mnih et al., 2013) to store and resample tuples $\langle s, a, s', r \rangle$ for training the DQN network, where $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a function that assigns a scalar reward based on the action taken in a given state. To train the DQN, we use the sacreBLEU (Post, 2018) score (normalized between [0,1]) of the final output translation $\widehat{y}$ generated by the fusion block.

3.2 Competitive Correction Block (CCB)

Building on the motivation from Section 2.2, we introduce the Competitive Correction Block (CCB), designed to selectively enhance the quality of input translations passed to the fusion block. This strategy can be applied to other ensemble approaches

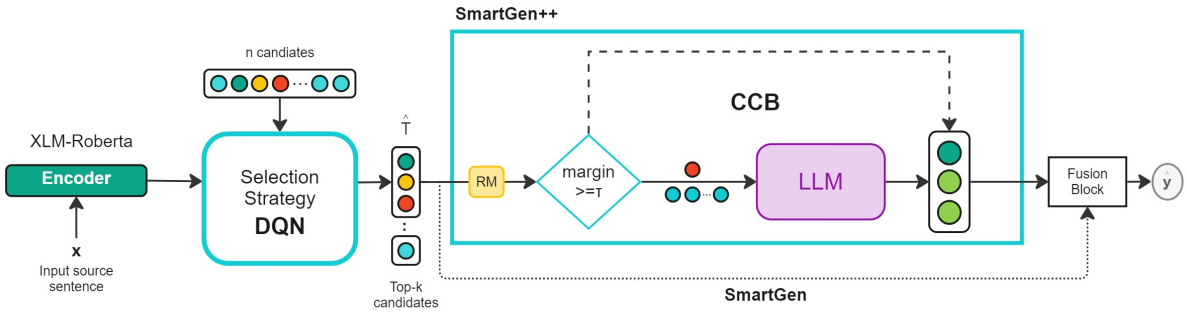

Figure 3: Detailed block diagram of our proposed methodology. Here DQN+FB represents SmartGen, and DQN+CCB+FB represents SmartGen++ as explained in text.

that rely on a select-and-fuse mechanism, making it of independent interest for improving final translation quality. The proposed CCB block consists of two key components: 1) the reward model and 2) the correction block.

Reward Model (RM): To evaluate the quality of selected candidates, we develop a Reward Model (RM) specifically designed to align Large Language Models (LLMs) with human preferences, providing a nuanced reward signal. Traditional reward modeling frameworks (Stiennon et al., 2022; Ouyang et al., 2022) face limitations due to their bias towards a *single* preferred response and their inability to capture contextual relationships among *multiple* similar candidates— a critical factor in translation tasks where multiple candidates may be very similar. To address this, we introduce a more comprehensive training methodology that considers a *set* of both preferred and rejected responses for each sample. Specifically, for each input sample $\mathbf{x}$, we select *four* preferred responses, denoted by $\mathcal{P}$ (including human preference and the top-3 BLEU candidates), with the remaining candidates designated as rejected, denoted by $\mathcal{R}$. This approach ensures a diverse, high-quality ensemble of responses that reflects the complexity of human preferences and the contextual relationships between candidates and reward scores. Thus, we can write modified RM loss as:

$$\mathcal{L}_{\text{RM}}(\theta) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D},\, y_p\in\mathcal{P},\, y_r\in\mathcal{R}}\Big[\log\sigma\big(r_\theta(\mathbf{x}, y_p) - r_\theta(\mathbf{x}, y_r)\big)\Big] \quad (1)$$

where $r_\theta(\mathbf{x}, y)$ is the scalar output of the reward model for input sample $\mathbf{x}$ and target candidates y with parameters θ, y_p and y_r represents preferred and rejected candidates respectively and $\mathcal{D}$ is the dataset used for training. The RM block is used to evaluate the quality of the selected candidates $\widehat{T}$, and depending on a user-specified 'margin', candidates are passed to the correction block, described next.

Correction Block: The functioning of the Correction Block (CB) is outlined in Alg. 1. Specifically, based on the reward margin between the selected candidates, we aim to enhance their quality using an LLM $\mathcal{G}$ and the set of rejected candidates from the DQN. We anticipate that by providing candidate translations and their respective rewards to $\mathcal{G}$, the model can generate a new candidate, $\widehat{I}$, that surpasses the current reward of I and the rewards of the candidates in $[L]\backslash\widehat{T}$. The exact prompt used for this process is detailed in the in Figure 7

Algorithm 1: Competitive Correction Block (CCB)

Terminologies: DQN ranked output (sorted based on rewards): $\widehat{T}$, Rewards of $\widehat{T}$: $R_{\widehat{T}}$, Rejected Candidates by DQN: $[L]\backslash\widehat{T}$, threshold: τ, Enhancer LLM: $\mathcal{G}$, Candidates of $[L]\backslash\widehat{T}$: $\{c_1, c_2, c_3, ..., c_{L-K}\}$, Reward of $L\backslash\widehat{T}$: $\{b_1, b_2, ..., b_{L-K}\}$

Input: $\mathcal{G}$, τ, $\widehat{T}$, $R_{\widehat{T}}$, $[L]\backslash\widehat{T}$

$R_{\widehat{T}} \leftarrow \{r_1, r_2, .., r_K\}$

$margin \leftarrow \{r_1, r_1 - r_2, ..., r_{K-1} - r_K\}$

for m *in* $2 : (K)$ **do**

  if $margin[m] \geqslant \tau$ **then**

    $I \leftarrow \widehat{T}[m]$

    $R_I \leftarrow R_{\widehat{T}}[m]$

    $\widehat{I} \leftarrow \mathcal{G}((I, R_I), \{(c_{1:L-K}, b_{1:L-K})\}$

    $\widehat{T} \leftarrow \widehat{T}\backslash I \cup \widehat{I}$

Output: $\widehat{T}$

4 Experiments

In this section, we present the experiments conducted to evaluate the effectiveness of our proposed methodology, detailing the datasets, baselines, setup and key results.

4.1 Datasets

For our experiments, we focus primarily on English-to-Hindi and Hindi-to-English translation tasks (en↔hi), using a diverse set of open-source and private datasets for training and evaluation. In addition, we also carry out experiments on Enlish-to-Chinese (en→zh) and English-to-German (en→de) using publicly available datasets. For the en↔hi tasks, the reward model was trained on a high-quality, privately collected dataset, and evaluations were conducted on benchmark datasets such as Flores (Goyal et al., 2021), WMT-14 (Bojar et al., 2014), IN22-Gen, and IN22-Conv (Gala et al., 2023a) along with entertainment-domain private data. For en→zh and the en→de translation tasks, we utilized the WMT18 (Bojar et al., 2018) and WMT19 (Barrault et al., 2019) datasets for training and testing respectively. Detailed dataset statistics can be found in the Appendix in Table 8.

Comment on private data. The private data used in the en↔hi experiments, is an internally created dataset consisting of high-quality human-annotated parallel data from entertainment content for Eng-Hin pair. This dataset represents textual data with challenging characteristics like varied emotions, cultural nuances and colloquiality, which makes the proposed ensembling method robust against challenging datasets such as this. Further, to have fair comparison all the baseline methods were also trained on this private dataset.

4.2 Baselines

en↔hi. For candidate translation generation, we utilize a combination of large language models (LLMs) such as Llama-3-8B (AI@Meta, 2024), Gemma-2-9B (Team, 2024), Airavata (Gala et al., 2024), Gajendra, and state-of-the-art machine translation models including IndicTrans2 (Gala et al., 2023b), NLLB200 (Team et al., 2022), m2m100 (Fan et al., 2020), and mBART (Liu et al., 2020).

en→zh. We use Aya-101 (Üstün et al., 2024), Deepseek-v2-lite-chat (DeepSeek-AI et al., 2024), Qwen2-7B-Instruct (Yang et al., 2024), Llama-3-8B (AI@Meta, 2024), Gemma-2-9B (Team, 2024), NLLB200 (Team et al., 2022), m2m100 (Fan et al., 2020), and mBART (Liu et al., 2020) as our candidate models.

en→de. We use Aya-101 (Üstün et al., 2024), Granite-3.0-8b-instruct (Granite Team, 2024), Qwen2-7B-Instruct (Yang et al., 2024), Llama-3-8B (AI@Meta, 2024), Gemma-2-9B (Team, 2024), NLLB200 (Team et al., 2022), m2m100 (Fan et al., 2020), and mBART (Liu et al., 2020) as our candidate MT systems.

Reward and Fusion models. For reward model training, we leverage the pre-trained Bloom-1b7 (Scao et al., 2023). The fusion model is based on mT5-large (Xue et al., 2021), and mdeberta-v3-base (He et al., 2021) is used for all baseline ranking systems. For encoding candidates during DQN state generation, we employ XLM-RoBERTa (Conneau et al., 2020). For GPT scoring we have GPT-4o (OpenAI et al., 2024)

4.3 Evaluations Metrics

We use SacreBLEU (Post, 2018) as the primary evaluation metric for our experiments. Additionally, we evaluate our models using other popular machine translation metrics such as chrF++ (Popović, 2015), WMT22-Comet-DA (Rei et al., 2020) (referred to as Comet).

4.4 Training Details

For training of the baseline rankers (SimCLS (Liu and Liu, 2021), SummaReranker (Ravaut et al., 2023), PairRanker (Jiang et al., 2023)), we utilized the default setting from Jiang et al. (2023). For DQN training, we implemented a ResNet-based DQN (Mnih et al., 2013; He et al., 2015) as the base model. The architecture begins with a linear layer that maps the input state dimension of 768 to a hidden dimension of 256, followed by three residual blocks. Each residual block contains two fully connected layers with ReLU activations and skip connections to improve gradient flow and model performance. The network concludes with an output linear layer that generates Q-values for the L possible actions. We trained the DQN using a 10% uniformly sampled subset of the dataset. The model converged after a few episodes, each consisting of 1,000 steps. The DQN training was performed on 1 x NVIDIA H100 80GB GPU, with convergence time varying between 24 to 48 hours. For training the fusion block, we employed an mT5-large model, instructed with the prompt: "*Translate the source text into the target language.*". The fusion block was trained using a configuration of 3

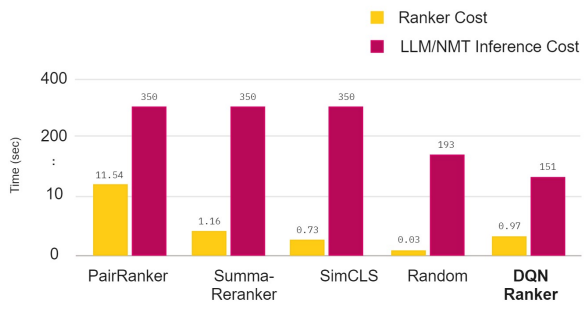

Figure 4: Demonstration of Inference Time Complexity/Cost among Rankers with their Ensemble LLMs Consumption; Reflecting DQN lowest Inference cost and better than all the baseline on 100 Uniformly Sampled test-set.

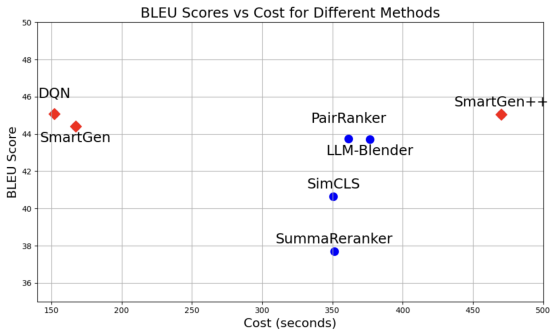

Figure 5: A scatter plot showing the tradeoff of various ensemble methods in terms of quality vs inference cost.

x NVIDIA H100 80GB GPUs. The main hyperparameters used in the experiments are shown in Table 9.

4.5 Results

In our study, we evaluate the performance of our proposed methodologies, namely SmartGen and SmartGen++, against a spectrum of ReRanking baselines and individual expert systems. We also show results in the case where we directly output the best candidate selected by the DQN, which we call DQN(proposed) in Table 3 and 2. This evaluation is conducted across a range of automatic metrics including BLEU, Comet, and chrF++. Our analysis encompasses bidirectional translation tasks, in English and Hindi, whose outcomes are detailed in Table 3 and 2, and English to Chinese and German in Table 4.

SmartGen is Fast: In ranking-based ensembling systems, two types of costs are associated, namely, 1) **Ranking cost**, which refers to the inference time taken by the ranker, and 2) NMT systems **inference cost**, which is the total time taken by Candidate NMT systems to generate the translations. From Figure 4, we observe that: 1) For Ranker Cost, Random Selection of Candidates is the fastest, followed by SimCLS which makes it fastest in baselines, and then SmartGen, which demonstrates better computational time compared to other baselines such as SummaReranker and LLM-Blender. LLM-Blender is the slowest, taking almost 10 times longer than any other baselines. 2) For Inference time of NMT systems, SmartGen takes almost 2.31 times more faster than any of the Rankers and is also 1.26 times faster than selecting random NMT systems, making it the fastest among all the systems.

DQN Ranker is the best Trade-off: We judged our approach against baselines ranking methods for the bidirectional translation task.

Hindi to English: From Table 2 we infer that our approach outperforms best baselines i.e PairRanker by 4.48% on BLEU, 0.60% on Comet and by 2.10% on chrF++, while against simCLS, we beat it by 11.40% on BLEU, 2.24% on comet and 7.83% on chrF++, suggesting that our training objectives were helpful.

English to Hindi: Furthermore, from Table 3 we can infer that on average across datasets DQN Ranker have scores of 34.9 (BLEU), 80.71 (Comet) and 51.65 (chrF++), whereas best baseline PairRankers performance is 35.41 (BLEU), 80.80 (Comet) and 52.66 (chrF++). we can see that our DQN ranker scores are almost comparable with best ranker baseline and with very high computational efficiency which can be seen in the Figure 5. The ranking methods outperformed random selection, except for summaReranker, which performed below the mark compared to baselines and even random selection.

SmartGen is better: We test SmartGen against LLM-Blender (Jiang et al., 2023) as it is the only baseline that uses Fusion Block after ranking. For English to Hindi direction SmartGen outperforms LLM-Blender very slightly on (0.22) BLEU on an average across all the test datasets. Moreover for Hindi to English we see SmartGen, outperforms LLM-Blender by 0.21 (BLEU) and is marginally better in terms of chrF++, on average across all testsets.

4.6 Ablation Studies

Mitigation of BLEU score inaccuracies due to contextual matches: The reward model reduces misleading BLEU evaluations caused by idiomatic

Category	Methods	Private data			IN22-Conv			IN22-Gen			FLORES			WMT-14		
		BLEU	COMET	chrF++	BLEU	COMET	chrF++	BLEU	COMET	chrF++	BLEU	COMET	chrF++	BLEU	COMET	chrF++
LLMs / NMTs	LLaMA-3-8B	44.50	83.45	56.38	34.67	86.81	50.57	26.44	82.39	47.10	32.31	85.45	53.45	27.16	82.83	47.43
	Gemma-2-9B	58.08	88.89	68.73	43.16	90.41	58.42	39.83	87.94	61.28	43.18	88.92	63.22	37.60	87.54	58.73
	Gajendra	31.51	73.44	39.61	24.72	77.99	38.00	23.25	78.90	43.10	25.79	79.24	44.97	21.07	74.89	37.27
	Airavata	45.53	81.66	54.32	37.03	85.30	49.88	30.42	82.39	50.24	36.05	83.74	53.93	30.71	81.24	47.80
	Indictrans2	**60.63**	**89.43**	**70.53**	**46.48**	**90.69**	**60.04**	**46.09**	**89.06**	**65.26**	**51.60**	**90.39**	**68.12**	**43.99**	**88.61**	**61.97**
	m2m100	42.15	81.77	54.42	34.16	84.54	49.09	29.47	82.84	52.35	39.24	86.28	58.73	31.53	82.26	51.59
	mBART	47.17	83.98	59.87	37.19	86.81	52.52	32.95	85.78	55.96	37.90	86.65	58.42	32.58	85.05	54.05
	nllb200	50.22	86.52	61.62	43.79	90.14	57.88	40.41	88.05	61.02	47.05	89.43	64.74	40.95	87.85	59.55
Analysis	Random	48.88	83.89	58.67	38.90	86.73	52.70	34.46	85.27	55.10	39.93	86.49	58.42	34.73	83.99	52.63
	Oracle (BLEU)	68.45	90.06	75.87	54.85	91.38	66.38	51.18	89.13	67.40	56.78	90.31	71.02	49.97	88.77	65.70
Rankers	simCLS	56.03	87.21	65.29	39.88	87.78	53.81	39.55	87.14	59.05	43.74	88.48	61.62	39.41	86.63	57.18
	summaReranker	55.06	86.73	64.76	38.92	87.76	54.02	35.23	85.81	57.30	40.88	87.64	60.79	35.81	85.15	55.58
	PairRanker	58.14	88.43	67.49	44.71	90.30	58.70	41.80	88.22	62.03	47.31	89.49	65.11	41.17	87.96	60.27
	DQN (proposed)	**61.75**	**89.59**	**70.75**	**45.78**	**90.51**	**59.55**	**43.26**	**88.67**	**63.01**	**48.79**	**89.90**	**66.09**	**42.47**	**88.40**	**60.81**
Blenders	LLM-Blender	60.13	89.43	70.01	44.72	90.30	58.70	41.76	88.21	61.90	47.25	89.48	65.09	41.12	87.95	60.26
	SmartGen	58.34	88.47	68.07	44.59	89.85	58.35	42.82	88.29	62.76	48.50	89.71	65.72	41.78	87.96	60.15
	SmartGen++	**62.77**	**90.09**	**70.75**	**45.78**	**90.51**	**59.56**	**43.26**	**88.67**	**62.96**	**48.73**	**89.89**	**66.07**	**42.42**	**88.39**	**60.79**

Table 2: Performance Metrics of Various Models Across Different Datasets in Translation from Hindi to English

Category	Methods	Private data			IN22-Conv			IN22-Gen			FLORES			WMT-14		
		BLEU	COMET	chrF++	BLEU	COMET	chrF++	BLEU	COMET	chrF++	BLEU	COMET	chrF++	BLEU	COMET	chrF++
LLMs / NMTs	LLaMA-3-8B	24.08	71.56	39.94	18.71	76.05	35.93	18.77	68.65	38.61	20.63	69.61	39.53	19.52	70.76	37.37
	Gemma-2-9B	33.26	79.09	49.34	26.64	82.48	43.34	25.39	70.92	45.66	33.35	77.61	52.72	27.44	77.53	45.70
	Gajendra	35.08	80.23	50.05	30.81	83.86	46.57	36.25	79.71	53.68	36.22	80.25	55.39	30.64	80.31	48.40
	Airavata	36.68	81.73	53.12	30.72	84.23	46.74	37.83	**80.51**	55.23	38.12	81.32	57.63	32.15	81.37	50.00
	Indictrans2	**41.96**	**82.92**	**57.46**	**32.96**	**85.10**	**48.93**	**37.90**	80.48	**55.83**	**40.01**	**81.59**	**59.63**	**33.34**	**81.68**	**51.39**
	m2m100	27.34	76.14	43.79	23.75	80.57	40.29	23.86	72.79	43.80	31.81	75.92	50.90	24.67	75.93	42.87
	mBART	25.84	76.00	43.36	25.82	81.30	41.42	25.10	76.53	45.94	26.64	76.51	46.30	24.94	76.91	42.81
	nllb200	30.60	74.63	44.60	29.95	82.61	45.34	16.11	66.73	36.03	26.98	71.09	45.73	22.29	72.85	39.88
Analysis	Random	45.42	83.30	59.41	32.22	84.60	48.07	28.86	77.81	48.85	34.33	80.08	53.75	28.02	79.70	46.35
	Oracle (BLEU)	54.41	85.23	66.73	77.31	93.18	85.97	66.61	85.48	76.81	70.31	87.77	83.66	35.41	81.79	52.82
Rankers	simCLS	41.01	82.07	54.79	32.98	84.84	47.88	30.47	75.95	50.00	35.95	80.60	55.24	30.11	80.58	48.10
	summaReranker	37.00	79.05	50.50	30.12	82.29	44.91	26.15	74.70	45.38	30.28	76.59	49.40	25.96	77.25	43.84
	PairRanker	**42.93**	**82.98**	**57.36**	**33.85**	**85.15**	**49.14**	32.12	**78.62**	**52.29**	**37.33**	**81.13**	**57.11**	**30.78**	**81.14**	47.42
	DQN (proposed)	42.70	82.65	57.03	30.60	83.37	46.22	**33.89**	77.67	51.48	36.60	80.44	55.65	30.71	79.42	**47.91**
Blenders	LLM-Blender	49.11	84.52	62.50	**34.86**	85.67	50.54	30.75	78.69	50.83	36.92	81.41	56.77	29.54	81.02	48.49
	SmartGen	48.13	84.12	61.60	33.71	85.29	49.97	32.29	79.10	51.89	37.33	81.16	56.77	29.86	80.57	48.52
	SmartGen++	**49.45**	**84.67**	**62.83**	34.68	**85.73**	**50.66**	**32.51**	**79.50**	**52.48**	**37.93**	**81.52**	**57.52**	**30.21**	**81.20**	**49.20**

Table 3: Performance Metrics of Various Models Across Different Datasets in Translation from English to Hindi

Category	Methods	en→zh	en→de
Rankers	SimCLS	34.87	36.00
	SummaReranker	34.44	36.48
	PairRanker	35.20	41.10
	DQN (proposed)	**35.41**	**41.16**
Blenders	LLM-Blender	35.50	38.08
	SmartGen (proposed)	**36.24**	**38.32**
	SmartGen++ (proposed)	**36.90**	**38.33**

Table 4: Performance Metrics of Various Models Across Different Datasets in Translation from English to Chinese and English to German ’

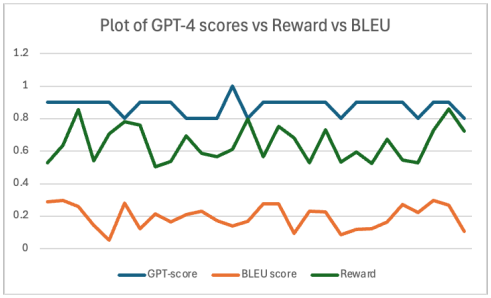

Figure 6: Demonstration of GPT Score *vs.* Reward Score *vs.* BLEU on curated samples having idiomatic expressions, cultural nuances and a mixture of colloquial and formal translations.

expressions and cultural nuances that inadvertently match reference translations, leading to false positives. This is further illustrated in the Figure 6, demonstrating that how BLEU and reward compares to GPT based scoring system we can infer that our model’s ability to correct such inaccuracies and providing more robust rewarding systems which can also be used to judge translation quality as a metrics.

Reward as *judge*: Using *reward* as the judge, we calculated the quality of our translations on Fusion-based systems. From Table 6 we infer that the reward for our approaches SmartGen and SmartGen++ is much higher than LLM-Blender, suggesting that having a training objective where the Ranker and Fusion blocks are inter-dependent results in better final translation quality.

Effect of Competitive Correction Block: In both translation directions, we observed that incorpo-

rating the CCB Block resulted in improved final translations, as evidenced by the metric scores in Table 2 and 3, and also increased the reusability of the generated candidates. Our experiment results in Table 5 compare the performance of different LLMs as the enhancer LLM $\mathcal{G}$ in the CCB block (refer to Alg. 1). We present the average BLEU score for the different LLMs in the two translation directions after utilizing them as $\mathcal{G}$ in the CCB block. Our findings indicate that the CCB block generates alternative translations that surpass the current translation on BLEU score metrics,

LLMs	Eng-Hin	Hin-Eng
LLaMA-3-8B	28.85	42.29
Gajendra	29.10	42.30
Gemma-2-9B	**29.27**	42.30
Airavata	29.25	42.28

Table 5: Demonstration of various LLMs on average enhancement of candidates with CCB block.

Methods	BLEU	Reward
LLM-Blender	33.01	8.59
SmartGen	33.29	**15.35**
SmartGen++	33.83	**15.48**

Table 6: Comparison of average BLEU and average reward for translation quality judgement.

Note: The CCB block, as defined, is independently valuable. While one could use the Q-values from the DQN block for correction block admissibility, employing a *separate* reward model allows compatibility with various candidate selection criteria. Additionally, a user-defined reward model enhances flexibility for human alignment and domain adaptation in the final translation.

Value of K	IN22-Conv	IN22-Gen	WMT14
$K = 1$	30.60	28.16	26.59
$K = 2$	32.82	31.12	28.98
$K = 3$	**33.71**	**32.29**	29.86
$K = 5$	33.51	31.54	30.23
$K = 7$	33.69	31.95	**30.58**

Table 7: Ablation experiment on parameter K.

Ablation on parameter K. We conduct an experiment by varying the values of K, i.e., the number of selected candidates from the DQN block and show the performance of SmartGen in Table 7 for en→hi direction across three benchmarks. Our observations indicate that SmartGen performs best with $K = 3$ for the IN22-Conv and IN22-Gen datasets, while for the WMT14 dataset, K=7 yields the best results. We conjecture that this value of K prevents the Fusion block from being overwhelmed by excessive selections while ensuring that relevant translations are not ignored.

5 Conclusion

In this paper, we identified key limitations in current state-of-the-art ensembling methods for machine translation. We successfully framed the candidate selection problem as a reinforcement learning task, leveraging a Deep Q-Network (DQN) to choose optimal candidates for fusion. This approach reduces inference costs by using selected candidates and improves translation quality through DQN's exploratory capabilities. Additionally, we found that the weakest selected candidate negatively impacts the overall system performance. To address this, we implemented a corrective strategy that enhances translation quality at the cost of increased inference time, which can be of independent interest. Our extensive experiments conducted on benchmark datasets demonstrate that the proposed method yields superior results across various metrics, resulting in SOTA performance.

Limitations

One limitation of our work is the fixed value of K (though small), which we plan to make adaptive in future research to improve performance further. Additionally, the criteria for pushing selected candidates to the CCB could be refined to enhance both translation quality and time efficiency.

Ethics Statement

This work presents a method for ensembling NMT systems and LLMs for machine translation. We evaluate our approach using open-source models trained on public data, which may contain gender and cultural biases. All models and datasets used are properly cited.

References

AI@Meta. 2024. Llama 3 model card.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen,

Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Ves Stoyanov. 2022. Efficient large scale language modeling with mixtures of experts. *Preprint*, arXiv:2112.10684.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *Preprint*, arXiv:2401.10774.

Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. 2024. Iterative translation refinement with large language models. *Preprint*, arXiv:2306.03856.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *Preprint*, arXiv:1911.02116.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

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. *Preprint*, arXiv:2010.11125.

Cong Feng and Jie Zhang. 2018. Reinforcement learning based dynamic model selection for short-term load forecasting. *Preprint*, arXiv:1811.01846.

Jay Gala, Pranjal A. Chitale, Raghavan AK, Varun Gumma, Sumanth Doddapaneni, Aswanth Kumar, Janki Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, Pratyush Kumar, Mitesh M. Khapra, Raj Dabre, and Anoop Kunchukuttan. 2023a. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *Preprint*, arXiv:2305.16307.

Jay Gala, Pranjal A Chitale, A K Raghavan, Varun Gumma, Sumanth Doddapaneni, Aswanth Kumar M, Janki Atul Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, Pratyush Kumar, Mitesh M Khapra, Raj Dabre, and Anoop Kunchukuttan. 2023b. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *Transactions on Machine Learning Research*.

Jay Gala, Thanmay Jayakumar, Jaavid Aktar Husain, Aswanth Kumar M, Mohammed Safi Ur Rahman Khan, Diptesh Kanojia, Ratish Puduppully, Mitesh M. Khapra, Raj Dabre, Rudra Murthy, and Anoop Kunchukuttan. 2024. Airavata: Introducing hindi instruction-tuned llm. *arXiv preprint arXiv: 2401.15006*.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *Preprint*, arXiv:2403.13257.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzman, and Angela Fan. 2021. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Preprint*, arXiv:2106.03193.

IBM Granite Team. 2024. Granite 3.0 language models.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *Preprint*, arXiv:2402.01680.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *Preprint*, arXiv:1512.03385.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Hieu Hoang, Huda Khayrallah, and Marcin Junczys-Dowmunt. 2024a. On-the-fly fusion of large language models and machine translation. *Preprint*, arXiv:2311.08306.

Hieu Hoang, Huda Khayrallah, and Marcin Junczys-Dowmunt. 2024b. 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.

Jianguo Jia, Wen Liang, and Youzhi Liang. 2024. A review of hybrid and ensemble in deep learning for natural language processing. *Preprint*, arXiv:2312.05589.

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.

Vishalteja Kosana, Kiran Teeparthi, Santhosh Madasthu, and Santosh Kumar. 2022. A novel reinforced online model selection using q-learning technique for wind speed prediction. *Sustainable Energy Technologies and Assessments*, 49:101780.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Llm inference serving: Survey of recent advances and opportunities. *Preprint*, arXiv:2407.12391.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Preprint*, arXiv:2001.08210.

Yixin Liu and Pengfei Liu. 2021. Simcls: A simple framework for contrastive learning of abstractive summarization. *Preprint*, arXiv:2106.01890.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *Preprint*, arXiv:2310.02170.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *Preprint*, arXiv:2311.08692.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *Preprint*, arXiv:1312.5602.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and Shyamal Anadkat et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *Preprint*, arXiv:2308.03188.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. 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.

Vikas Raunak, Amr Sharaf, Yiren Wang, Hany Hassan Awadallah, and Arul Menezes. 2023. Leveraging gpt-4 for automatic translation post-editing. *Preprint*, arXiv:2305.14878.

Mathieu Ravaut, Shafiq Joty, and Nancy F. Chen. 2023. Summareranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. *Preprint*, arXiv:2203.06569.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Unbabel's participation in the WMT20 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 911–920, Online. Association for Computational Linguistics.

Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249.

BigScience Workshop: Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, and Matthias Gallé et al. 2023. Bloom: A 176b-parameter open-access multilingual language model. *Preprint*, arXiv:2211.05100.

Naman Shukla, Arinbjörn Kolbeinsson, Lavanya Marla, and Kartik Yellepeddi. 2019. Adaptive model selection framework: An application to airline pricing. *Preprint*, arXiv:1905.08874.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. Learning to summarize from human feedback. *Preprint*, arXiv:2009.01325.

Gemma Team. 2024. Gemma.

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. *Preprint*, arXiv:2207.04672.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *Preprint*, arXiv:2401.10491.

Jiayi Xie, Michael Tashman, John Hoffman, Lee Winikor, and Rouzbeh Gerami. 2021. Online and scalable model selection with multi-armed bandits. *Preprint*, arXiv:2101.10385.

Wenda Xu, Danqing Wang, Liangming Pan, Zhenqiao Song, Markus Freitag, William Yang Wang, and Lei Li. 2023. Instructscore: Explainable text generation evaluation with finegrained feedback. *Preprint*, arXiv:2305.14282.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. *Preprint*, arXiv:2010.11934.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, Yan Yan, Beidi Chen, Guangyu Sun, and Kurt Keutzer. 2024. Llm inference unveiled: Survey and roofline model insights. *Preprint*, arXiv:2402.16363.

Ahmet Üstün, Viraat Aryabumi, Zheng-Xin Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. 2024. Aya model: An instruction finetuned open-access multilingual language model. *Preprint*, arXiv:2402.07827.

A Additional details on datasets and methods

A.1 Training and Evaluation Data Statistics

en↔hi	Private Data	Open-Source Data
Training Data	98K	-
Test Data	-	IN22-Conv-1.5K
	-	IN22Gen-1.02K
	-	WMT-14-2.51K
	2K	
	-	Flores-2.01K
en→de	**Private Data**	**Open-Source Data**
Training Data	-	WMT-19 60K
Test Data	-	WMT-19 1.5K
en→zh	**Private Data**	**Open-Source Data**
Training Data	-	WMT-18 100K
Test Data	-	WMT-18 3.98K

Table 8: Training and Test Data Statistics. The numbers represent pairs of parallel sentences.

A.2 Hyper-parameters used in DQN training

Hyperparameter	Value
Batch Size	128
Steps Batch Size	8
γ	0.99
$\varepsilon_{\text{start}}$	0.9
ε_{end}	0.05
$\varepsilon_{\text{decay}}$	8000
τ	1×10^{-3}
Target Update	100
Memory Size	50000
Learning Rate	4×10^{-5}
Episode	30
Moving Average Window	100

Table 9: Hyperparameters for DQN Training

A.3 Prompt used for the CCB block

We mention here that we have used ChatGPT for language correction and para-phrasing in drafting the paper.

B Additional observations on the Experiments

B.1 Justification of SmartGen performance despite IndicTrans2's superior BLEU score and the efficacy of the picking strategy

As seen in Tables 2 and 3, IndicTrans2 (ITv2) performs very well in many cases compared to other candidates. We perform an experiment on the IN22-Gen, Flores, WMT14 test datasets (English to Hindi direction, where ITv2 is the best performing model), to count the number of times the proposed DQN-based ranker (which is the ranking strategy for SmartGen and SmartGen++) selects ITv2 as one of the chosen K candidate translations. We observe that our proposed strategy selects ITv2 translations most of the times as shown below in Table 10. However, the reason behind the decrease in performance of SmartGen is due to the inclusion of sub-optimal translations in the selected set which can drastically decrease the performance of the final translation (described in Section 2.2 Motivation for Competitive Correction Block, Table 1 in the manuscript). Due to this, the final BLEU score of all the ensembling methods show a degradation in performance in terms of the BLEU score.

```
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that
appropriately completes the request.

### Instruction:
Below are your previous translated texts with their REWARD scores (positive means good translation, negative means
bad translation) in the Example section. Use them to improve your final English translation (similarly it's REWARD score)
for the same Hindi input text.Don't provide any unnecessary words, reward score, only the final English translation text.

Example:


### Input:
Previous English Translation: Today your father's leave will also be.
REWARD Score: 7.5703125

Previous English Translation: Tomorrow, your father will also be cremated.
REWARD Score: 8.5859375

Previous English Translation: your father is going on a vacation tomorrow.
REWARD Score: 8.0703125

Previous English Translation: Tomorrow will also be your father's day off.
REWARD Score: 8.546875

Previous English Translation: Tomorrow your father is on vacation.
REWARD Score: 8.703125

Previous English Translation: Tomorrow your father will also be on leave.
REWARD Score: 10.4921875

Hindi: कल तुम्हारे पिताजी की भी छुट्टी होगी।
```

Figure 7: An example of a detailed prompt that we provide to the CCB block LLM $\mathcal{G}$ along with set of rejected candidate translations and their corresponding scores.

Method	IN22-Gen	FLoRes	WMT14	Average
SimCLS	63.67%	64.69%	64.77%	64.50%
SummaReranker	42.28%	46.53%	47.74%	46.23%
PairRanker	72.16%	72.01%	70.72%	71.33%
DQN-ranker (proposed)	**86.32%**	**94.78%**	**89.30%**	**89.83%**

Table 10: Percentage of ITv2 Selection by the Ranking methods

B.2 Comparison with traditional ensembling methods like static weight sharing and output probability averaging and with MoE

Static weight sharing involves sharing parameters across models, which isn't feasible for independent, pre-trained LLMs since their parameters are

not jointly optimized. Output probability averaging could work theoretically, but it requires a common output vocabulary and comparable probability distributions, which may not apply across LLMs trained for different tasks or domains.

Furthermore, there is a subtle distinction between MoE (Mixture of Expert) and ensembling methods. MoE routes tokens to specialized, trainable sub-networks (experts like FFNs), whereas ensembling combines candidate translations from pre-trained, non-trainable black-box models. We provide a preliminary comparison between ensembling and MoE in Table 11.

Model	BLEU	Number of Parameters
Phi-3.5-MOE-Instruct	28.38	43B
DQN Ranker (Ours)	**33.66**	10B (max)
SmartGen (Ours)	33.60	27B (max)

Table 11: Percentage of ITv2 Selection by the Ranking methods

B.3 Comparison with commercially available translation engines.

We provide a comparison with Google Translate and GPT-4o here for completion in Table 12. We see that our proposed methods perform at par and often better than Google Translate and GPT-4o. Specifically, for IN22-Conv, SmartGen++ outperforms both Google Translate and GPT-4o by a margin of 1.45 BLEU points, whereas for IN22-Gen and WMT14 datasets our methods perform slightly inferior to Google Translate but better than GPT-4o.

Model	IN22-Conv	IN22-Gen	WMT14
Google Translate	33.23	33.83	**33.56**
GPT-4o	32.87	31.85	30.33
DQN-ranker (proposed)	30.60	**33.89**	30.71
SmartGen (proposed)	**33.71**	32.29	29.86
SmartGen++ (proposed)	**34.68**	32.51	30.21

Table 12: Comparison of proposed methods with commercial engines in terms of BLEU.