Frustratingly Easy Model Ensemble for Abstractive Summarization

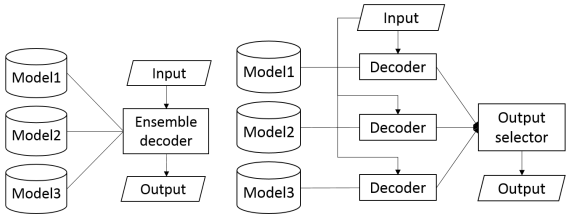
Hayato Kobayashi
Yahoo Japan Corporation / RIKEN AIP
hakobaya@yahoo-corp.jp

Abstract

Ensemble methods, which combine multiple models at decoding time, are now widely known to be effective for text-generation tasks. However, they generally increase computational costs, and thus, there have been many studies on compressing or distilling ensemble models. In this paper, we propose an alternative, simple but effective unsupervised ensemble method, *post-ensemble*, that combines multiple models by selecting a majority-like output in post-processing. We theoretically prove that our method is closely related to kernel density estimation based on the von Mises-Fisher kernel. Experimental results on a news-headline-generation task show that the proposed method performs better than the current ensemble methods.

1 Introduction

Recent success in deep learning, especially encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015), has dramatically improved the performance of various text-generation tasks, such as translation (Johnson et al., 2017), summarization (Ayana et al., 2017), question-answering (Choi et al., 2017), and dialogue response generation (Dhingra et al., 2017). In these studies on neural text generation, it has been known that a model-ensemble method, which predicts output text by averaging multiple text-generation models at decoding time, is effective even for text-generation tasks, and many state-of-the-art results have been obtained with ensemble models. However, an ensemble method has a clear drawback in that it increases computational costs, i.e., the increase in time as the number of models increases, since it averages the word-prediction probabilities of all models in each decoding step. Therefore, there have been many studies on model compression or distillation for ensemble methods, each of which has successfully shrunk an ensemble model (Hinton et al., 2015; Chebotar and Waters, 2016; Kuncoro et al., 2016; Kim and Rush, 2016; Stahlberg and Byrne, 2017; Freitag et al., 2017).



(a) Runtime-ensemble (b) Post-ensemble

Figure 1: Flow charts of current runtime-ensemble (a) and our proposed post-ensemble (b).

In this paper, we propose an alternative method for model ensemble inspired by the majority vote in classification tasks (Littlestone and Warmuth, 1994). Majority vote is a method that selects the most frequent label from the predicted labels of multiple classifiers in post-processing. Similarly, our method involves selecting a majority-like output from the generated outputs of multiple text-generation models in post-processing as in Fig. 1(b), instead of averaging models at decoding time as in Fig. 1(a). The difference between a classification task and text-generation task is that we need to consider a sequence of labels for each model output in a text-generation task, although we consider only one label in a classification task. This means a majority output may not exist since each output will be basically different from other outputs, which are generated from different models. To overcome this problem, we propose an unsupervised method for selecting a majority-like output close to the other outputs by using cosine similarity. The idea is quite simple, but experiments showed that our method is more effective than the current ensemble methods.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4165–4176
Brussels, Belgium, October 31 - November 4, 2018.

Our work can open up a new direction for two research communities: model ensemble and hypotheses reranking (see Sec. 6 for detailed descriptions of the related studies). For the first, we suggest a new category of ensemble algorithms that corresponds to the output selection in classification tasks. In classification tasks, there are roughly three approaches for model ensemble: model selection in preprocessing, model average at runtime, and output selection in post-processing. In text generation studies, model selection by cross-validation and model average with an ensemble decoder have been frequently used, but output selection as typified by majority vote has received less attention because of the fact that a majority output may not exist, as described above. Therefore, there is enough room to study this direction in the future. Since our algorithm in this paper is quite simple, we expect that more sophisticated methods can improve the results even over our approach.

For the hypotheses reranking research community, we suggest a new category of reranking tasks, where we need to select the best output from the generated outputs of multiple models, instead of the N-best hypotheses of a single model. Hypotheses reranking for a text-generation model is related to our task, but in this case, a reranking method based on a language model is frequently used and is basically enough to correct the scoring of a beam search with a single model (Chen et al., 2006; Vaswani et al., 2013; Luong and Popescu-Belis, 2016) since the purpose is to obtain a fluent output and remove erroneous outputs, assuming the model can generate good outputs. A clear difference between our task and the reranking task is that we should consider all outputs to decide the goodness of an output because a fluent output is not always appropriate in this task. This is similar to extractive summarization (Erkan and Radev, 2004) but is significantly different from our task in that our output candidates have almost the same meaning.

Our contributions in this paper are as follows.

- We propose a simple, fast, and effective method for unsupervised ensembles of text generation models, where (i) the implementation is "frustratingly easy" without any modification of model code (Alg. 1), (ii) the computational time is enough for practical use (Sec. 5.3), i.e., an ensemble time of 3.7 ms per sentence against a decoding time of 44 ms, and (iii) the performance is competitive with the state-of-the-art results (Sec. 5.2), i.e., our method (ensemble of 32 models) for 37.52 ROUGE-1 against the state-of-the-art method (single model) for 37.27 ROUGE-1 on a news-headline-generation task.
- We prove that our method is an approximation of finding the maximum density point by kernel density estimation based on the von Mises-Fisher kernel (Sec. 4). In addition, we derive a formula of the error bound of this approximation.
- We will release the 128 prepared models used in this paper (Sec. 5.1), each of which was trained for more than two days, as a new dataset to improve ensemble methods.

2 Preliminaries

In Sec. 2.1, we briefly explain an encoder-decoder model for text generation, and in Sec. 2.2, we discuss the current ensemble methods for combining multiple text generation models at decoding time.

2.1 Encoder-Decoder Model

An encoder-decoder model is a conditional language generation model, which can learn rules for generating an appropriate output sequence corresponding to an input sequence by using the statistics of many correct pairs of input and output sequences, e.g., news articles and their headlines. When training this model, we calculate a conditional likelihood,

$$p(y \mid x) = \prod_{t=1}^{T-1} p(y_{t+1} \mid y_{\leq t}, x), \qquad (1)$$

with respect to each pair (x, y) of input sequence $x = x_1 \cdots x_S$ and output sequence $y = y_1 \cdots y_T$, where $y_{\leq t} = y_1 \cdots y_t$, and maximize its mean. The model $p(y \mid x)$ in Eq. (1) is achieved by combining two recurrent neural networks, called an *encoder* and *decoder*. The former reads an input sequence x to recognize its content, and the latter predicts an output sequence y corresponding to the content.

After training, we can obtain an output y from an input x by using a learned model $p(y \mid x)$. Since the calculation of an optimal output is clearly intractable, most studies used a beam search, which is a greedy search algorithm that keeps a limited number of best partial solutions, whose size is called the *beam size*. Formally, a set of best partial solutions of beam size b at step t is represented as $Y^b_{\leq t}$, which is recursively defined

as the top b elements with respect to $p(y_{\leq t} \mid x)$, where $y_{\leq t} \in Y^b_{\leq t-1} \times Y$. The Y is a set of available elements for y_i, or a target dictionary. Let start and goal meta symbols be <s> and </s>, respectively. A beam search procedure starts from $Y_{\leq 0} = \{\text{<s>}\}$ and finishes when the last symbols of all elements in $Y^b_{\leq t}$ are the goal element </s> or when its length t becomes larger than some threshold.

2.2 Runtime-Ensemble

In a text-generation task, model ensemble is a method of predicting a next word by averaging the word-prediction probabilities of multiple text-generation models at decoding time. Fig. 1(a) shows a flow chart of the current ensemble methods, which we call *runtime-ensemble* to distinguish them from our method. There are mainly two variants of runtime-ensemble using arithmetic mean $\overline{p}_{\text{a}}$ and geometric mean $\overline{p}_{\text{g}}$, which are defined as

$$\overline{p}_{\text{a}}(y_{\leq t} \mid x) = \frac{1}{|M|} \sum_{p \in M} p(y_{\leq t} \mid x), \quad (2)$$

$$\overline{p}_{\text{g}}(y_{\leq t} \mid x) = \left(\prod_{p \in M} p(y_{\leq t} \mid x) \right)^{\frac{1}{|M|}}, \quad (3)$$

where M is a set of learned models. We call the former `EnsSum` and the latter `EnsMul`. Although there have been no comparative experiments, `EnsMul` is usually used since most decoding programs keep $\log p$ and calculating $\sum_{p \in M} \log p$ is enough to obtain the top b words with respect to $\overline{p}_{\text{g}}$ for a beam search procedure.

3 Post-Ensemble

Our alternative ensemble method combines multiple text-generation models by selecting a majority-like output close to the other outputs, which is calculated with a similarity function such as cosine similarity. We call this method *post-ensemble* since it is executed in post-processing, i.e., after a decoding process. Fig. 1(b) shows a flow chart of post-ensemble, and Alg. 1 shows its algorithm. When our method receives an input x, a normal decoder calculates the output s of each model p from the input in parallel (lines 2–4), and the output selector selects the majority-like output y from all outputs (lines 6–9). In line 7, we calculate the score c of each output s by using a similarity function K, where $K(s, s')$ represents the similarity between s and s'. A higher score means that the output s is in a denser part in the output space since the score c means the average similarity in other outputs.

Input: Input text x, set M of learned models, and similarity function K, such as cos.
Output: Output prediction y.

1 $S \leftarrow \emptyset$;
2 **foreach** $p \in M$ **do**
3 $\quad s \leftarrow$ output of model p for input x;
4 $\quad S \leftarrow S \cup \{s\}$;
5 $C \leftarrow \{\}$; `// as a hash map`
6 **foreach** $s \in S$ **do**
7 $\quad c \leftarrow \frac{1}{|S|} \sum_{s' \in S} K(s, s')$;
8 $\quad C[s] \leftarrow c$;
9 $y = \text{argmax}_{s \in S} C[s]$;
10 **return** y

Algorithm 1: Post-ensemble procedure.

The post-ensemble procedure has two main advantages compared with the current runtime-ensemble procedure. One is that we do not need to develop an ensemble decoder by modifying a decoding program on a deep learning framework. The concept of runtime-ensemble is simple, but its implementation is not that simple in recent sophisticated open source software. For example, we need to modify about 100 lines to add an ensemble feature to the decoding program of an open source neural machine translator, OpenNMT[1], which requires understanding the overall mechanism of the software. The other advantage is that we can easily parallelize decoding processes in our method since each output can be calculated by using a single model. If we have a server program for text generation, we can improve its performance with all our machine resources (ideally) by assigning a server to each model and allowing the output selector to communicate with it.

One drawback of our method is that its expressive power is basically the same as that of each single model. However, this alternatively means that the lower bound of the quality of each output is guaranteed with the worst case of the outputs of single models, while the current runtime-ensemble method can perform worse than each single model for the worst case input. Furthermore, experiments showed our post-ensemble method is more effective than the current runtime-ensemble methods.

4 Theoretical Analysis

In this section, we prove that when $K(s, s') = \cos(s, s')$, Alg. 1 is an approximation of find-

[1] https://github.com/OpenNMT/OpenNMT-py

ing the maximum density point by kernel density estimation based on the von Mises-Fisher kernel. First, we briefly explain kernel density estimation and how to apply it to our method in Sec. 4.1. Then, we introduce the von Mises-Fisher kernel used in this analysis and later experiments in Sec. 4.2. Finally, we prove a theorem that guarantees the approximation error in Sec. 4.3.

4.1 Kernel Density Estimation

Kernel density estimation is a non-parametric method for estimating the probability density function of a random variable. Let $(X_1, \cdots, X_n)$ be an independent and identically distributed (i.i.d.) sample that was drawn from a distribution with an unknown density function f. The kernel density estimator based on the sample is defined as

$$\tilde{f}(X) = \frac{1}{n}\sum_{i=1}^{n} K(X, X_i). \quad (4)$$

Using an appropriate kernel such as the Gaussian kernel, this estimator $\tilde{f}$ converges to the true density f, and it can be proved that there is no non-parametric estimator that converges faster than this kernel density estimator (Wahba, 1975).

Here, let us consider our outputs $(s_1, \cdots, s_n)$, which correspond to S in Alg. 1. They are generated from text generation models $(p_1, \cdots, p_n)$, which correspond to M in Alg. 1. We assume that these models are trained with randomly initialized parameters $(\theta_1, \cdots, \theta_n)$, each of which includes a random seed for the optimizer, and the other settings are deterministic. In this case, we can construct a function $F : P \to O$ that maps the parameter space P onto the output space O. In other words, if each parameter θ_i is an i.i.d. random variable, the corresponding output $s_i = F(\theta_i)$ is also an i.i.d. random variable. Therefore, Eq. (4) can be directly used for line 7 in Alg. 1.

Our method can be regarded as a heuristic approach based on the characteristics of our encoder-decoder model, where there are many local solutions for optimization. We expect that our method can be applied to other models on the basis of a theoretical study (Kawaguchi, 2016), that showed that deep neural networks can have many local optima, but there are no poor local optima (formally, every local minimum of deep neural networks is a global minimum under a certain condition). We do not consider this direction since theoretical justification is beyond our scope.

4.2 von Mises-Fisher Kernel

The von Mises-Fisher kernel (Hall et al., 1987) is a natural extension of the Gaussian kernel to a unit hypersphere. This kernel is especially useful for directional or angular statistics, so it is expected to be compatible with the cosine similarity frequently used in natural language processing. The definition is

$$K_{\text{vmf}}(s, s') = C_q(\kappa)\exp(\kappa\cos(s, s')), \quad (5)$$

where κ is a smoothing factor called the *concentration parameter*, and cos is a cosine similarity, i.e., $\cos(s, s') = \frac{s \cdot s'}{||s||_2||s'||_2}$. $C_q(\kappa)$ is the normalization constant, which is defined as

$$C_q(\kappa) = \left(\kappa^{\frac{q-1}{2}}\right) \Big/ \left((2\pi)^{\frac{q+1}{2}} \mathcal{I}_{\frac{q-1}{2}}(\kappa)\right), \quad (6)$$

where $\mathcal{I}_v$ is the modified Bessel function of the first kind at order v, and q is the dimension of directional data (angular expression of data).

In the experiments described later, we implemented Alg. 1 with this kernel by using the log-sum-exp trick (Nielsen and Sun, 2016) to avoid overflow/underflow problems since $\operatorname{argmax}\sum\exp(x) = \operatorname{argmax}\log\sum\exp(x)$. In addition, we used Garcia-Portugues's rule (Garcia-Portugues, 2013) to adjust the concentration parameter $\kappa = \hat{h}^{-2}$, defined as

$$\hat{h} = \left(\frac{4\pi^{\frac{1}{2}}\mathcal{I}_{\frac{q-1}{2}}(\tilde{\kappa})^2}{\tilde{\kappa}^{\frac{q+1}{2}}\left(2q\mathcal{I}_{\frac{q+1}{2}}(2\tilde{\kappa}) + (2+q)\tilde{\kappa}\mathcal{I}_{\frac{q+3}{2}}(2\tilde{\kappa})\right)n} \right)^{\frac{1}{4+q}} \quad (7)$$

where $\tilde{\kappa}$ is an approximation of κ derived from the maximum likelihood estimation (Sra, 2012), defined as $\tilde{\kappa} = \frac{\tilde{\mu}(q-\tilde{\mu})}{1-\tilde{\mu}^2}$, where $\tilde{\mu}$ is the sample mean of the directional data in a unit hypersphere.

4.3 Approximation Error Analysis

We prove an approximation error bound of Alg. 1 when $K(s, s') = \cos(s, s')$, as shown in the following theorem.

Theorem 1. The output y of Alg. 1 with $K(s, s') = \cos(s, s')$ is equivalent to the maximization of the first order Taylor series approximation $\tilde{p}$ of the kernel density estimator p based on the von Mises-Fisher kernel, i.e.,

$$\tilde{p}(y) = \max_{s \in S} \tilde{p}(s), \quad (8)$$

where the approximation error R^* of the output y with respect to the true density estimator p, i.e,. $R^* = \max_{s \in S} p(s) - p(y)$, is bounded by

$$R^* \leq C_q(\kappa)\kappa^2 \exp(\kappa)(\sigma^2 + \mu^2), \quad (9)$$

where $\mu = \max_{s \in S} \mathbb{E}_{s'}[\cos(s, s')]$, and $\sigma^2 = \max_{s \in S} \mathbb{V}_{s'}[\cos(s, s')]$.

Proof sketch. Eq. (8) can be obtained by using the first order Taylor series approximation at 0 of $\exp(x)$, i.e., $\exp(x) \approx 1 + x$, and the nature of argmax, i,e., $\mathrm{argmax}(1 + \kappa x) = \mathrm{argmax}\, x$. Eq. (9) can be derived by the Lagrange error bound $\tilde{R}(x)$ for $\exp(x) \approx 1 + x$, where $x = \kappa \cos(s, s')$, and $-\kappa \leq x \leq \kappa$, as

$$\tilde{R}(x) = \frac{\max_{x'} \exp(x')}{2!} x^2 \leq \frac{\exp(\kappa)}{2} x^2. \quad (10)$$

See Appendix A for the complete proof. □

This theorem implies that the approximation error becomes smaller as κ becomes smaller. Since κ is the concentration parameter, the shape of the density estimation will be smooth when κ is small, while it will be a peak when κ is large. This means that, when κ is large, the density estimation is almost the same as the majority vote. Therefore, we can naturally choose a small value for κ for our purpose. In fact, the concentration parameter was set as $\kappa = 0.69$ by using Garcia-Portugues's rule in our experiments. The normalization constant using κ was calculated as $C_q(\kappa) = 0.14$, and the average values of μ and σ with respect to the set S of output candidates were $\mathbb{E}_S[\sigma] = 0.30$ and $\mathbb{E}_S[\mu] = 0.78$, respectively. In this case, the theoretical average approximation error was calculated as $\mathbb{E}_S[R^*] \leq 0.093 = 0.14 \times 0.69^2 \times \exp(0.69) \times (0.78^2 + 0.30^2)$. This is quite small in view of the approximation error for a probability. In addition, the actual average approximation error can be much smaller, and it was about 1.95×10^{-7} in our experiments. The accuracy defined by the rate at which the approximate maximum is the true maximum, i.e., $p(y) = \max_{s \in S} p(s)$, was 96.36%. Detail on the settings of our experiments will be given in the next section.

5 Experiments

We first explain the basic settings of our experiments in Sec. 5.1 and report a comparative experiment and analysis on the news-headline-generation task in Sec. 5.2. Then, we discuss the change in some of the settings to conduct an experiment by changing the number of models and the settings of model preparation in Sec. 5.3 and Sec. 5.4, respectively.

5.1 Basic Settings

Dataset: We used a well-known dataset `Gigaword` of a news-headline-generation task, which was prepared by Rush et al. (2015). This dataset has been extensively used in recent studies on abstractive summarization (Takase et al., 2016; Chopra et al., 2016; Kiyono et al., 2017; Zhou et al., 2017; Suzuki and Nagata, 2017; Cao et al., 2018). The `Gigaword` dataset was created from the English Gigaword corpus[2], in which the input is the first sentence in a news article, and the output is the headline of the article. The training, validation, and test sets included 3.8M, 189K, and 2K sentences, respectively. The preprocessed data are publicly available[3]. The dataset is also used to train official pretrained models of OpenNMT[4].

Model and Training: We basically used the default PyTorch implementation of OpenNMT[5] on June 11, 2017 throughout our experiments, but the unidirectional long short-term memory (LSTM) for the encoder was replaced with a bidirectional one to obtain nearly state-of-the-art results. The basic settings are as follows. Our model consisted of a bidirectional LSTM for the encoder and a stacked LSTM with input feeding for the decoder. These LSTMs had two layers with 500-dimensional hidden layers whose dropout rates were 0.3, and their input vectors were created by a 500-dimensional word-embedding layer.

The model was trained with a stochastic gradient descent method with a learning rate of 1.0, where the mini-batch size was set to 64. The learning process ended in 13 epochs, decaying the learning rate with a decay factor of 0.5 in each epoch after 8 epochs. These training settings are the same as the training of the official pretrained models of OpenNMT, and we confirmed that these settings performed better than training with Adam (Kingma and Ba, 2014) in our preliminary experiments. We prepared 10 learned models by random initialization for the ensemble methods in our experiments.

Decoding and Evaluation: When decoding input sequences, we used a beam-search algorithm with a beam width of 5. The maximum size of decoded sequences was 100. The generated unknown token <unk> was replaced by the source word with the highest attention weight.

To evaluate decoded sequences, we calculated ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004), mainly used in the headline-generation-task (Rush et al., 2015). ROUGE-1 and ROUGE-

[2]https://catalog.ldc.upenn.edu/LDC2012T21
[3]https://github.com/harvardnlp/sent-summary
[4]http://opennmt.net/Models/
[5]SHA: c13a558767cbc19b612968eb4d01a1f26d5df688

2 are the co-occurrence rates of unigrams and bigrams, respectively, between a generated headline and its reference. ROUGE-L is the rate of the longest common subsequence between them to the reference length. We used a Python wrapper of the ROUGE-1.5.5.pl script[6] and took the average value of 10 times, each of which used 10 models.

Compared Methods: We compared the following methods. `Single` is a baseline with a single model. `EnsSum` and `EnsMul` are strong baselines with runtime-ensemble. `MaxLik` and `MajVote` are weak baselines with naive post-processing. `LexRank` and `LMRank` are simple unsupervised methods from two other related tasks, extractive summarization and hypotheses reranking, respectively. `PostCosE` and `PostCosB` are variants of the proposed method with post-ensemble. `PostVmfE` and `PostVmfB` are true density estimators corresponding to `PostCosE` and `PostCosB`, respectively. Their descriptions are listed below in detail.

- `Single` decodes an output by using the best single model with respect to the word level accuracy on a validation set.
- `EnsSum` and `EnsMul` decode an output averaging multiple models with Eq. (2) and Eq. (3), respectively.
- `MaxLik` selects an output with the maximum likelihood, which is calculated by the corresponding model p in Alg. 1, from candidate outputs generated by multiple models.
- `MajVote` selects an output by majority vote based on exact matching, i.e., $y = \mathrm{argmax}_{s \in S} |\{s' \in S \mid s = s'\}|$.
- `LexRank` selects an output with the LexRank algorithm (Erkan and Radev, 2004). We used a Python implementation[7], where a graph is constructed on the basis of cosine similarities between the tf-idf vectors (without stop-words) of candidate outputs. The idf weights are calculated from the training set.
- `LMRank` selects an output that maximizes the likelihood of a (non-conditional) language model p_{LM}, i.e., $y = \mathrm{argmax}_{s \in S} p_{\mathrm{LM}}(s)$, as in (Vaswani et al., 2013). We used the decoder part of the encoder-decoder model described in Sec. 5.1, which was trained with both source and target sentences in the training set. This allows this model to learn the fluency in both normal and headline-like sentences.
- `PostCosE` and `PostVmfE` select an output on the basis of Alg. 1 with the cosine similarity i.e., $K(s, s') = \cos(s, s')$, and the von Mises-Fisher kernel, i.e., $K(s, s') = K_{\mathrm{vmf}}(s, s')$ in Eq. (5), respectively. The feature of each output is the average of pretrained 300-dimensional word embeddings[8].
- `PostCosB` and `PostVmfB` are variants of `PostCosE` and `PostVmfE` with simple bag-of-words features (sparse vectors), respectively.

In addition, we used the following measurements for analysis. `MaxRef` represents the upper bound for the performance of our method. `Mean`, `Max`, and `Min` represent the performance statistics of the single models.

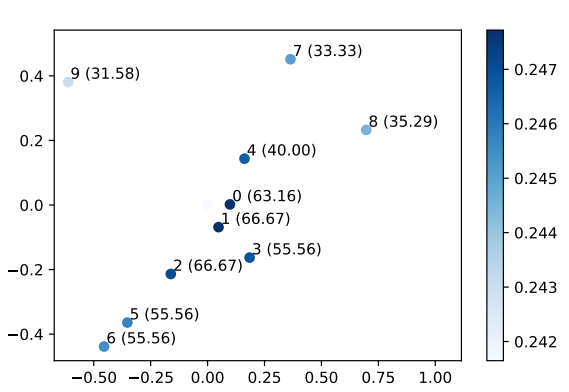
- `MaxRef` selects the best output with respect to ROUGE-1, which is calculated by using the references in the test set.
- `Mean`, `Max`, and `Min` are the mean, maximum, and minimum of the (non-ensemble) ROUGE-1 values for the 10 models, respectively. The difference between `Single` and `Max` is that the former uses the validation set, while the latter uses the test set.

5.2 Main Results

We conducted a comparative experiment on the news-headline-generation task to verify the effectiveness of our post-ensemble method compared with the current runtime-ensemble methods.

Tab. 1 shows the experimental results for the `Gigaword` dataset, including the results of our method with 32 models and other previous results. First of all, we can see that the variant of our post-ensemble method, `PostCosB`, clearly outperformed the runtime-ensemble methods (strong baselines), `EnsSum` and `EnsMul`, and the other baselines. The differences between our best method `PostCosB` and the best baseline `EnsSum` were all statistically significant on the basis of a one-tailed, paired t-test ($p < 0.05$). Comparing with the recent results of Cao et al. (2018) obtained with open information extraction and dependency parse technologies and the other previous results[9], our method with 32 models also performed better, although the algorithm of our

[6] https://github.com/pltrdy/files2rouge
[7] https://github.com/wikibusiness/lexrank
[8] https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[9] We did not present the results of Raffel et al. (2017) since Kiyono et al. (2017) pointed out that their settings are different from the previous studies.



0: **interpol asks** members **to** devise **rules for policing** ...
1: **interpol asks** members **to** devise **rules for policing**
2: **interpol asks** members **to** devise **rules for policing** at ...
3: **interpol asks** members **to** devise **rules** on **policing**
4: **interpol asks** members **to** devise **rules** and procedures ...
5: **interpol** seeks **rules for policing** of **global** level
6: **interpol** seeks **rules for policing** at **global** level
7: **interpol asks** members **to** act against wanted fugitives
8: **interpol asks** members **to** help fight fugitives
9: **interpol asks for** legal status **for** red corner notices

Figure 2: Left scatter-plot shows two-dimensional visualization of outputs generated from 10 models on basis of multi-dimensional scaling (Cox and Cox, 2008), and right list shows their contents. Each point in plot represents sentence embedding of corresponding output, and label indicates model ID and ROUGE-1, i.e., "ID (ROUGE)." Color intensity means score of kernel density estimation of `PostCosE` (see right color bar), and outputs are sorted by scores. Reference and input are as follows. Each bold word in above list means co-occurrence with reference below.

Reference: *interpol asks world govts to make rules for global policing*
Input: *top interpol officers on wednesday asked its members to devise rules and procedures for policing at the global level and providing legal status to red corner notices against wanted fugitives .*

	R-1	R-2	R-L
`Single`	35.57	17.47	33.19
`EnsSum`	36.55	18.48	34.24
`EnsMul`	36.47	18.35	34.16
`MaxLik`	35.04	17.37	32.80
`MajVote`	35.97	18.09	33.67
`LexRank`	36.03	17.64	33.60
`LMRank`	35.07	17.12	32.61
`PostCosE`	37.02	18.46	34.54
`PostVmfE`	**37.06**	18.53	34.60
`PostCosB`	37.05	**18.59**	**34.61**
`PostVmfB`	37.02	18.58	34.59
`MaxRef`*	45.40	24.61	42.09
`Mean`*	35.57	17.48	33.19
`Max`*	36.03	17.83	33.63
`Min`*	35.00	17.08	32.67
`PostCosE` (32 models)	37.52	18.55	34.86
`PostCosB` (32 models)	37.48	18.76	34.99
(Rush et al., 2015)♯	31.00	12.65	28.34
(Takase et al., 2016)♯	31.64	12.94	28.54
(Chopra et al., 2016)♯	33.78	15.96	31.15
(Kiyono et al., 2017)♯	35.79	17.84	33.34
(Zhou et al., 2017)♯	36.15	17.54	33.63
(Suzuki and Nagata, 2017)♯	36.30	17.31	33.88
(Cao et al., 2018)♯	37.27	17.65	34.24

Table 1: F-measure ROUGE-1, ROUGE-2, and ROUGE-L scores (%) for news-headline-generation task. Bold and underlined scores represent best scores for ensembles of 10 models and all methods excluding measurements with "∗," respectively. Results with "♯" are taken from corresponding papers.

method is quite simple. Note that our method can be easily applied to their models to improve their results. Looking at the row for `MaxRef`, the results imply that our post-ensemble method still has room for improvement without any changes to model structure. Although we also conducted an experiment by changing the settings of model preparation, the results had a similar tendency to those of the main results (see Sec. 5.4).

Fig. 2 illustrates how our method worked with kernel density estimation (see the figure caption for detailed descriptions). The left scatter-plot shows a two-dimensional visualization of 10 outputs generated from the 10 models and the estimated densities (represented by color intensity in the right bar). Looking at the center part of the plot, we can see that there are many good outputs with high ROUGE-1 results (noted in brackets in the plot) in the dense part. The right list shows the corresponding outputs of the points in the left plot, where these outputs are sorted by the estimated density. The list shows that our method successfully obtained the majority-like output (model ID of 0) in the dense part of the output space, although there are no exact match outputs. Looking at the bottom part of the list, we can see that our method clearly eliminated unpromising outputs (model ID of 7, 8, and 9) with less information, since they are scattered.

5.3 Effect of Number of Models

We compared the effect of changing the number of models on the performance of our best method `PostCosB` and the best baseline `EnsSum`. We pre-

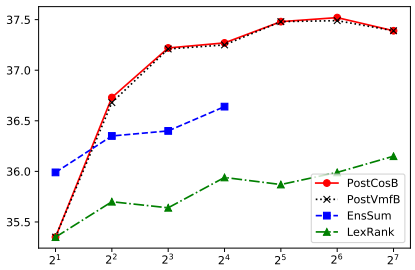


Figure 3: F-measure ROUGE-1 performance (%) vs. number of models for news-headline-generation task. X-axis is log scale (2^1–2^7).

	Random	Self	Hetero	Bagging
Single	35.57	35.34	35.67	34.87
EnsSum	36.55	35.46	36.42	36.25
EnsMul	36.47	35.22	36.49	35.80
MaxLik	35.04	34.21	34.86	34.95
MajVote	35.97	35.49	35.89	35.22
LexRank	36.03	33.57	35.91	35.72
LMRank	35.07	33.47	34.71	34.39
PostCosE	37.02	**35.91**	36.57	**36.89**
PostVmfE	**37.06**	35.72	36.69	36.84
PostCosB	37.05	35.74	**36.76**	36.78
PostVmfB	37.02	35.75	36.75	36.81
MaxRef*	45.40	43.37	45.32	46.44
Mean*	35.57	34.43	35.28	34.85
Max*	36.03	35.34	35.96	35.31
Min*	35.00	33.43	34.49	34.36

Table 2: F-measure ROUGE-1 scores (%) of random-ensemble (Random), self-ensemble (Self), hetero-ensemble (Hetero), and bagging-ensemble (Bagging) for news-headline-generation task. Bold scores represent best scores for all methods excluding measurements with "$*$."

pared 128 models, in which each training took more than two days. The ROUGE-1 performance was measured by varying the number of models, i.e., 2, 4, $\cdots$, 128.

Fig. 3 shows the performance of our best method PostCosB, the corresponding true estimator PostVmfB, the best baseline EnsMul, and the most widely-used baseline LexRank versus the number of models. Note that we could not calculate the results of EnsMul for more than 16 models due to out of memory errors. The figure shows that PostCosB performed better than EnsMul even for these 16 models. We obtained a 37.48 ROUGE-1 score with 32 models, which was better than the state-of-the-art results in Tab. 1, but the performance seems to be saturated with more than 32 models. Looking at PostCosB and PostVmfB, we can see that the performances are almost the same, which also supports our theoretical analysis in Sec. 4. LexRank did not work well even though the number of models was large.

The complexity of the post-ensemble procedure in Alg. 1 is $O(\beta\nu+\delta\nu^2)$, where ν is the number of models, δ is the dimension of the output space, and β is the number of operations of the beam-search. We can reduce it to $O(\beta+\delta\nu)$ by simply parallelizing lines 2–4 and 6–8 in Alg. 1 without any change to the model code on the deep learning framework. Since the operations of β includes all matrix calculations in the model, we can basically assume $\beta \gg \delta\nu$. In fact, the actual calculation times of PostCosE and PostCosB with a naive implementation in Python were 0.0097 and 0.0037 seconds per sentence when $\nu = 32$, respectively. They are enough for practical use in comparison with the decoding speeds of 0.044 (GPU) and 0.49 (CPU) seconds per sentence. In addition, the complexity of the runtime-ensemble is $O(\beta\nu)$, which cannot be parallelized without modifying more than a hundred lines of code after understanding a whole system.

5.4 Effect of Model Preparation

We conducted experiments to verify the effect of changing the model preparation on post-ensemble performance. In addition to random initialization (*random-ensemble*), we address three variations of model preparation: *self-ensemble*, *hetero-ensemble*, and *bagging-ensemble*. The first one, self-ensemble, is a method of extracting models from "checkpoints" saved in each epoch in a training. We prepared the models of self-ensemble by using 10 checkpoints from 4–13 epochs. The second one, hetero-ensemble, is a method of training models varying in model structure. We prepared 10 models for hetero-ensemble, consisting of 8 models prepared by changing the number of layers in the LSTM encoder/decoder in $\{2, 3\}$, the size of LSTM hidden states in $\{250, 500\}$, and the size of word embedding in $\{250, 500\}$, and two models prepared by replacing the bidirectional encoder with a unidirectional encoder and a bidirectional encoder with a different merge action, i.e., summation instead of concatenation. The third one, bagging-ensemble, is a method of training models by bagging of training data. We randomly extracted 80% of the training data 10 times and prepared 10 models for bagging-ensemble. We used the same dictionary of the original data for these models, since the runtime-ensemble methods, EnsSum and EnsMul, failed to average the models with different dictionaries.

Note that the outputs for self-ensemble and hetero-ensemble cannot be regarded as i.i.d samples, but we believe the basic idea can be practically applied.

Tab. 2 shows the F-measure ROUGE-1 scores for the `Gigaword` dataset of the above three variations, self-, hetero-, and bagging-ensembles, as well as random-ensemble. The table indicates that all variants of our post-ensemble method performed better than the current runtime-ensemble methods, `EnsSum` and `EnsMul`, for all variations of model preparation. Looking at the row for `PostCosE`, random-ensemble was the most effective, while self-ensemble was the worst, as expected. Bagging-ensemble was relatively effective for post-ensemble according to the relative improvement from `Single`, despite the fact that we trained the models with 80% of the training data. Hetero-ensemble performed worse than random-ensemble for these settings, but we expect that if the model structure can be randomly chosen, hetero-ensemble will perform better.

6 Related Work

Distillation techniques for an ensemble of multiple models have been widely studied (Kuncoro et al., 2016; Chebotar and Waters, 2016; Kim and Rush, 2016; Freitag et al., 2017; Stahlberg and Byrne, 2017), especially after a study by Hinton et al. (2015). Kuncoro et al. (2016) and Chebotar and Waters (2016) studied distillation techniques for ensembles of multiple dependency parsers and speech recognition models, respectively. There are several ensemble methods for ensembles of machine translation models (Kim and Rush, 2016; Freitag et al., 2017; Stahlberg and Byrne, 2017). For example, Stahlberg and Byrne (2017) proposed a method of unfolding an ensemble of multiple translation models into a single large model once and shrinking it down to a small one. However, all methods require extra implementation on a deep-learning framework, and it is not easy to apply them to other models. Our post-ensemble method does not require such coding skills. In addition, since the predictions of post-ensemble can be regarded as a teacher model, these distillation techniques should be combined with a teacher model based on post-ensemble.

Hypotheses reranking of language generation has been extensively studied, but most studies focused on discriminative training using costly annotated data (Shen et al., 2004; White and Rajkumar, 2009; Duh et al., 2010; Kim and Mooney, 2013; Mizumoto and Matsumoto, 2016). The main stream of our focused unsupervised approach was a reranking method based on a language model (Chen et al., 2006; Vaswani et al., 2013; Luong and Popescu-Belis, 2016), and other approaches include reranking methods based on key phrase extraction (Boudin and Morin, 2013), dependency analysis (Hasan et al., 2010), and search results (Peng et al., 2013). All of the above described studies were not used for model ensemble. Tomeh et al. (2013) used an ensemble learning, but the purpose was to improve the performance of the reranking model for hypotheses reranking of a single model. Li et al. (2009), which work is the most related one, proposed a reranking algorithm for model ensemble. However, their method was constructed to perform at decoding time, so it can be regarded as runtime-ensemble.

The term "frustratingly easy" in this paper is borrowed from "frustratingly easy" papers (Daumé III, 2007; Daumé III et al., 2010; Tommasi and Caputo, 2013; Sun et al., 2016; Kim et al., 2016).

7 Conclusion

We proposed a simple but effective model-ensemble method, called *post-ensemble*, for abstractive-summarization models, i.e., encoder-decoder models. We verified the effectiveness of our method on the news-headline-generation task.

We will release the 128 prepared models used in this paper[10], each of which was trained for more than two days, as a new dataset for improving ensemble methods. For example, future research includes applying learning-to-rank regarding all outputs as features, conducting active learning to select a new model setting online, and developing boosting-like-ensemble based on the bagging of training data.

Acknowledgments

The author would like to thank the anonymous reviewers of all versions of this paper for sparing their valuable time and leaving many insightful comments.

[10] https://research-lab.yahoo.co.jp/en/software/

References

Ayana, Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, and Mao-Song Sun. 2017. Recent Advances on Neural Headline Generation. *Journal of Computer Science and Technology*, 32(4):768–784.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Florian Boudin and Emmanuel Morin. 2013. Keyphrase Extraction for N-best Reranking in Multi-Sentence Compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 298–305. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the Original: Fact Aware Neural Abstractive Summarization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 4784–4791. AAAI Press.

Yevgen Chebotar and Austin Waters. 2016. Distilling Knowledge from Ensembles of Neural Networks for Speech Recognition. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association*, pages 3439–3443.

Yi Chen, Ming Zhou, and Shilong Wang. 2006. Reranking Answers for Definitional QA Using Language Modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 1081–1088. Association for Computational Linguistics.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-Fine Question Answering for Long Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 209–220. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 93–98. Association for Computational Linguistics.

Michael A. A. Cox and Trevor F. Cox. 2008. *Multidimensional Scaling*. Springer Berlin Heidelberg.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2017)*, pages 256–263. Association for Computational Linguistics.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly Easy Semi-Supervised Domain Adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 484–495. Association for Computational Linguistics.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 375–383. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22(1):457–479.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble Distillation for Neural Machine Translation. *CoRR*, abs/1702.01802.

Eduardo Garcia-Portugues. 2013. Exact risk improvement of bandwidth selectors for kernel density estimation with directional data. *Electronic Journal of Statistics*, 7:1655–1685.

Peter Hall, G. S. Watson, and Javier Cabrera. 1987. Kernel Density Estimation with Spherical Data. *Biometrika*, 74(4):751–762.

Sasa Hasan, Oliver Bender, and Hermann Ney. 2010. Reranking Translation Hypotheses Using Structural Properties. In *Proceedings of the EACL'06 Workshop on Learning Structured Information in Natural Language Applications*, pages 41–48. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Kenji Kawaguchi. 2016. Deep Learning without Poor Local Minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 586–594. Curran Associates, Inc.

Joohyun Kim and Raymond Mooney. 2013. Adapting Discriminative Reranking to Grounded Language Learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 218–227. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1317–1327. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly Easy Neural Domain Adaptation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 387–396. The COLING 2016 Organizing Committee.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. 2017. Source-side Prediction for Neural Headline Generation . *CoRR*, abs/1712.08302.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1744–1753. Association for Computational Linguistics.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. Collaborative Decoding: Partial Hypothesis Re-ranking Using Translation Consensus between Decoders. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, pages 585–592. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL Workshop on Text Summarization Branches Out*.

Nick Littlestone and Manfred K. Warmuth. 1994. The Weighted Majority Algorithm. *Information and Computation*, 108(2):212–261.

Ngoc Quang Luong and Andrei Popescu-Belis. 2016. A Contextual Language Model to Improve Machine Translation of Pronouns by Re-ranking Translation Hypotheses. *Baltic Journal of Modern Computing*, 4(2):292–304.

Tomoya Mizumoto and Yuji Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1133–1138. Association for Computational Linguistics.

Frank Nielsen and Ke Sun. 2016. Guaranteed Bounds on Information-Theoretic Measures of Univariate Mixtures Using Piecewise Log-Sum-Exp Inequalities. *Entropy*, 18(12).

F. Peng, S. Roy, B. Shahshahani, and F. Beaufays. 2013. Search results based N-best hypothesis rescoring with maximum entropy classification. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 422–427.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and Linear-Time Attention by Enforcing Monotonic Alignments. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 2837–2846.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389. Association for Computational Linguistics.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative Reranking for Machine Translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*, pages 177–184. Association for Computational Linguistics.

Suvrit Sra. 2012. A Short Note on Parameter Approximation for Von Mises-Fisher Distributions: And a Fast Implementation of I s (x). *Computational Statistics*, 27(1):177–190.

Felix Stahlberg and Bill Byrne. 2017. Unfolding and Shrinking Neural Machine Translation Ensembles. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 1946–1956. Association for Computational Linguistics.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 2058–2065. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112. Curran Associates, Inc.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 291–297. Association for Computational Linguistics.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural Headline Generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1054–1059. Association for Computational Linguistics.

Nadi Tomeh, Nizar Habash, Ryan Roth, Noura Farra, Pradeep Dasigi, and Mona Diab. 2013. Reranking with Linguistic and Semantic Features for Arabic Optical Character Recognition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 549–555. Association for Computational Linguistics.

Tatiana Tommasi and Barbara Caputo. 2013. Frustratingly Easy NBNN Domain Adaptation. In *IEEE International Conference on Computer Vision (ICCV 2013)*, pages 897–904.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1387–1392. Association for Computational Linguistics.

Grace Wahba. 1975. Optimal Convergence Properties of Variable Knot, Kernel, and Orthogonal Series Methods for Density Estimation. *The Annals of Statistics*, 3(1):15–29.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron Reranking for CCG Realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 410–419. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104. Association for Computational Linguistics.