Who Needs Decoders? Efficient Estimation of Sequence-Level Attributes with Proxies

Anonymous ACL submission

Abstract

Sequence-to-sequence models often require an expensive autoregressive decoding process. However, for some downstream tasks such as out-of-distribution (OOD) detection and resource allocation, the actual decoding output is not needed, just a scalar attribute of this sequence. In such scenarios, where knowing the quality of a system's output to predict poor performance prevails over knowing the output itself, is it possible to bypass the autoregressive decoding? We propose Non-Autoregressive Proxy (NAP) models that can efficiently predict scalar-valued sequence-level attributes. Importantly, NAPs predict these metrics directly from the encodings, avoiding the expensive decoding stage. We consider two sequence tasks: Machine Translation (MT) and Automatic Speech Recognition (ASR). In OOD for MT, NAPs outperform ensembles while being significantly faster. NAPs are also proven capable of predicting metrics such as BERTScore (MT) or word error rate (ASR). For downstream tasks, such as data filtering and resource optimization, NAPs generate performance predictions that outperform predictive uncertainty while being highly inference efficient.

1 Introduction

Autoregressive encoder-decoder models have emerged as the dominant approach for many sequence-to-sequence tasks (Sutskever et al., 2014) and are the state-of-the-art for a range of tasks such as Automatic Speech Recognition (ASR) (Gulati et al., 2020), Machine Translation (MT) (Vaswani et al., 2017; Xue et al., 2021), and Abstractive Text Summarization (Chung et al., 2022; Raffel et al., 2020). However, for many applications, the decoded output sequence is not required, only attributes of the sequence. In out-of-distribution (OOD) detection, only a sequence-level metric such as confidence is required (Hendrycks and Gimpel, 2017; Malinin and Gales, 2021). In selective classification (Geifman and El-Yaniv, 2017; Xia and Bouganis, 2022; El-Yaniv and Wiener, 2010) the output is only needed if the prediction is trusted. Another example is deferral strategies for resource allocation (Li et al., 2015; Teerapittayanon et al., 2016; Viola and Jones, 2001; Xia and Bouganis, 2023; Zhu et al., 2006), where computation is allocated between systems of different complexity. Standard deferral strategy approaches use the predictive uncertainty of a simpler system to decide whether or not to pass it on to a better-performing system of higher complexity (Wang et al., 2022).

All of the examples above require some form of predictive uncertainty metric from the output, which in the case of transformer-based autoregressive models are expensive to obtain (Brown et al., 2020; Chowdhery et al., 2022; Raffel et al., 2020; Wu et al., 2016). Combined with the quadratic cost of self-attention (Vaswani et al., 2017) and autoregressive decoding (equipped with beam-search (Koehn, 2009)), this can limit the application of these systems in real-world settings, such as those that have limited computational resources or require low latency (Viola and Jones, 2001). Furthermore, ensembling generally improves system performance and can be leveraged for useful analysis, such as for robust uncertainty estimation (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017). However, ensembles' memory and inference costs scale linearly with the number of members in the ensemble, making them even more impractical for real-world scenarios. There are methods including Knowledge Distillation (KD) (Ranzato et al., 2016; Hinton et al., 2014) and Ensemble Distribution Distillation (EDD) (Malinin et al., 2020; Fathullah et al., 2021, 2023; Fathullah and Gales, 2022) that distill knowledge from an autoregressive ensemble but this does not circumvent the high costs fundamentally associated with autoregressive generation.

Previous works have investigated adding a second output head explicitly trained to capture a specific metric such as epistemic uncertainty in image segmentation (Landgraf et al., 2023) or the true class probability in image classification (Corbière et al., 2019). The work of (Li et al., 2021) extends this style of approach to ASR by adding a second head to the decoder, to predict token-level decoding errors. Despite its success in providing robust estimates, computing the output uncertainties still requires an expensive autoregressive decoding process. The work of (Coleman et al., 2020) trains an independent proxy model for estimating uncertainties. This method is based on training a much smaller image classification model in an identical manner to the primary model, instead using the uncertainties produced by the small model's outputs to guide the primary one. In the space of autoregressive encoder-decoder models, this approach is still not feasible; the costs of training and decoding persist even for small autoregressive models.

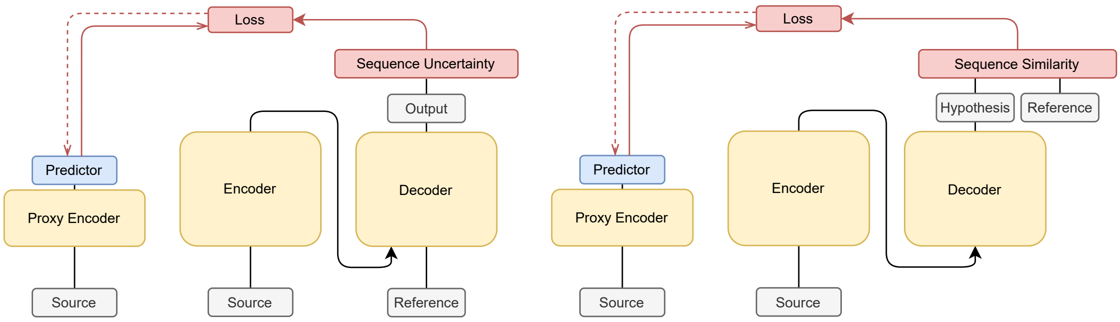
In this paper, we propose *Non-Autoregressive Proxy* (NAP) models that directly estimate sequence-level attributes, bypassing the expensive autoregressive decoding process. When deployed, these lightweight proxy models can be used to robustly predict sequence properties using a fraction of the computational requirements. Our approach is kept general and applicable to any sequence attribute, demonstrating the usefulness of this framework to diverse metrics such as sequence-level predictive uncertainty, BERTScore for MT, and word error rate (WER) for ASR. Investigations into downstream tasks such as out-of-distribution (OOD) detection show that NAPs can outperform an ensemble at a fraction of the inference time. Due to the flexibility of the proposed framework, we also investigate training NAPs on sequence-level performance metrics (BERTScores and WERs), outperforming uncertainty-based approaches on data filtering and resource optimization.

2 Background

There has been a range of work on predicting sequence-level attributes. One common example is estimating uncertainties from the outputs of autoregressive systems (Malinin and Gales, 2021; Notin et al., 2021), where unsupervised token-level uncertainties from some decoding process are combined to form sequence-level estimates. Such sequence-level uncertainties are then used in downstream tasks such as OOD detection (Malinin and Gales, 2021), quality estimation (Fomicheva et al., 2020) and curriculum learning (Zhou et al., 2020).

Previous work has also explored task-specific supervised approaches to confidence/metric estimation. The work of (Gamper et al., 2020) explores training a small independent model to predict the sub-utterance-level word error rate (WER) of a primary ASR model for short-duration audio when the reverberant conditions change. However, the approach is not generalizable to other domains such as MT due to the specific focus on reverberant speech. Other work has also focused on training an error detection module attached to the decoder of some ASR or MT system (Evermann and Woodland, 2000; Koehn, 2009; Kumar and Sarawagi, 2019; Li et al., 2021; Liao and Gales, 2007; Ragni et al., 2018). For example, a typical approach to training the decoder-side error detector is based on token-level error labels from the minimum Levenshtein distance alignment to the ground truth. From these token-level estimates, a sequence-level confidence score can be derived. In ASR where there is often one clear true transcription of the input audio, such an error detection module is appropriate. However, these approaches are inappropriate for MT where multiple translations could all have the same meaning and be considered valid. Such a token-level error detector would flag other valid translations as errorful even when conveying the same information and meaning.

This final example is one of the main motivations behind BERTScore and related approaches (Sellam et al., 2020; Yuan et al., 2021; Zhang et al., 2020; Zhao et al., 2019). BLEU (Papineni et al., 2002; Post, 2018) has long been the main MT evaluation metric for measuring sequence similarity between a translation and a reference using some measure of overlap. However, it suffers from similar issues as (Levenshtein) edit-distance metrics. BERTScore resolves such issues by leveraging bidirectional language models in generating contextual variable-length embeddings for both the translation and reference sequence, computing an automatic sequence similarity score in this embedding space. There has also been a set of work on supervised MT quality estimation (Specia et al., 2020, 2021; Zerva et al., 2022) in which models are trained to estimate the quality (human expert estimated metric) of a translation by making use of the source, the decoded translation and additional token-level probability. However, both the automatic BERTScore and quality metrics require an expensive autoregressive decoding stage to obtain the estimate.



(a) Setup 1: Capturing sequence uncertainties. (b) Setup 2: Capturing sequence similarities.

Figure 1: Our proposed proxy training scheme: A teacher encoder-decoder model trains a proxy encoder student to predict consistent sequence scores using some loss function. In **(a)** we train the proxy to extract sequence uncertainties from a decoder that is fed the reference. In **(b)** we train a proxy to capture sequence-level similarity scores (e.g. BERTScore or WER) from decoded outputs.

3 Non-Autoregressive Proxy

We are interested in the general problem of estimating sequence-level attributes whilst remaining highly inference-efficient. These sequence-level metrics include: (1) information-theoretic uncertainties (Malinin and Gales, 2021); (2) neural-based evaluation scores such as BERTScore (Zhang et al., 2020); and (3) discrete sequence-similarity metrics such as word error rate. The standard approach to obtaining these sequence-level metrics is to run an expensive autoregressive decoding scheme to produce a set of hypotheses. One can either extract sequence attributes directly from this hypothesis set (Malinin and Gales, 2021) or compare them with their corresponding references to obtain a measure of sequence similarity. The aim of this paper is to avoid the costly autoregressive generation stage and instead train an encoder-only, non-autoregressive proxy (NAP) model to imitate the sequence metrics produced by an autoregressive system, using only the source, see Figure 1.

We employ two different setups as shown in Figures 1a and 1b. The aim of the first setup is to train a proxy to directly extract sequence uncertainties when the main model is additionally given the reference sequence. This is in order to teach the proxy model to imitate the uncertainties from the gold reference. The second setup aims to teach the proxy a sequence similarity score when the autoregressive generated hypothesis is compared to the reference. Both setups are highly challenging as the non-autoregressive proxy is tasked with predicting sequence-level metrics from only the source. However, the key feature of the NAP is that it directly predicts these metrics without a decoding scheme (e.g. beam search) and without any reference sequences, allowing the user to extract useful information from large amounts of unlabelled data with little cost. Furthermore, in the first setup of Figure 1a, the proxy also avoids the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2016), by directly training on the teacher-forced (Williams and Zipser, 1989) sequence uncertainties.

In this work, we follow Figure 1a in training a proxy on both single teacher confidence and entropy scores or ensemble mutual information, evaluating its imitation ability and downstream out-of-distribution detection ability. We also follow Figure 1b in training a proxy to predict BERTScores in Machine Translation and WER in Speech Recognition and evaluate the performance of the NAP on a data filtering and resource optimization task.

Loss Function: Sequence-level metrics are represented by single scalar values. Therefore, the proxy student can be trained using any regression loss function. However, unlike standard regression tasks, we seek to learn the relative ordering (rankings) of our scores, as this simplifies the task and is more pertinent for downstream applications such as OOD detection. Therefore, we will mainly opt for the Spearman Rank and Pearson correlation coefficient (SCC & PCC) depending on the specific task considered. Consider a batch of n items with teacher scores $\{s_i\}_{i=1}^{n}$ and corresponding proxy predictions $\{\hat{s}_i\}_{i=1}^{n}$. The Spearman loss function is then defined as:

$$\mathcal{L}_{\text{SCC}} = -\left(1 - \frac{6\sum_i (r(s_i) - r(\hat{s}_i))^2}{n(n^2-1)}\right) \quad (1)$$

where $r(s) \in \{1, 2, \ldots, n\}$ signifies the rank of s. Since the rank operator is discrete and non-differentiable it is not directly applicable to our application. We resort to a differentiable Spearman Rank extension (Blondel et al., 2020) with an open source implementation[1]. Note that unlike its original usage (Blondel et al., 2020), where the system is trained to rank class values for a single instance, we are using this loss to sort single values associated with multiple different items in a batch. We also investigate alternative loss functions such as the root mean squared error (RMSE) and mean absolute error (MAE), see Appendix B.1.

Predictor Design: In order to produce a scalar score from a variable-length encoder-output representation, we make use of a pooling operation. We utilize two options, temporal averaging or multi-head attention with a single trainable query. The encoder vector outputs $\{\boldsymbol{v}_l\}_{l=1}^{L}$ are therefore pooled to form a fixed-size representation $\boldsymbol{v}$ which is fed into a three-layer multi-layer perception (MLP). Furthermore, early exploratory experiments found that a softmax activation is vital for good performance as it can be seen as introducing inductive bias into the estimation of information-theoretic and related metrics. Details on MLP architecture and ablation studies are provided in Appendix B.2.

Proxy Encoder Backbone: By default, the NAP backbone is initialized from the encoder weights of the main encoder-decoder model. Since pre-trained models such as T5 (Raffel et al., 2020) and Whisper (Radford et al., 2022) are released in different sizes, one can utilize smaller architectures to initialize smaller proxies, and train them to predict attributes of larger systems. Appendix B.4 further explores 'mismatched' encoders, e.g. using a RoBERTa NAP to predict the output attributes of a T5 system. Furthermore, all experiments in this paper freeze the encoder backbone and only train the small predictor on top of the NAP encoder. This improves the training speed and memory usage allowing a user to train multiple predictor heads on top of the same backbone, each for a different metric (e.g. estimating sequence-level confidence and BERTScores in the same forward pass). Note that the purpose of our investigations is not to create the best possible NAP model (for example, finetuning the backbone encoder could improve performance at no cost of inference speed). We only seek to demonstrate that this approach is highly flexible and applicable to a range of sequence-level metrics and can provide cheap but useful information for sequence-to-sequence tasks.

[1] `github.com/google-research/fast-soft-sort`

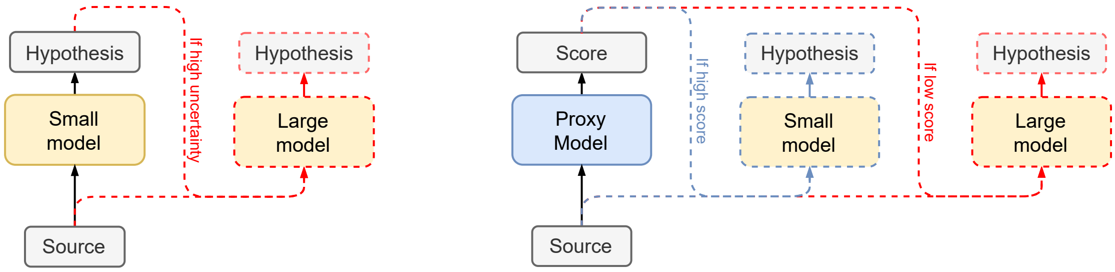
4 Experimental Evaluation

Predicting Uncertainties: We will evaluate the imitation ability of NAP models on various tasks. Following Setup 1, the first set of experiments will focus on the ability of a proxy system to capture sequence-level confidence or entropy from a single T5 transformer (Raffel et al., 2020) fine-tuned on a spoken-language Machine Translation (MT) dataset. We further explore the ability of NAPs to imitate mutual information (epistemic uncertainty (Der Kiureghian and Ditlevsen, 2009; Hora, 1996)) from an ensemble of T5 systems. The performance of the NAPs will then be evaluated by measuring the Spearman Rank correlation between the teacher (under teacher-forcing (Williams and Zipser, 1989)) and the proxy estimates on a range of in-domain (ID) and out-of-domain (OOD) datasets. We also investigate the performance of the proposed NAP on OOD detection.

Predicting BERTScores: Following Setup 2, we also investigate if proxy systems can capture much more complex sequence metrics such as BERTScores (Zhang et al., 2020) from a single T5 in MT. Capturing this metric is especially challenging since the beam-search output of the T5 decoder and corresponding reference will be fed through a language model such as BERT (Devlin et al., 2019) which then computes the final score. The performance will be measured by computing the Spearman Rank between proxy outputs and BERTScores on both ID and OOD datasets. Furthermore, the proxy is compared to sequence-level confidence and entropy scores from the T5 model to see how well they correlate with BERTScores.

The performance of a BERTScore estimating proxy system can also be evaluated on two downstream tasks: *Filtering task* (Li et al., 2021): Given a dataset, we remove the examples with the lowest proxy or highest uncertainty estimate. For good estimates, the filtered subset should display a higher average BERTScore. *Resource optimization task* (Viola and Jones, 2001): Under a fixed resource budget, one seeks to allocate inputs to models of different complexity in order to maximize performance. A well-performing allocation system would achieve higher performance with a smaller budget, see Figure 2.

Predicting WER: Finally, we follow Setup 2 in investigating if a NAP can imitate the sentence-level WER and the total number of errors produced by an ASR system. In this case, we utilize the



(a) Baseline deferral system.

(b) Proxy deferral system.

Figure 2: In the baseline deferral system, the inputs with high uncertainty (under the small model) are fed into the larger model. In the proxy deferral system, model selection is based on the output of an efficient proxy.

pretrained state-of-the-art Whisper (Radford et al., 2022) models on the LibriSpeech corpus (Panayotov et al., 2015). Since the Whisper model is very well-performing, it is able to perfectly decode a large fraction of the dataset, which would cause issues for a rank-based loss such as Spearman. We, therefore, resort to Pearson for these experiments. Note, the corpus-level WER performance of an ASR system is a length-weighted average of the sentence-level WERs. Therefore, we also train NAPs to predict the number of decoding errors in an utterance. Similar to the BERTScore experiments, the performance of NAPs will be evaluated in a similar manner using both filtering and resource optimization tasks.

4.1 Machine Translation

We use the IWSLT 2017 English-to-German training set for finetuning T5 systems on spoken language translation. We generate a three-model ensemble of T5 systems which we use as a stronger baseline for uncertainty estimation. We also investigate if Knowledge Distillation (KD) (Hinton et al., 2014) and Ensemble Distribution Distillation (EDD) (Malinin et al., 2020; Ryabinin et al., 2021) are able to imitate the uncertainties produced by a single or ensemble systems respectively.

We use a range of in-domain and out-of-domain datasets for downstream tasks. These include the Web Inventory Talk (Ted IWSLT 2016; ID), Newstest-19 & 20 news commentary (OOD-1), Khresmoi medical data (OOD-2), MTNT-2019 Reddit text (OOD-3) and KFTT Kyoto-related Wikipedia articles (OOD-3) datasets. All but the latter two datasets are English-to-German, while the final two are English-to-Japanese. Due to the language mismatch, OOD-3 datasets cannot be used to evaluate BERTScore prediction in Section 4.1.2. Setup details are provided in Appendix A.

Table 1 shows the inference time of `iwslt-2017` test set for various models. This demonstrates a primary desideratum of a NAP, the ability to quickly process large amounts of data. For example, a large proxy being 46x faster than a T5 Large model using a beam of $B = 12$ (used in experiments below) and is approximately 138x faster than the three-model ensemble (if run serially). Given the shared architecture between the proxy and primary model encoders, this vast difference in inference time is due to the ability to bypass expensive decoding.

Table 1: Inference time for `iwslt-2017` using Hugging Face (Wolf et al., 2020), with an `NVIDIA A100`. `BERTScore` (BS) measured for the `B = 12` setting.

Model	T5 Model				NAP
	`B = 1`	`B = 4`	`B = 12`	`BS`	
`Small`	41.9s	85.9s	178.6s	67.4	2.7s
`Base`	117.7s	270.3s	537.6s	68.2	5.5s
`Large`	313.7s	583.4s	826.6s	68.6	17.9s

4.1.1 Uncertainties in Machine Translation

We trained NAPs (of different sizes, see Table 1) to predict sequence-level confidence $\mathcal{P}$ or entropy $\mathcal{H}$ (using the conditional approximation described in (Malinin and Gales, 2021)) of a T5 Large model. We also trained NAPs to predict the mutual information $\mathcal{I}$ score produced by an ensemble of fine-tuned T5 Large models. The performance of the proxies is compared to two baseline systems: KD when capturing confidence or entropy of a single model, and EDD in capturing mutual information from an ensemble. The autoregressive distilled baselines will also be of various sizes, see Table 1.

In the case of confidence $\mathcal{P}$ and mutual information scores $\mathcal{I}$, the proxy achieves a better rank ordering of instances for both datasets and at all sizes than the corresponding encoder-decoder student, despite being an order of magnitude faster at in-

Table 2: Spearman Rank correlation of uncertainties when comparing baseline distillation and proxy to the teacher ensemble. Averaged over 3 runs. Standard deviations in the order of ± 1.0.

Model Size	S	B	L	S	B	L	S	B	L
Dataset	**Distillation $\mathcal{P}$**			**Distillation $\mathcal{H}$**			**EDD $\mathcal{I}$**		
`iwslt-2017`	18.7	19.8	20.8	69.4	73.1	74.5	43.7	51.5	55.1
`ted-iwslt-2016`	21.4	21.1	21.8	57.5	59.5	60.6	46.8	47.0	48.0
Dataset	**NAP $\mathcal{P}$**			**NAP $\mathcal{H}$**			**NAP $\mathcal{I}$**		
`iwslt-2017`	39.9	42.6	42.1	40.4	58.8	62.7	53.7	54.3	55.6
`ted-iwslt-2016`	26.2	25.3	25.2	44.8	52.3	53.8	50.0	49.7	51.3

ference (Table 2). Knowledge-distilled models are better at imitating their teacher's $\mathcal{H}$, however, this is not indicative of downstream task performance such as OOD detection, as explored below (Table 3). Note that the NAP here is unique in its ability to predict any scalar sequence metric, whereas KD is unable to mimic mutual information scores.

Finally, we perform downstream OOD detection using confidence, entropy, and MI scores from a T5 Large ensemble, EDD (T5 Large), and Proxy Large. We use `iwslt-2017` as in-domain and measure performance with `AUROC` (50% represents random detection). Results in Table 3 show that in all but one scenario, the uncertainties predicted by the proxy model are best suited for the task, particularly considering inference speeds. Note that overall, the detection performance of a NAP exceeds that of the Deep Ensemble. A potential explanation is that the proxy is directly trained to predict uncertainties while the ensemble estimates uncertainties based on the beam-search decoded outputs (Malinin and Gales, 2021), suffering from exposure bias (Bengio et al., 2015; Ranzato et al., 2016).

4.1.2 BERTScores in Machine Translation

Table 4 directly compares the rank correlation between model confidence/proxy scores and sentence BERTScore performance. We include proxies with attentive pooling as this is a more challenging task. These suggest that training NAPs directly on performance metrics provides a better predictor of a system's performance than using information-theoretic metrics such as confidence and entropy.

Dataset filtering is an alternative approach to evaluating the quality of uncertainty estimates, with emphasis on the highest-performing examples. A well-suited predictor of performance will show a monotonic increase in filtered dataset performance, as harder examples are removed. Figure 3 shows this desired behavior is best achieved with NAPs (equipped with attention pooling) that are directly trained to predict BERTScores of the primary model, in both an ID and OOD dataset. Entropy produced by the model itself is promising on the ID dataset but fails on OOD since the performance does not increase as we filter more examples. Failure to reproduce these trends using uncertainty estimates of the primary model output

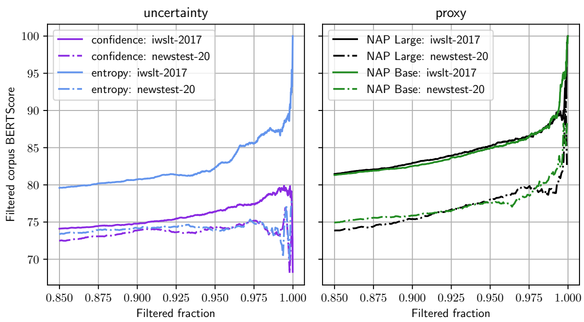


Figure 3: Measuring T5 Large performance on a filtered dataset when removing the worst examples according to some metric.

Table 3: %`AUROC` detection performance of autoregressive and proxy models using various uncertainties. Averaged over 3 runs. Standard deviations in the order of ± 2.0.

Split	**Dataset**	**Deep Ensemble** $\mathcal{P}$	$\mathcal{H}$	$\mathcal{I}$	**EDD** $\mathcal{P}$	$\mathcal{H}$	$\mathcal{I}$	**NAP** $\mathcal{P}$	$\mathcal{H}$	$\mathcal{I}$
OOD-1	`newstest-19`	42.9	53.1	58.5	45.5	54.6	55.7	51.0	53.4	**70.5**
	`newstest-20`	35.9	50.8	63.4	40.6	54.0	61.2	51.6	53.2	**78.1**
OOD-2	`khresmoi-dev`	38.1	51.8	67.2	43.6	57.2	63.4	50.4	51.1	**77.9**
	`khresmoi-test`	39.4	53.8	67.6	44.4	58.5	63.4	55.5	54.9	**81.2**
OOD-3	`mtnt-2019`	66.0	**72.2**	64.4	67.0	72.0	61.9	70.4	72.0	71.4
	`kftt`	31.9	33.8	47.0	32.6	35.8	40.8	27.3	34.8	**54.7**

Table 4: Spearman Rank correlation score between model confidence/entropy and the model BERTScore. The NAPs were trained to predict this score directly. Averaged over 3 runs. Standard deviations are approx. ±2.0.

Split	Dataset	T5 Large $\mathcal{P}$	T5 Large $\mathcal{H}$	NAP S	NAP B	NAP L	NAP w/ Attention S	NAP w/ Attention B	NAP w/ Attention L
ID	iwslt-2017	16.6	41.6	42.0	43.7	44.9	42.5	44.4	**45.6**
	ted-iwslt-2016	11.6	37.3	35.8	36.3	37.3	35.7	37.0	**38.1**
OOD-1	newstest-19	32.9	**39.3**	34.3	36.7	37.6	34.7	37.1	39.2
	newstest-20	34.2	38.3	38.6	38.7	**39.6**	38.9	39.0	39.3
OOD-2	khresmoi-dev	41.4	**45.5**	40.8	43.1	44.7	41.3	42.3	44.8
	khresmoi-test	42.9	46.1	42.0	46.5	45.5	42.3	**47.8**	45.2
	average	29.9	41.3	38.9	40.8	41.6	39.2	41.3	**42.0**

suggests over-confidence (Guo et al., 2017) in low-performing examples.

Figure 4 shows results for resource allocation, where examples are allocated to either a T5 Small or Large based on whether a performance-based related metric is above or below a threshold. Depending on the fraction allocated to the larger system, different levels of overall inference time and performance are achieved. As expected from the

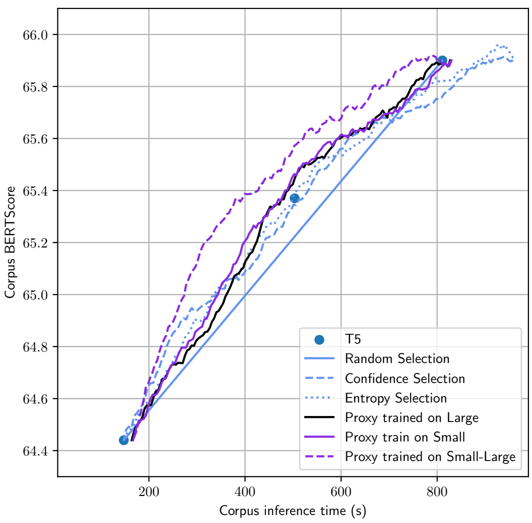


Figure 4: Newstest 20: Measuring BERTScore and inference time when distributing inputs between a T5 Small and Large according to some metric.

dataset filtering results, proxy outputs can better predict instances for which the small model will perform poorly and it does so with a minuscule time cost. By contrast, relying on the output of the small model itself to decide whether the large model is required causes serious delays due to the time spent decoding, delays that the NAP preempts. The best performance was achieved by NAPs trained on the *difference* in BERTScore between the two available systems. The aim of this difference metric is to assign to the large model, examples for which we expect a maximal *increase* in performance. Obtaining such a difference metric using the original models would defeat the whole purpose of resource optimization. Finally, it is possible to be more efficient or better performing than a T5 Base using this deferral system while matching its performance or efficiency respectively.

4.2 WERs in Automatic Speech Recognition

We repeat experiments from Section 4.1.2 using pre-trained Whisper models from Hugging Face (Wolf et al., 2020) on the LibriSpeech corpus (Panayotov et al., 2015). We will by default use greedy decoding as opposed to beam-search since it was found to be robust enough (Radford et al., 2022). Table 5 shows real-time factors (RTFs) demonstrating the inference efficiency of NAPs which do not require a decoder. Compared to greedy ($B = 1$) decoding of Whisper Large-V2, medium and large-sized NAPs are 43 and 33 times faster, respectively.

Table 5: Real-time Factors for `test.other` using Hugging Face, with an `NVIDIA A100`. Corpus WER measured for the `B = 1` setting.

Model	Whisper Models `B = 1`	Whisper Models `B = 5`	Whisper Models `%WER`	NAP
`Small`	0.0480	0.0507	7.62	0.0014
`Medium`	0.0722	0.1075	6.26	0.0024
`Large-V2`	0.1029	0.1625	5.16	0.0031

Table 6 recreates the prior success of proxies in imitating model performance, in this case, sentence-level WER. Furthermore, since Whisper encoders pad all inputs to 30s, including an attention pooling layer can discount the padding and significantly improve performance. The following experiments will use the medium-sized NAP with attention pooling as default since it was found to have similar performance to its larger counterpart on the devel-

Table 6: Pearson correlation between Whisper Large-V2 confidence/entropy and sentence WER. The NAPs were trained to predict WER directly. Standard deviations in the order of ± 1.0.

Dataset	Whisper Large-V2		NAP			NAP w/ Attention		
	$\mathcal{P}$	$\mathcal{H}$	S	M	L	S	M	L
test.clean	13.3	16.8	32.4	36.3	33.9	43.9	**49.7**	47.2
test.other	51.9	60.1	38.0	42.4	43.8	49.8	59.0	**61.5**

opment sets but with a 23% smaller RTF.

Figure 5 shows the filtered corpus WER of test.clean and test.other when removing the worst examples according to model confidence/entropy or proxy outputs. While all are successful on test.other, sequence-level confidence and entropy significantly suffer on test.clean showing increasing corpus WER in certain regions when supposedly removing bad examples, a sign of over-confidence. This failure on test.clean could have been somewhat predicted by the small correlations in Table 6 while NAPs with attention show a significantly better correlation performance with sentence WER.

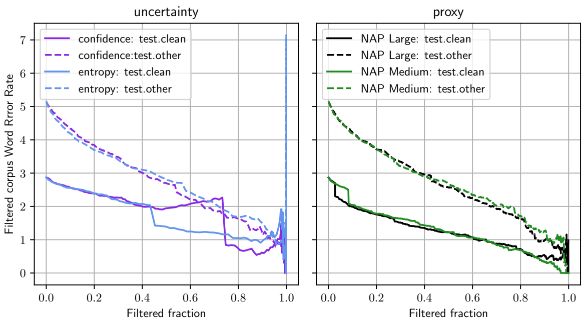


Figure 5: Measuring the corpus WER of Whisper Large-V2 on a filtered dataset when removing the worst examples according to some metric.

Figure 6 shows results for resource allocation, where examples are allocated to a Whisper Small or Large-V2 based on some performance-based related metric. Again, deferral systems using NAPs (with attention) significantly outperform decoder uncertainty-based selection schemes. In fact, the best-performing NAP here was one trained on the number of errors in a transcription, rather than the WER. This is simply because the ordinate in Figure 6 is the corpus WER, rather than the average sentence WER. This is proportional to the error count in the whole corpus, making this a more suitable optimization target. Finally, we note that resource optimization by training a proxy to predict a difference in WER or errors is not presented here. Since the Whisper Small and Large-V2 make the same number of word errors in approximately 75% of examples on the training set, training a proxy on such a sparse label set is difficult.

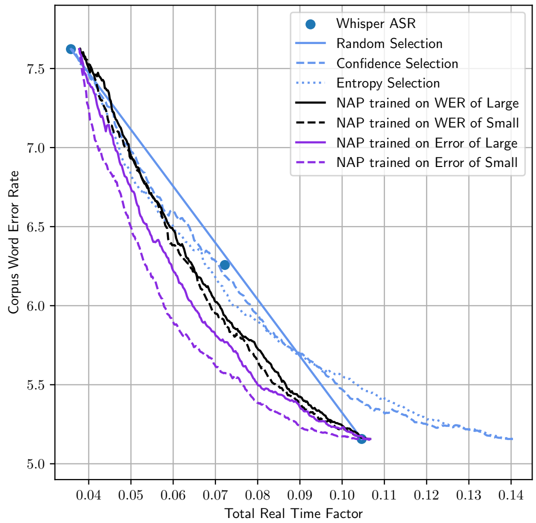


Figure 6: Resource allocation: Measuring corpus WER and RTF when allocating inputs between a Whisper Small and Large-V2 according to a metric.

5 Conclusion

For many downstream sequence-to-sequence tasks, only attributes of the output sequence are needed, and not the output itself. In this paper, we propose a simple efficient framework for directly estimating scalar sequence-level attributes using only the source. While conditioning on the decoding can provide performance gains, this fundamentally defeats the idea behind the inference-efficient Non-Autoregressive Proxies which make them useful and practical for preemptive performance prediction. We show that NAPs can learn information-theoretic uncertainties as well as performance metrics, such as BERTScores for MT or WERs for ASR, in terms of both mimicking attribute score ranks and the impact on downstream tasks. For MT systems they outperform a deep ensemble on OOD detection with an order of magnitude higher inference speed. Furthermore, NAPs are able to outperform predictive uncertainty on downstream tasks such as data filtering and resource optimization on both ASR and MT tasks.

Limitations

This work only investigates using proxies to estimate metrics for encoder-decoder models, and the approach is not directly applicable to decoder-only transformers such as language models unless modifications are made to the proxy framework. Furthermore, the aim of this piece of work is inference-efficient and preemptive prediction of performance using only the source. Future work can extend the work to Autoregressive Proxy models that consider the decoded output as well, which could improve performance at the cost of no longer being efficient and feasible to the downstream tasks considered such as resource allocation.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Conference on Neural Information Processing Systems*.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. 2020. Fast differentiable sorting and ranking. *International Conference on Machine Learning*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuitho Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the iwslt 2017 evaluation campaign. *International Workshop on Spoken Language Translation*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning. *International Conference on Learning Representations*.

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. *Association for Computational Linguistics*.

Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. 2019. Addressing failure prediction by learning model confidence. *Conference on Neural Information Processing Systems*.

Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? does it matter? *Structural safety*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ran El-Yaniv and Yair Wiener. 2010. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*.

Gunnar Evermann and Philip C. Woodland. 2000. Large vocabulary decoding and confidence estimation using word posterior probabilities. *International Conference. on Acoustics, Speech and Signal Processing (ICASSP)*.

Yassir Fathullah and Mark J. F. Gales. 2022. Self-distribution distillation: Efficient uncertainty estimation. *Uncertainty in Artificial Intelligence*.

Yassir Fathullah, Mark J.F. Gales, and Andrey Malinin. 2021. Ensemble distillation approaches for grammatical error correction. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Yassir Fathullah, Guoxuan Xia, and Mark J. F. Gales. 2023. Logit-based ensemble distribution distillation for robust autoregressive sequence uncertainties. *Uncertainty in Artificial Intelligence*.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning (ICML)*.

Hannes Gamper, Dimitra Emmanouilidou, Sebastian Braun, and Ivan J Tashev. 2020. Predicting word error rate for reverberant speech. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. *International Conference on Neural Information Processing Systems*.

Anmol Gulati, Chung-Cheng Chiu, James Qin, Jiahui Yu, Niki Parmar, Ruoming Pang, Shibo Wang, Wei Han, Yonghui Wu, Yu Zhang, and Zhengdong Zhang. 2020. Conformer: Convolution-augmented transformer for speech recognition. *Interspeech*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. *International Conference on Machine Learning*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. *Conference on Neural Information Processing Systems Deep Learning Workshop*.

Stephen C Hora. 1996. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv arXiv:1903.00802*.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *International Conference on Learning Representations (ICLR)*.

Steven Landgraf, Kira Wursthorn, Markus Hillemann, and Markus Ulrich. 2023. Dudes: Deep uncertainty distillation using ensembles for semantic segmentation. *arXiv, arXiv:2303.09843*.

Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. 2015. A convolutional neural network cascade for face detection.

Qiujia Li, David Qiu, Yu Zhang, Bo Li, Yanzhang He, Philip C. Woodland, Liangliang Cao, and Trevor Strohman. 2021. Confidence estimation for attention-based sequence-to-sequence models for speech recognition. *International Conference on Acoustics, Speech and Signal Processing*.

Hank Liao and Mark JF Gales. 2007. Uncertainty decoding for noise robust speech recognition. *Interspeech*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv, arXiv:1907.11692*.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. *International Conference on Learning Representations*.

Andrey Malinin, Bruno Mlodozeniec, and Mark J. F. Gales. 2020. Ensemble distribution distillation. *International Conference on Learning Representations*.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. 2021. Improving black-box optimization in VAE latent space using decoder uncertainty. *Advances in Neural Information Processing Systems*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Association for Computational Linguistics*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. *Conference on Machine Translation: Research Papers*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv, arXiv:2212.04356*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Anton Ragni, Qiujia Li, Mark JF Gales, and Yongqiang Wang. 2018. Confidence estimation and deletion prediction using bidirectional recurrent neural networks. *IEEE Spoken Language Technology Workshop (SLT)*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *International Conference on Learning Representations*.

Max Ryabinin, Andrey Malinin, and Mark J. F. Gales. 2021. Scaling ensemble distribution distillation to many classes with proxy targets. *Conference on Neural Information Processing Systems*.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *Annual Meeting of the Association for Computational Linguistics*.

Lucia Specia, Frédéric Blain, Marina Fomicheva, Erick Fonseca, Vishrav Chaudhary, Francisco Guzmán, and André F. T. Martins. 2020. Findings of the WMT 2020 shared task on quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online. Association for Computational Linguistics.

Lucia Specia, Frédéric Blain, Marina Fomicheva, Chrysoula Zerva, Zhenhao Li, Vishrav Chaudhary, and André F. T. Martins. 2021. Findings of the WMT 2021 shared task on quality estimation. *Proceedings of the Sixth Conference on Machine Translation (WMT)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. *International Conference on Pattern Recognition (ICPR)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiaofang Wang, Dan Kondratyuk, Eric Christiansen, Kris M. Kitani, Yair Movshovitz-Attias, and Elad Eban. 2022. Wisdom of committees: An overlooked approach to faster and more accurate models. *International Conference on Learning Representations (ICLR)*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Guoxuan Xia and Christos-Savvas Bouganis. 2022. Augmenting softmax information for selective classification with out-of-distribution data. *Computer Vision – Asian Conference on Computer Vision*.

Guoxuan Xia and Christos-Savvas Bouganis. 2023. Window-based early-exit cascades for uncertainty estimation: When deep ensembles are more efficient than single models. *arXiv preprint arXiv:2303.08010*.

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. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*.

Chrysoula Zerva, Frédéric Blain, Ricardo Rei, Piyawat Lertvittayakumjorn, José G. C. de Souza, et al. 2022. Findings of the WMT 2022 shared task on quality estimation. *Proceedings of the Seventh Conference on Machine Translation (WMT)*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *Conference on Empirical Methods in Natural Language Processing and Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Yikai Zhou, Baosong Yang, Derek F Wong, Yu Wan, and Lidia S Chao. 2020. Uncertainty-aware curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6934–6944.

Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and S. Avidan. 2006. Fast human detection using a cascade of histograms of oriented gradients. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Table 7: Dataset statistics post tokenization.

Split	Dataset	#Sequences	#Tokens/Sequence src	#Tokens/Sequence ref
Training	iwslt-2017	206,112	29.1	28.5
Validation	iwslt-2017	888	31.9	32.7
Evaluation	iwslt-2017	8,079	27.8	27.5
ID	ted-iwslt-2016	3,662	46.4	54.2
OOD-1	newstest-19	1,997	35.3	39.7
	newstest-20	1,418	49.1	61.6
OOD-2	khresmoi-dev	500	33.7	38.6
	khresmoi-test	1,000	34.7	40.4
OOD-3	mtnt-2019	1,392	26.8	-
	kftt	1,160	40.2	-

A Experimental Configuration

This section will describe the experimental setup of all experiments. Details about datasets, models, and training hyperparameters and evaluation are provided. Hugging Face was used extensively for all experiments in terms of loading various pretrained models, corresponding tokenizers and processed datasets.

A.1 Machine Translation

A.1.1 Datasets

Table 7 reports information about the datasets used for training and evaluation. Note that we use the T5 (Raffel et al., 2020) approach for English-to-German tokenization meaning that we prepend the following prompt to all inputs "translate English to German: " prior to tokenization. We use `iwslt-2017` training set for finetuning T5 systems on spoken language translation and evaluate the corresponding test set. We furthermore use the in-domain (ID) spoken language test set and OOD news commentary (OOD-1), medical data (OOD-2), and a final mixed category of noisy text and Japanese articles (OOD-3) for downstream tasks.

A.1.2 Models

All experiments use the T5 model. In Table 8 we report parameter counts of various models. The T5 is an encoder-decoder model with a language model head which predicts a probability mass function over every token in the output sequence. The proxy model consists of a T5 encoder and a head for predicting uncertainty. The parameter counts below are reported for a proxy with an average pooling layer; an attentive pooling layer would add some parameters. Note, although the embedding layer is expensive parameter-wise, it is extremely fast inference-wise since it is equivalent to a lookup table.

A.1.3 Finetuning T5 Models

All T5 models were finetuned on the IWSLT-2017 (Cettolo et al., 2017) training set and evaluated on several ID and OOD datasets using both SacreBLEU (Post, 2018) and BERTScore (BS) (Zhang et al., 2020), see Table 9. We set the beam size to 12 and used a length penalty of 0.60.

Table 8: Parameter counts of models. NAPs do not use a decoder during inference.

Model	Embeddings	Encoder	Decoder	Head	Total
T5 Small	16.4M	35.3M	41.6M	16.4M	60.5M
NAP Small			-	5.2M	40.6M
T5 Base	24.7M	109.6M	137.9M	24.7M	222.9M
NAP Base			-	11.8M	121.4M
T5 Large	32.9M	334.9M	435.6M	32.9M	737.7M
NAP Large			-	20.9M	355.9M

Table 9: SacreBLEU and BERTScore performance of finetuned T5 models.

Split	Dataset	Small		Base		Large	
		BLEU	BS	BLEU	BS	BLEU	BS
ID	iwslt-2017	32.0	67.4	33.8	68.2	34.3	68.6
	ted-iwslt-2016	30.9	65.2	31.9	65.9	32.3	66.3
OOD-1	newstest-19	37.3	68.0	38.9	69.8	38.9	69.9
	newstest-20	29.4	64.4	30.8	65.4	31.4	65.9
OOD-2	khresmoi-dev	27.1	68.9	29.2	70.7	29.4	70.7
	khresmoi-test	27.4	68.0	30.0	70.2	30.2	70.3

The learning rate was fixed to 0.0001 and the batch size was selected to maximize GPU memory usage on a single NVIDIA A100 SXM4 80GBs. The performance was tracked on the validation set 10 times per epoch and training was terminated when performance stalled for a whole epoch.

The table shows that increasing the size of the T5 model improves performance on the ID datasets. Surprisingly the performance gap between the base and large configuration is very small for most OOD datasets, showing that the base model is particularly effective despite being more than a third of the size.

A.1.4 Training Non-Autoregressive Proxies

We generated scores (uncertainty or BERTScore) from finetuned T5 Large models and used them to train NAP models. We used the smooth and differentiable extension to the Spearman Rank loss function (Blondel et al., 2020) which requires a hyperparameter controlling the level of smoothing. This hyperparameter was set to 0.000001 in all experiments. Similar to the section above, all experiments used a learning rate of 0.0001, maximised batch size and training was stopped when performance did not improve after an epoch.

A.1.5 Estimating Uncertainties in MT

The experiments in this section used the training set of IWSLT-2017 and followed Setup 1, see Figure 1a. The main T5 model produced sequence-level confidence or entropy uncertainty estimates under the reference sequence. The NAP model was then trained to capture this uncertainty. We could have also opted to generate sequence-level uncertainties using Setup 2 (see Figure 1b) but the quality of the uncertainties then depends on the quality of the decoded hypotheses. If we work with unlabelled datasets, we can always revert back to Setup 2 and train our proxy to imitate the uncertainties of the free-running hypotheses.

The performance of the uncertainty estimation NAP was then compared to the main model in two ways. We first computed the Spearman Rank correlation between the NAP output and the main model which was given the reference output. The second and more important evaluation was based on out-of-distribution detection. For this task, we took one in-domain dataset (IWSLT-2017 test set) and compared it with one of the out-of-distribution datasets mentioned above. We sought low uncertainties for the ID dataset and high uncertainties for the OOD dataset. We used the AUROC (Manning and Schütze, 1999) metric for measuring detection performance, where 50% represents a fully random system.

A.1.6 Estimating BERTScores in MT

We decoded a finetuned T5 Large system (with a beam of $B = 12$ and length-penalty of 0.60) on the IWSLT-2017 training set. The decoded outputs were used to compute the BERTScore for each instance, following Setup 2. The NAP was then trained using the exact same hyperparameters as the above section.

Similar to the section above, the outputs of the NAP were first compared with the main model on several unseen datasets. Following, we evaluated the performance of this system on two downstream tasks. First, we took a dataset and filtered out samples with the lowest estimated BERTScore and computed the average BERTScore of the remaining samples. For a well-performing metric, we expect the average BERTScore of the remaining samples to increase monotonically.

Next, we also performed a resource optimization task in which we used the NAP output to decide whether an input should be passed to a smaller (T5 Small) or larger more robust (T5 Large) system. When a proxy output is above a threshold, the input was passed to a smaller system and otherwise to the slower and larger system. The threshold therefore had a large impact on the performance and inference speed of the two model system. By selecting different thresholds, different operating points were achieved. A good system would achieve better performance while deferring as few samples as possible to the slower system.

Furthermore, we also train a NAP to predict the BERTScore difference between the two models in the deferral system. This can be motivated by a simple example: Consider two different models, a smaller $\mathcal{M}_1$ and a larger more robust $\mathcal{M}_2$. Given two different inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ the two models achieve the following BERTScores:

Table 11: Simple example.

	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_2 - \mathcal{M}_1$
$\boldsymbol{x}_1$	0.70	0.90	0.20
$\boldsymbol{x}_2$	0.50	0.40	-0.10

Clearly, the first input is easier to handle since both models achieve higher BERTScores with $\mathcal{M}_2$ being stronger. If we performed an allocation based on the isolated performance of a single model itself, we would give the simpler example $\boldsymbol{x}_1$ to the smaller model $\mathcal{M}_1$ and the harder input $\boldsymbol{x}_2$ to the larger model achieving an average performance of 0.55 BERTScore. However, if we instead perform an allocation based on the performance difference, and refer samples to the stronger model $\mathcal{M}_2$ where it dominates (and vice versa), we would allocate $\boldsymbol{x}_1$ to model $\mathcal{M}_2$ and $\boldsymbol{x}_2$ to model $\mathcal{M}_1$ achieving an average score of 0.70. This shows that an allocation system should focus on the performance difference of the relevant metric.

A.2 Automatic Speech Recognition

A.2.1 Datasets

Table 12 includes information about the LibriSpeech corpus (Panayotov et al., 2015). The number of words per sequence is computed based on the Whisper text normalization scheme. In this task, we do not finetune the ASR models and do not use any out-of-domain datasets. Instead, focus is on the noisy `validation.other` and `test.other` sets.

Table 12: Dataset statistics.

Dataset	#Seq.	#Words per Sequence
`train.clean.100`	28,539	35.0
`train.clean.360`	104,014	34.8
`train.other.500`	148,688	32.7
`valid.clean`	2,703	20.3
`valid.other`	2,864	18.0
`test.clean`	2,620	20.2
`test.other`	2,939	18.0

A.2.2 Models

In Table 10 we report parameter counts of various models. Whisper is an encoder-decoder model with a language model head that predicts a probability mass function over every token in the output sequence. The proxy model consists of a Whisper encoder and a head for predicting uncertainty. The parameter counts below are reported for a NAP with an average pooling layer; an attentive pooling layer would add some parameters.

A.2.3 Training Non-Autoregressive Proxies

We generated sentence-level word error rates (WERs) from the Whisper Large-V2 model using greedy search. While it was found that a beam of $B = 5$ was the best-performing setting in the original work (Radford et al., 2022), this was only achieved using a highly non-standard decoding mechanism; simply using beam search with $B = 5$ actually degrades performance. Therefore, we opted for a simpler setup using greedy search, see Table 13.

Table 10: Parameter counts of models. NAPs do not use a decoder during inference.

Model	Encoder	Decoder	Head	Total
`Whisper Small`	88.1M	153.6M	39.8M	241.7M
`NAP Small`		-	14.2M	102.3M
`Whisper Medium`	307.2M	456.6M	53.1M	763.9M
`NAP Medium`		-	25.2M	332.4M
`Whisper Large-v2`	636.8M	906.5M	66.4M	1543.3M
`NAP Large-v2`		-	39.3M	676.1M

Table 13: Baseline %WER performance with greedy decoding.

Dataset	Small	Medium	Large-v2
`valid.clean`	3.70	2.69	2.48
`valid.other`	7.35	5.46	4.96
`test.clean`	3.45	2.88	2.87
`test.other`	7.62	6.26	5.16

When generating the sentence WERs on the training data of the LibriSpeech corpus, it was found that approximately half of all instances were correctly decoded. This would present problems for a ranking loss and we instead opted to train all NAP models using the Pearson correlation loss. Similar to the section above, all experiments used a learning rate of 0.0001, maximised batch size and training was stopped when performance did not improve after an epoch.

A.3 Estimating WERs in ASR

Following the exact same line of experiments as in Section A.1.6. A NAP was trained to imitate the sentence-level WERs and was evaluated on two downstream tasks, filtering and resource allocation. Note that we train additional proxy systems to capture the total number of errors (instead of the error rate) since this is more aligned with the resource allocation task. The resource allocation was done between the Whisper Large-V2 and Whisper Small models.

We are unable to train a system to capture the error difference for the resource allocation task since training the NAP was unstable. Approximately 74% of all error differences on the training set were 0 making it a highly imbalanced dataset.

B Ablation Studies

We run all of our ablation studies on capturing mutual information of a T5 Large ensemble on the machine translation task. The ensemble consists of three members.

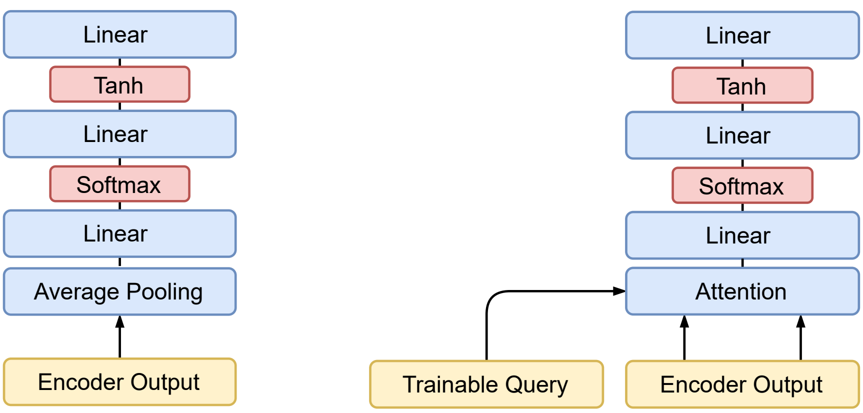
Table 14: NAP OOD performance using MI $\mathcal{I}$.

Dataset	NAP Large			
	`mae`	`rmse`	`pcc`	`scc`
`newstest-19`	67.3	66.9	69.6	**70.5**
`newstest-20`	74.9	73.6	76.0	**78.1**
`khresmoi-dev`	77.9	78.2	**79.1**	77.9
`khresmoi-test`	80.5	81.0	**81.5**	81.2
`mtnt-2019`	69.5	71.4	**73.4**	71.4
`kftt`	50.2	50.2	52.8	**54.7**
`average`	70.1	70.2	72.1	**72.3**

B.1 Choice of Loss Function

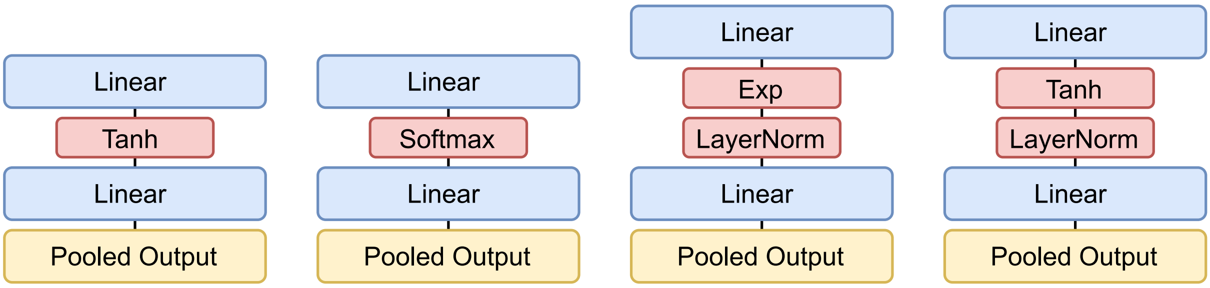
All of the experiments in the main paper used a differentiable Spearman correlation coefficient (`scc`) loss. This section explores alternative loss functions including mean absolute error (`mae`), root mean squared error (`rmse`) and pearson correlation coefficient (`pcc`), see Table 14.

The correlation-based loss functions are consistently better than mean absolute and root mean squared error losses, possibly because the correlation losses do not require accurate prediction of the uncertainties, only their ordering.

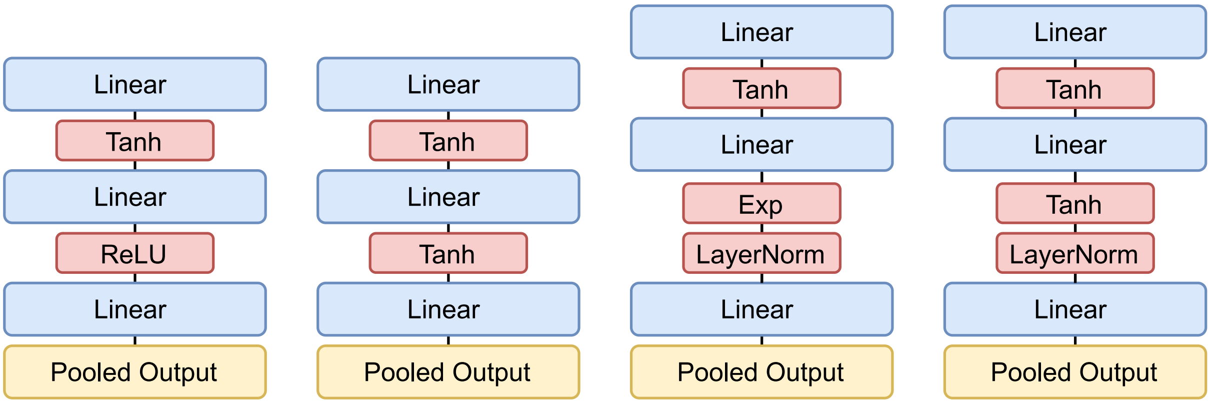


(a) Standard. (b) With attentive pooling.

Figure 7: The standard three-layer network is used on top of a non-autoregressive proxy. When average pooling the encoder output is restrictive, an attention layer is used instead with a trainable query.



(a) From left to right: {`2L Tanh`, `2L SM`, `2L LN-Exp` & `2L LN-Tanh`}.



(b) From left to right: {`3L ReLU`, `3L Tanh`, `3L LN-Exp` & `3L LN-Tanh`}.

Figure 8: Various configurations of proxy heads investigated.

B.2 Predictor Architecture

We also investigate the architecture, and specifically the activations of the MLP that are added on top of the NAP encoder, see Figure 7. In a toy example, we found that a two-layer (with tanh activation) network is better able to predict entropy scores from categorical predictions. This motivates using a three-layer network with a softmax activation to produce 'virtual' probabilities. This section also explores a range of different (parameter-matched) two-layer and three-layer MLPs with various activation functions, see Figure 8.

Table 15 shows the performance of various MLPs (with average pooling) in the out-of-distribution detection task. The two-layer and three-layer MLPs are parameter matched. The final model `3L SM` is the default MLP head used in all experiments. Clearly, the use of a softmax activation is extremely important for achieving the best possible performance.

Table 15: Detection performance of NAPs using MI $\mathcal{I}$.

Split	**Dataset**	**NAP Large**								
		`2L Tanh`	`2L SM`	`2L LN-Exp`	`2L LN-Tanh`	`3L ReLU`	`3L Tanh`	`3L LN-Exp`	`3L LN-Tanh`	`3L SM`
OOD-1	`newstest-19`	56.6	67.7	50.5	48.4	46.4	57.2	59.9	59.7	**70.5**
	`newstest-20`	66.2	75.4	58.6	56.0	47.0	68.2	67.7	63.2	**78.1**
OOD-2	`khresmoi-dev`	55.6	77.5	66.4	49.8	39.2	52.8	65.1	59.1	**77.9**
	`khresmoi-test`	56.0	80.6	67.4	51.8	38.9	53.8	65.2	62.2	**81.2**
OOD-3	`mtnt-2019`	54.1	**71.6**	48.4	52.6	63.4	47.8	61.4	50.6	71.4
	`kftt`	55.2	50.4	55.9	52.0	43.0	**62.0**	58.1	44.8	54.7
	`average`	57.3	70.5	57.9	51.8	46.3	56.9	62.9	56.6	**72.3**

Table 16: Parameter counts and inference time of models on `iwslt-2017`.

Layers	Embeddings	Encoder	Head	Total	Inference Time
Default 24L	32.9M	334.9M	20.9M	355.9M	17.9s
21L	32.9M	289.2M	20.9M	310.1M	15.3s
18L	32.9M	259.4M	20.9M	280.4M	12.7s
15L	32.9M	221.7M	20.9M	242.7M	9.9s
12L	32.9M	184.0M	20.9M	204.9M	7.5s

B.3 Intermediate Outputs of Encoder

It is not necessary to pick the final layer output as the input to the predictor MLP. One can use intermediate layer outputs as well. Previous work has found that using intermediate outputs can even improve upon a task (Hsu et al., 2021; Zhang et al., 2020). Using intermediate layer outputs also leads to faster inference and lower parameter counts, see Table 16.

According to Table 17, the performance of NAPs remains arguably consistent when utilizing intermediate outputs down until the 12th layer, where performance starts dropping. Therefore, it is possible based on this experiment to remove the top 9 layers of the T5 encoder reducing the total parameter count by 32% and inference time by 45% without notably sacrificing performance.

B.4 Mismatched Pretrained Encoders

This section investigates if it is possible to use alternative mismatched encoders as the backbone for a proxy system when predicting sequence-level attributes for the T5 model. We, therefore, investigate replacing the T5 encoder with RoBERTa (Liu et al., 2019), XLM-RoBERTa (Conneau et al., 2020) or the lightweight ALBERT (Lan et al., 2020). See Table 18 for information about the model size and inference time.

The detection performance of alternative backbones such as base RoBERTa and base XLM-RoBERTa are slightly worse but with significantly lower inference times. The large RoBERTa and XLM-RoBERTa are approximately as fast as the T5 Encoder-based proxy but only the latter achieves similar detection performance. The lightweight ALBERT pretrained backbone significantly suffers at this task.

B.5 Decorrelating Epistemic and Aleatoric Uncertainty

Epistemic and aleatoric uncertainties are of different natures. The former is a measure of the lack of knowledge in our model parameters and model choice under the given dataset. As the dataset increases the epistemic uncertainty should decrease. The latter is an intrinsic measure of uncertainty in the data itself which might be caused by noisy data collection methods or labelling errors. Therefore, we propose a new loss function in which we aim to maximise the correlation between the proxy outputs $\{\hat{s}_i\}_i$ and teacher sequence-level epistemic scores $\{s_{ei}\}_i$ whilst also decorrelating its outputs from teacher sequence-level aleatoric scores $\{s_{ai}\}_i$:

$$\mathcal{L}_{\texttt{scc}}\Big(\{\hat{s}_i\}, \{s_{ei}\}\Big) - \alpha \Big|\mathcal{L}_{\texttt{scc}}\Big(\{\hat{s}_i\}, \{s_{ai}\}\Big)\Big| \quad (2)$$

Table 17: Detection performance of NAPs using MI $\mathcal{I}$.

Split	Dataset	NAP Large 24L	21L	18L	15L	12L
OOD-1	newstest-19	70.5	68.7	69.1	68.6	68.1
	newstest-20	78.1	77.0	77.1	76.0	75.4
OOD-2	khresmoi-dev	77.9	78.5	77.2	77.0	76.4
	khresmoi-test	81.2	81.2	80.3	80.2	80.1
OOD-3	mtnt-2019	71.4	70.0	70.9	72.8	70.6
	kftt	54.7	48.9	54.5	56.0	48.8
	average	72.3	70.7	71.5	71.8	69.9

Table 18: Parameter counts and inference time of models on iwslt-2017.

Layers	Embeddings	Encoder	Head	Total	Inference Time
T5 Large Encoder	32.9M	334.9M	20.9M	355.9M	17.9s
RoBERTa Base	39.0M	124.1M	11.8M	135.9M	4.3s
RoBERTa Large	52.0M	354.3M	20.9M	375.3M	17.5s
XLM-RoBERTa Base	192.4M	277.5M	11.8M	289.3M	4.5s
XLM-RoBERTa Large	256.5M	558.8M	20.9M	579.8M	19.2s
ALBERT Base	3.9M	11.1M	11.8M	22.9M	4.8s
ALBERT Large	3.9M	16.6M	20.9M	37.6M	19.4s

Table 19: Detection performance of NAPs using MI $\mathcal{I}$.

Split	Dataset	T5 Encoder	RoBERTa		XLM-RoBERTa		ALBERT	
		Large	Base	Large	Base	Large	Base	Large
OOD-1	newstest-19	**70.5**	64.3	62.6	68.8	69.3	60.8	63.2
	newstest-20	**78.1**	72.0	69.1	76.8	77.4	67.9	68.0
OOD-2	khresmoi-dev	77.9	78.7	77.2	69.2	**80.0**	73.2	71.0
	khresmoi-test	81.2	81.9	78.0	72.1	**83.0**	75.8	74.2
OOD-3	mtnt-2019	**71.4**	61.6	62.1	61.7	61.6	63.5	68.3
	kftt	54.7	61.7	62.1	**62.6**	62.3	51.4	43.0
	average	**72.3**	70.1	68.5	68.6	**72.3**	65.4	64.6

where α controls the level of decorrelation. Table 20 shows that by using this style of loss function, the proxy can be made to perform significantly better. The base model $\alpha = 0.0$ already outperforms a deep ensemble at detection, and furthermore, setting $\alpha = 1.0$ shows even better overall performance.

Table 20: NAP OOD performance using MI $\mathcal{I}$.

Dataset	NAP Large			
	$\alpha = 0.0$	0.5	1.0	2.0
newstest-19	70.5	**76.1**	76.0	75.3
newstest-20	78.1	85.9	**86.3**	84.0
khresmoi-dev	77.9	86.1	**88.0**	83.5
khresmoi-test	81.2	86.8	**87.7**	83.3
mtnt-2019	**71.4**	61.7	57.3	51.1
kftt	54.7	70.2	76.5	**77.9**
average	72.3	77.8	**78.6**	75.9

C Deferral Between Whisper Systems

This section will provide a brief look into the inference speed or performance gains that can be achieved by using a deferral system. Following the results in Figure 6, Table 21 shows the WER or RTF of various deferral systems (allocating between Whisper Small and Large-V2) when operating at the Whisper Medium RTF or WER respectively. The best deferral system, a NAP trained on the number of errors of Whisper Small, reduces WER by 11% while matching the inference speed of Whisper Medium. For the same WER performance, this system can reduce the RTF by 26%.

Table 21: Columns show (1) corpus WER performance of various deferral systems operating at the same RTF as Whipser Medium and (2) the RTF when operating at the same WER as Whipser Medium.

Selection	WER	RTF
Whisper Medium	6.26	0.0722
Confidence Selection	6.19	0.0707
Entropy Selection	6.09	0.0677
NAP trained on WER of Large	5.94	0.0645
NAP trained on WER of Small	5.89	0.0640
NAP trained on Error of Large	5.77	0.0596
NAP trained on Error of Small	5.57	0.0534