Pointing the Unknown Words

Caglar Gulcehre
Université de Montréal

Sungjin Ahn
Université de Montréal

Ramesh Nallapati
IBM T.J. Watson Research

Bowen Zhou
IBM T.J. Watson Research

Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Abstract

The problem of rare and unknown words is an important issue that can potentially effect the performance of many NLP systems, including traditional count-based and deep learning models. We propose a novel way to deal with the rare and unseen words for the neural network models using attention. Our model uses two softmax layers in order to predict the next word in conditional language models: one predicts the location of a word in the source sentence, and the other predicts a word in the shortlist vocabulary. At each timestep, the decision of which softmax layer to use is adaptively made by an MLP which is conditioned on the context. We motivate this work from a psychological evidence that humans naturally have a tendency to point towards objects in the context or the environment when the name of an object is not known. Using our proposed model, we observe improvements on two tasks, neural machine translation on the Europarl English to French parallel corpora and text summarization on the Gigaword dataset.

1 Introduction

Words are the basic input/output units in most of the NLP systems, and thus the ability to cover a large number of words is a key to building a robust NLP system. However, considering that (i) the number of all words in a language including named entities is very large and that (ii) language itself is an evolving system (people create new words), this can be a challenging problem.

A common approach followed by the recent neural network based NLP systems is to use a softmax output layer where each of the output dimension corresponds to a word in a predefined word-shortlist. Because computing high dimensional softmax is computationally expensive, in practice the shortlist is limited to have only top-K most frequent words in the training corpus. All other words are then replaced by a special word, called the *unknown word (UNK)*.

The shortlist approach has two fundamental problems. The first problem, which is known as the *rare word* problem, is that some of the words in the shortlist occur less frequently in the training set and thus are difficult to learn a good representation, resulting in poor performance. Second, it is obvious that we can lose some important information by mapping different words to a single dummy token UNK. Even if we have a very large shortlist including all unique words in the training set, it does not necessarily improve the test performance, because there still exists a chance to see an unknown word at test time. This is known as the *unknown word* problem. In addition, increasing the shortlist size mostly leads to increasing rare words due to Zipf's Law.

These two problems are particularly critical in language understanding tasks such as factoid question answering (Bordes et al., 2015) where the words that we are interested in are often named entities which are usually unknown or rare words.

In a similar situation, where we have a limited information on how to call an object of interest, it seems that humans (and also some primates) have an efficient behavioral mechanism of drawing attention to the object: *pointing* (Matthews et al., 2012). Pointing makes it possible to deliver information and to associate context to a particular object without knowing how to call it. In particular, human infants use pointing as a fundamental communication tool (Tomasello et al., 2007).

In this paper, inspired by the pointing behavior of humans and recent advances in the atten-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 140–149,
Berlin, Germany, August 7-12, 2016.

tion mechanism (Bahdanau et al., 2014) and the pointer networks (Vinyals et al., 2015), we propose a novel method to deal with the rare or unknown word problem. The basic idea is that we can see many NLP problems as a task of predicting target text given context text, where some of the target words appear in the context as well. We observe that in this case we can make the model *learn to point* a word in the context and copy it to the target text, as well as *when to point*. For example, in machine translation, we can see the source sentence as the context, and the target sentence as what we need to predict. In Figure 1, we show an example depiction of how words can be copied from source to target in machine translation. Although the source and target languages are different, many of the words such as named entities are usually represented by the same characters in both languages, making it possible to copy. Similarly, in text summarization, it is natural to use some words in the original text in the summarized text as well.

Specifically, to predict a target word at each timestep, our model first determines the source of the word generation, that is, whether to take one from a predefined shortlist or to copy one from the context. For the former, we apply the typical softmax operation, and for the latter, we use the attention mechanism to obtain the pointing softmax probability over the context words and pick the one of high probability. The model learns this decision so as to use the pointing only when the context includes a word that can be copied to the target. This way, our model can predict even the words which are not in the shortlist, as long as it appears in the context. Although some of the words still need to be labeled as UNK, i.e., if it is neither in the shortlist nor in the context, in experiments we show that this *learning when and where to point* improves the performance in machine translation and text summarization.

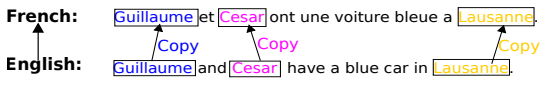

Figure 1: An example of how copying can happen for machine translation. Common words that appear both in source and the target can directly be copied from input to source. The rest of the unknown in the target can be copied from the input after being translated with a dictionary.

The rest of the paper is organized as follows. In the next section, we review the related works including pointer networks and previous approaches to the rare/unknown problem. In Section 3, we review the neural machine translation with attention mechanism which is the baseline in our experiments. Then, in Section 4, we propose our method dealing with the rare/unknown word problem, called the **Pointer Softmax (PS)**. The experimental results are provided in the Section 5 and we conclude our work in Section 6.

2 Related Work

The attention-based pointing mechanism is introduced first in the pointer networks (Vinyals et al., 2015). In the pointer networks, the output space of the target sequence is constrained to be the observations in the input sequence (not the input space). Instead of having a fixed dimension softmax output layer, softmax outputs of varying dimension is dynamically computed for each input sequence in such a way to maximize the attention probability of the target input. However, its applicability is rather limited because, unlike our model, there is no option to choose whether to point or not; it always points. In this sense, we can see the pointer networks as a special case of our model where we always choose to point a context word.

Several approaches have been proposed towards solving the rare words/unknown words problem, which can be broadly divided into three categories. The first category of the approaches focuses on improving the computation speed of the softmax output so that it can maintain a very large vocabulary. Because this only increases the shortlist size, it helps to mitigate the unknown word problem, but still suffers from the rare word problem. The hierarchical softmax (Morin and Bengio, 2005), importance sampling (Bengio and Senécal, 2008; Jean et al., 2014), and the noise contrastive estimation (Gutmann and Hyvärinen, 2012; Mnih and Kavukcuoglu, 2013) methods are in the class.

The second category, where our proposed method also belongs to, uses information from the context. Notable works are (Luong et al., 2015) and (Hermann et al., 2015). In particular, applying to machine translation task, (Luong et al., 2015) learns to point some words in source sentence and copy it to the target sentence, similarly to our method. However, it does not use attention mechanism, and by having fixed sized soft-

max output over the relative pointing range (e.g., -7, ..., -1, 0, 1, ..., 7), their model (the Positional All model) has a limitation in applying to more general problems such as summarization and question answering, where, unlike machine translation, the length of the context and the pointing locations in the context can vary dramatically. In question answering setting, (Hermann et al., 2015) have used placeholders on named entities in the context. However, the placeholder id is directly predicted in the softmax output rather than predicting its location in the context.

The third category of the approaches changes the unit of input/output itself from words to a smaller resolution such as characters (Graves, 2013) or bytecodes (Sennrich et al., 2015; Gillick et al., 2015). Although this approach has the main advantage that it could suffer less from the rare/unknown word problem, the training usually becomes much harder because the length of sequences significantly increases.

Simultaneously to our work, (Gu et al., 2016) and (Cheng and Lapata, 2016) proposed models that learn to copy from source to target and both papers analyzed their models on summarization tasks.

3 Neural Machine Translation Model with Attention

As the baseline neural machine translation system, we use the model proposed by (Bahdanau et al., 2014) that learns to (soft-)align and translate jointly. We refer this model as NMT.

The encoder of the NMT is a bidirectional RNN (Schuster and Paliwal, 1997). The forward RNN reads input sequence $\mathbf{x} = (x_1, \ldots, x_T)$ in left-to-right direction, resulting in a sequence of hidden states $(\overrightarrow{\mathbf{h}}_1, \ldots, \overrightarrow{\mathbf{h}}_T)$. The backward RNN reads $\mathbf{x}$ in the reversed direction and outputs $(\overleftarrow{\mathbf{h}}_1, \ldots, \overleftarrow{\mathbf{h}}_T)$. We then concatenate the hidden states of forward and backward RNNs at each time step and obtain a sequence of *annotation* vectors $(\mathbf{h}_1, \ldots, \mathbf{h}_T)$ where $\mathbf{h}_j = \left[\overrightarrow{\mathbf{h}}_j || \overleftarrow{\mathbf{h}}_j\right]$. Here, $||$ denotes the concatenation operator. Thus, each annotation vector $\mathbf{h}_j$ encodes information about the j-th word with respect to all the other surrounding words in both directions.

In the decoder, we usually use gated recurrent unit (GRU) (Cho et al., 2014; Chung et al., 2014). Specifically, at each time-step t, the soft-alignment mechanism first computes the relevance weight e_{tj} which determines the contribution of annotation vector $\mathbf{h}_j$ to the t-th target word. We use a non-linear mapping f (e.g., MLP) which takes $\mathbf{h}_j$, the previous decoder's hidden state $\mathbf{s}_{t-1}$ and the previous output y_{t-1} as input:

$$e_{tj} = f(\mathbf{s}_{t-1}, \mathbf{h}_j, y_{t-1}).$$

The outputs e_{tj} are then normalized as follows:

$$l_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})}. \quad (1)$$

We call l_{tj} as the relevance score, or the alignment weight, of the j-th annotation vector.

The relevance scores are used to get the *context vector* $\mathbf{c}_t$ of the t-th target word in the translation:

$$\mathbf{c}_t = \sum_{j=1}^{T} l_{tj}\mathbf{h}_j \ ,$$

The hidden state of the decoder $\mathbf{s}_t$ is computed based on the previous hidden state $\mathbf{s}_{t-1}$, the context vector $\mathbf{c}_t$ and the output word of the previous time-step y_{t-1}:

$$\mathbf{s}_t = f_r(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t), \quad (2)$$

where f_r is GRU.

We use a deep output layer (Pascanu et al., 2013) to compute the conditional distribution over words:

$$p(y_t = a|y_{<t}, \mathbf{x}) \propto \exp\left(\psi^a_{(\mathbf{W}_o, \mathbf{b}_o)} f_o(\mathbf{s}_t, y_{t-1}, \mathbf{c}_t)\right), \quad (3)$$

where $\mathbf{W}$ is a learned weight matrix and $\mathbf{b}$ is a bias of the output layer. f_o is a single-layer feed-forward neural network. $\psi_{(\mathbf{W}_o, \mathbf{b}_o)}(\cdot)$ is a function that performs an affine transformation on its input. And the superscript a in ψ^a indicates the a-th column vector of ψ.

The whole model, including both the encoder and the decoder, is jointly trained to maximize the (conditional) log-likelihood of target sequences given input sequences, where the training corpus is a set of $(\mathbf{x}_n, \mathbf{y}_n)$'s. Figure 2 illustrates the architecture of the NMT.

4 The Pointer Softmax

In this section, we introduce our method, called as the pointer softmax (PS), to deal with the rare and unknown words. The pointer softmax can be

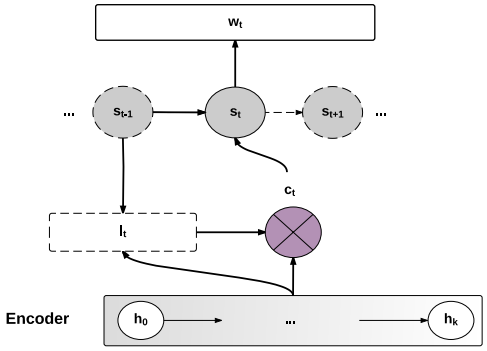

Figure 2: A depiction of neural machine translation architecture with attention. At each timestep, the model generates the attention weights $\mathbf{l}_t$. We use $\mathbf{l}_t$ the encoder's hidden state to obtain the context $\mathbf{c}_t$. The decoder uses $\mathbf{c}_t$ to predict a vector of probabilities for the words $\mathbf{w}_t$ by using softmax.

applicable approach to many NLP tasks, because it resolves the limitations about unknown words for neural networks. It can be used in parallel with other existing techniques such as the large vocabulary trick (Jean et al., 2014). Our model learns two key abilities jointly to make the pointing mechanism applicable in more general settings: (i) to predict whether it is required to use the pointing or not at each time step and (ii) to point any location of the context sequence whose length can vary widely over examples. Note that the pointer networks (Vinyals et al., 2015) are in lack of the ability (i), and the ability (ii) is not achieved in the models by (Luong et al., 2015).

To achieve this, our model uses two softmax output layers, the *shortlist* softmax and the *location* softmax. The shortlist softmax is the same as the typical softmax output layer where each dimension corresponds a word in the predefined word shortlist. The location softmax is a pointer network where each of the output dimension corresponds to the location of a word in the context sequence. Thus, the output dimension of the location softmax varies according to the length of the given context sequence.

At each time-step, if the model decides to use the shortlist softmax, we generate a word w_t from the shortlist. Otherwise, if it is expected that the context sequence contains a word which needs to be generated at the time step, we obtain the location of the context word l_t from the location softmax. The key to making this possible is deciding when to use the shortlist softmax or the location softmax at each time step. In order to accomplish this, we introduce a switching network to the model. The switching network, which is a multilayer perceptron in our experiments, takes the representation of the context sequence (similar to the input annotation in NMT) and the previous hidden state of the output RNN as its input. It outputs a binary variable z_t which indicates whether to use the shortlist softmax (when $z_t = 1$) or the location softmax (when $z_t = 0$). Note that if the word that is expected to be generated at each timestep is neither in the shortlist nor in the context sequence, the switching network selects the shortlist softmax, and then the shortlist softmax predicts UNK. The details of the pointer softmax model can be seen in Figure 3 as well.

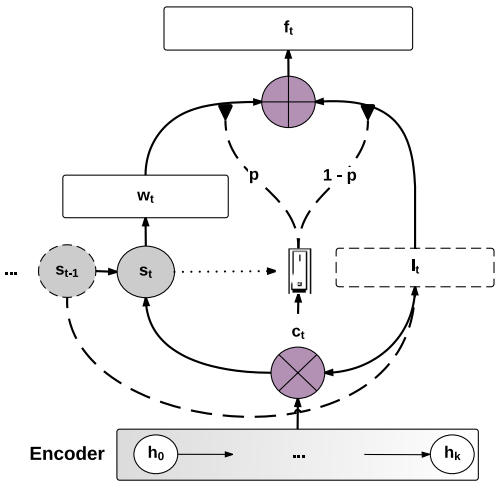

Figure 3: A simple depiction of the Pointer Softmax(PS) architecture. At each timestep as usual$\mathbf{l}_t$, $\mathbf{c}_t$ and the $\mathbf{w}_t$ for the words over the limited vocabulary(shortlist) is being generated. We have an additional switching variable z_t that decides whether to use $\mathbf{w}_t$ or copy the word from the input via $\mathbf{l}_t$. The final word prediction will be performed via pointer softmax $\mathbf{f}_t$ which can either copy the word from the source or predict the word from the shortlist vocabulary.

More specifically, our goal is to maximize the probability of observing the target word sequence $\mathbf{y} = (y_1, y_2, \ldots, y_{T_y})$ and the word generation source $\mathbf{z} = (z_1, z_2, \ldots, z_{T_y})$, given the context sequence $\mathbf{x} = (x_1, x_2, \ldots, x_{T_x})$:

$$p_\theta(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \prod_{t=1}^{T_y} p_\theta(y_t, z_t|y_{<t}, z_{<t}, \mathbf{x}). \quad (4)$$

Note that the word observation y_t can be either a word w_t from the shortlist softmax or a location l_t from the location softmax, depending on the switching variable z_t.

Considering this, we can factorize the above equation further

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \prod_{t \in \mathcal{T}_w} p(w_t, z_t|(y,z)_{<t}, \mathbf{x}) \times \prod_{t' \in \mathcal{T}_l} p(l_{t'}, z_{t'}|(y,z)_{<t'}, \mathbf{x}). \quad (5)$$

Here, $\mathcal{T}_w$ is a set of time steps where $z_t = 1$, and $\mathcal{T}_l$ is a set of time-steps where $z_t = 0$. And, $\mathcal{T}_w \cup \mathcal{T}_l = \{1, 2, \ldots, T_y\}$ and $\mathcal{T}_w \cap \mathcal{T}_l = \emptyset$. We denote all previous observations at step t by $(y,z)_{<t}$. Note also that $\mathbf{h}_t = f((y,z)_{<t})$.

Then, the joint probabilities inside each product can be further factorized as follows:

$$p(w_t, z_t|(y,z)_{<t}) = p(w_t|z_t = 1, (y,z)_{<t}) \times p(z_t = 1|(y,z)_{<t}) \quad (6)$$

$$p(l_t, z_t|(y,z)_{<t}) = p(l_t|z_t = 0, (y,z)_{<t}) \times p(z_t = 0|(y,z)_{<t}) \quad (7)$$

here, we omitted $\mathbf{x}$ which is conditioned on all probabilities in the above.

The switch probability is modeled as a multilayer perceptron with binary output:

$$p(z_t = 1|(y,z)_{<t}, \mathbf{x}) = \sigma(f(\mathbf{x}, \mathbf{h}_{t-1}; \theta)) \quad (8)$$

$$p(z_t = 0|(y,z)_{<t}, \mathbf{x}) = 1 - \sigma(f(\mathbf{x}, \mathbf{h}_{t-1}; \theta)). \quad (9)$$

And $p(w_t|z_t = 1, (y,z)_{<t}, \mathbf{x})$ is the shortlist softmax and $p(l_t|z_t = 0, (y,z)_{<t}, \mathbf{x})$ is the location softmax which can be a pointer network. $\sigma(\cdot)$ stands for the sigmoid function, $\sigma(x) = \frac{1}{\exp(-x)+1}$.

Given N such context and target sequence pairs, our training objective is to maximize the following log likelihood w.r.t. the model parameter θ

$$\arg\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y_n, z_n|x_n). \quad (10)$$

4.1 Basic Components of the Pointer Softmax

In this section, we discuss practical details of the three fundamental components of the pointer softmax. The interactions between these components and the model is depicted in Figure 3.

Location Softmax $\mathbf{l}_t$: The location of the word to copy from source text to the target is predicted by the location softmax $\mathbf{l}_t$. The location softmax outputs the conditional probability distribution $p(l_t|z_t = 0, (y,z)_{<t}, \mathbf{x})$. For models using the attention mechanism such as NMT, we can reuse the probability distributions over the source words in order to predict the location of the word to point. Otherwise we can simply use a pointer network of the model to predict the location.

Shortlist Softmax $\mathbf{w}_t$: The subset of the words in the vocabulary V is being predicted by the shortlist softmax $\mathbf{w}_t$.

Switching network d_t : The switching network d_t is an MLP with sigmoid output function that outputs a scalar probability of switching between $\mathbf{l}_t$ and $\mathbf{w}_t$, and represents the conditional probability distribution $p(z_t|(y,z)_{<t}, \mathbf{x})$. For NMT model, we condition the MLP that outputs the switching probability on the representation of the context of the source text $\mathbf{c}_t$ and the hidden state of the decoder $\mathbf{h}_t$. Note that, during the training, d_t is observed, and thus we do not have to sample.

The output of the pointer softmax, $\mathbf{f}_t$ will be the concatenation of the the two vectors, $d_t \times \mathbf{w}_t$ and $(1 - d_t) \times \mathbf{l}_t$.

At test time, we compute Eqn. (6) and (7) for all shortlist word w_t and all location l_t, and pick the word or location of the highest probability.

5 Experiments

In this section, we provide our main experimental results with the pointer softmax on machine translation and summarization tasks. In our experiments, we have used the same baseline model and just replaced the softmax layer with pointer softmax layer at the language model. We use the Adadelta (Zeiler, 2012) learning rule for the training of NMT models. The code for pointer softmax model is available at `https://github.com/caglar/pointer_softmax`.

5.1 The Rarest Word Detection

We construct a synthetic task and run some preliminary experiments in order to compare the results with the pointer softmax and the regular softmax's performance for the rare-words. The vocabulary size of our synthetic task is $|V| = 600$ using sequences of length 7. The words in the sequences are sampled according to their unigram distribu-

tion which has the form of a geometric distribution. The task is to predict the least frequent word in the sequence according to unigram distribution of the words. During the training, the sequences are generated randomly. Before the training, validation and test sets are constructed with a fixed seed.

We use a GRU layer over the input sequence and take the last-hidden state, in order to get the summary $\mathbf{c}_t$ of the input sequence. The $\mathbf{w}_t$, $\mathbf{l}_t$ are only conditioned on $\mathbf{c}_t$, and the MLP predicting the d_t is conditioned on the latent representations of $\mathbf{w}_t$ and $\mathbf{l}_t$. We use minibatches of size 250 using adam adaptive learning rate algorithm (Kingma and Adam, 2015) using the learning rate of 8×10^{-4} and hidden layers with 1000 units.

We train a model with pointer softmax where we assign pointers for the rarest 60 words and the rest of the words are predicted from the shortlist softmax of size 540. We observe that increasing the inverse temperature of the sigmoid output of d_t to 2, in other words making the decisions of d_t to become sharper, works better, i.e. $d_t = \sigma(2x)$.

At the end of training with pointer softmax we obtain the error rate of 17.4% and by using softmax over all 600 tokens, we obtain the error-rate of 48.2%.

5.2 Summarization

In these series of experiments, we use the annotated Gigaword corpus as described in (Rush et al., 2015). Moreover, we use the scripts that are made available by the authors of (Rush et al., 2015) [1] to preprocess the data, which results to approximately 3.8M training examples. This script generates about 400K validation and an equal number of test examples, but we use a randomly sampled subset of 2000 examples each for validation and testing. We also have made small modifications to the script to extract not only the tokenized words, but also system-generated named-entity tags. We have created two different versions of training data for pointers, which we call UNK-pointers data and entity-pointers data respectively.

For the UNK-pointers data, we trim the vocabulary of the source and target data in the training set and replace a word by the UNK token whenever a word occurs less than 5 times in either source or target data separately. Then, we create pointers from each UNK token in the target data to the position in the corresponding source document where the same word occurs in the source, as seen in the data before UNKs were created. It is possible that the source can have an UNK in the matching position, but we still created a pointer in this scenario as well. The resulting data has 2.7 pointers per 100 examples in the training set and 9.1 pointers rate in the validation set.

In the entity-pointers data, we exploit the named-entity tags in the annotated corpus and first anonymize the entities by replacing them with an integer-id that always starts from 1 for each document and increments from left to right. Entities that occur more than once in a single document share the same id. We create the anonymization at token-level, so as to allow partial entity matches between the source and target for multi-token entities. Next, we create a pointer from the target to source on similar lines as before, but only for exact matches of the anonymized entities. The resulting data has 161 pointers per 100 examples in the training set and 139 pointers per 100 examples in the validation set.

If there are multiple matches in the source, either in the UNK-pointers data or the entity-pointers data, we resolve the conflict in favor of the first occurrence of the matching word in the source document. In the UNK data, we model the UNK tokens on the source side using a single placeholder embedding that is shared across all documents, and in the entity-pointers data, we model each entity-id in the source by a distinct placeholder, each of which is shared across all documents.

In all our experiments, we use a bidirectional GRU-RNN (Chung et al., 2014) for the encoder and a uni-directional RNN for the decoder. To speed-up training, we use the large-vocabulary trick (Jean et al., 2014) where we limit the vocabulary of the softmax layer of the decoder to 2000 words dynamically chosen from the words in the source documents of each batch and the most common words in the target vocabulary. In both experiments, we fix the embedding size to 100 and the hidden state dimension to 200. We use pretrained word2vec vectors trained on the same corpus to initialize the embeddings, but we finetuned them by backpropagating through the embeddings during training. Our vocabulary sizes are fixed to 125K for source and 75K for target for both exper-

[1] https://github.com/facebook/NAMAS

iments.

We use the reference data for pointers for the model only at the training time. During the test time, the switch makes a decision at every timestep on which softmax layer to use.

For evaluation, we use full-length Rouge F1 using the official evaluation tool [2]. In their work, the authors of (Bahdanau et al., 2014) use full-length Rouge Recall on this corpus, since the maximum length of limited-length version of Rouge recall of 75 bytes (intended for DUC data) is already long for Gigaword summaries. However, since full-length Recall can unfairly reward longer summaries, we also use full-length F1 in our experiments for a fair comparison between our models, independent of the summary length.

The experimental results comparing the Pointer Softmax with NMT model are displayed in Table 1 for the UNK pointers data and in Table 2 for the entity pointers data. As the experiments show, pointer softmax improves over the baseline NMT on both UNK data and entities data. Our hope was that the improvement would be larger for the entities data since the incidence of pointers was much greater. However, it turns out this is not the case, and we suspect the main reason is anonymization of entities which removed data-sparsity by converting all entities to integer-ids that are shared across all documents. We believe that on de-anonymized data, our model could help more, since the issue of data-sparsity is more acute in this case.

Table 1: Results on Gigaword Corpus when pointers are used for UNKs in the training data, using Rouge-F1 as the evaluation metric.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	34.87	16.54	32.27
NMT + lvt + PS	**35.19**	**16.66**	**32.51**

Table 2: Results on anonymized Gigaword Corpus when pointers are used for entities, using Rouge-F1 as the evaluation metric.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	34.89	**16.78**	32.37
NMT + lvt + PS	**35.11**	16.76	**32.55**

[2] `http://www.berouge.com/Pages/default.aspx`

Table 3: Results on Gigaword Corpus for modeling UNK's with pointers in terms of recall.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	36.45	17.41	33.90
NMT + lvt + PS	**37.29**	**17.75**	**34.70**

In Table 3, we provide the results for summarization on Gigaword corpus in terms of recall as also similar comparison done by (Rush et al., 2015). We observe improvements on all the scores with the addition of pointer softmax. Let us note that, since the test set of (Rush et al., 2015) is not publicly available, we sample 2000 texts with their summaries without replacement from the validation set and used those examples as our test set.

In Table 4 we present a few system generated summaries from the Pointer Softmax model trained on the UNK pointers data. From those examples, it is apparent that the model has learned to accurately point to the source positions whenever it needs to generate rare words in the summary.

5.3 Neural Machine Translation

In our neural machine translation (NMT) experiments, we train NMT models with attention over the Europarl corpus (Bahdanau et al., 2014) over the sequences of length up to 50 for English to French translation. [3]. All models are trained with early-stopping which is done based on the negative log-likelihood (NLL) on the development set. Our evaluations to report the performance of our models are done on `newstest2011` by using BLUE score. [4]

We use $30,000$ tokens for both the source and the target language shortlist vocabularies (1 of the token is still reserved for the unknown words). The whole corpus contains $134,831$ unique English words and $153,083$ unique French words. We have created a word-level dictionary from French to English which contains translation of 15,953 words that are neither in shortlist vocabulary nor dictionary of common words for both the source and the target. There are about $49,490$ words shared between English and French parallel corpora of Europarl.

[3] In our experiments, we use an existing code, provided in `https://github.com/kyunghyuncho/dl4mt-material`, and on the original model we only changed the last softmax layer for our experiments

[4] We compute the BLEU score using the multi-blue.perl script from Moses on tokenized sentence pairs.

Table 4: Generated summaries from NMT with PS. Boldface words are the words copied from the source.

Source #1	china 's tang **gonghong** set a world record with a clean and jerk lift of ### kilograms to win the women 's over-## kilogram weightlifting title at the asian games on tuesday .
Target #1	china 's tang <unk>,sets world weightlifting record
NMT+PS #1	china 's tang **gonghong** wins women 's weightlifting weightlifting title at asian games
Source #2	owing to criticism , nbc said on wednesday that it was ending a **three-month-old** experiment that would have brought the first liquor advertisements onto national broadcast network television .
Target #2	advertising : nbc retreats from liquor commercials
NMT+PS #2	nbc says it is ending a **three-month-old** experiment
Source #3	a senior trade union official here wednesday called on ghana 's government to be " **mindful** of the plight " of the ordinary people in the country in its decisions on tax increases .
Target #3	tuc official,on behalf of ordinary ghanaians
NMT+PS #3	ghana 's government urged to be **mindful** of the plight

During the training, in order to decide whether to pick a word from the source sentence using attention/pointers or to predict the word from the short-list vocabulary, we use a simple heuristic. If the word is not in the short-list vocabulary, we first check if the word $\mathbf{y}_t$ itself appears in the source sentence. If it is not, we check if the word itself is in the source sentence by using the shared words lookup table for the source and the target language. If the word is in the source sentence, we then use the location of the word in the source as the target. Otherwise we check if one of the English senses from the cross-language dictionary of the French word is in the source. If it is in the source sentence, then we use the location of that word as our translation. Otherwise we just use the argmax of $\mathbf{l}_t$ as the target.

For switching network d_t, we observed that using a two-layered MLP with noisy-tanh activation (Gulcehre et al., 2016) function with residual connection from the lower layer (He et al., 2015) activation function to the upper hidden layers improves the BLEU score about 1 points over the d_t using ReLU activation function. We initialized the biases of the last sigmoid layer of d_t to -1 such that if d_t becomes more biased toward choosing the shortlist vocabulary at the beginning of the training. We renormalize the gradients if the norm of the gradients exceed 1 (Pascanu et al., 2012).

In Table 5, we provided the result of NMT with pointer softmax and we observe about 3.6 BLEU score improvement over our baseline.

Table 5: Europarl Dataset (EN-FR)

	BLEU-4
NMT	20.19
NMT + PS	**23.76**

In Figure 4, we show the validation curves of the NMT model with attention and the NMT model with shortlist-softmax layer. Pointer softmax converges faster in terms of number of minibatch updates and achieves a lower validation negative-log-likelihood (NLL) (63.91) after $200k$ updates over the Europarl dataset than the NMT model with shortlist softmax trained for $400k$ minibatch updates (65.26). Pointer softmax converges faster than the model using the shortlist softmax, because the targets provided to the pointer softmax also acts like guiding hints to the attention.

6 Conclusion

In this paper, we propose a simple extension to the traditional soft attention-based shortlist softmax by using pointers over the input sequence. We show that the whole model can be trained jointly with single objective function. We observe noticeable improvements over the baselines on machine translation and summarization tasks by using pointer softmax. By doing a very simple mod-

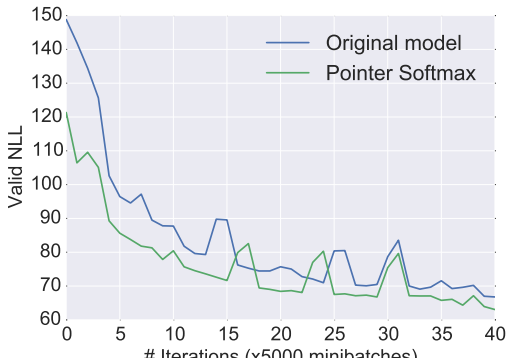

Figure 4: A comparison of the validation learning-curves of the same NMT model trained with pointer softmax and the regular softmax layer. As can be seen from the figures, the model trained with pointer softmax converges faster than the regular softmax layer. Switching network for pointer softmax in this Figure uses ReLU activation function.

ification over the NMT, our model is able to generalize to the unseen words and can deal with rare-words more efficiently. For the summarization task on Gigaword dataset, the pointer softmax was able to improve the results even when it is used together with the large-vocabulary trick. In the case of neural machine translation, we observed that the training with the pointer softmax is also improved the convergence speed of the model as well. For French to English machine translation on Europarl corpora, we observe that using the pointer softmax can also improve the training convergence of the model.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Bengio and Senécal2008] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722.

[Bordes et al.2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

[Cheng and Lapata2016] Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chung et al.2014] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

[Gillick et al.2015] Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Gu et al.2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

[Gulcehre et al.2016] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. 2016. Noisy activation functions. *arXiv preprint arXiv:1603.00391*.

[Gutmann and Hyvärinen2012] Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for mage recognition. *arXiv preprint arXiv:1512.03385*.

[Hermann et al.2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

[Jean et al.2014] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

[Kingma and Adam2015] Diederik P Kingma and Jimmy Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representation*.

[Luong et al.2015] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

[Matthews et al.2012] Danielle Matthews, Tanya Behne, Elena Lieven, and Michael Tomasello. 2012. Origins of the human pointing gesture: a training study. *Developmental science*, 15(6):817–829.

[Mnih and Kavukcuoglu2013] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

[Morin and Bengio2005] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

[Pascanu et al.2012] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

[Pascanu et al.2013] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

[Rush et al.2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

[Schuster and Paliwal1997] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

[Sennrich et al.2015] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

[Theano Development Team2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

[Tomasello et al.2007] Michael Tomasello, Malinda Carpenter, and Ulf Liszkowski. 2007. A new look at infant pointing. *Child development*, 78(3):705–722.

[Vinyals et al.2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

7 Acknowledgments

We would also like to thank the developers of Theano [5], for developing such a powerful tool for scientific computing (Theano Development Team, 2016). We acknowledge the support of the following organizations for research funding and computing support: NSERC, Samsung, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. C. G. thanks for IBM T.J. Watson Research for funding this research during his internship between October 2015 and January 2016.

[5] `http://deeplearning.net/software/theano/`