Pointing the Unknown Words

Amr Aly (1848399)

July 11, 2019

Abstract

Out-of-vocabulary (OOV) words are a common problem in NLP systems that decrease the system's performance in many tasks. To solve this problem in sequence-to-sequence tasks, the authors [1] propose a combination of two previous works: (1) Encoder-Decoder architecture (with attention) [2], (2) Pointer Networks [3]. The first model predicts the next output word from a shortlist vocabulary while the second predicts the location of a word in the input sentence to copy from. To combine both models, at each time-step, the decision to choose which output is made by an MLP which is conditioned on the context.

1 Introduction

Most NLP systems work on words as the basic block of language. However, this creates a problem since language can contain an arbitrary number of words and new words can be always created with a language's vocabulary constantly evolving. Thus, a common approach in many systems is to limit the vocab to a list of predefined words which introduces us to the problem of unknown words.

Unknown words lead to the loss of important information and context in a given text, resulting in a sub-optimal learning procedure. Also according to Zipf's law, frequency of words that are present in the predefined shortlist are inversely proportional with their rank in that list. This means that we can not learn a good representation for rare words like the others.

Tasks that are critically dependant on language understanding, like question answering and named entity recognition -among others- motivate the search for a method to deal with rare and unknown words.

By observing human behavior, we notice an efficient mechanism to draw attention to unknown objects: *pointing*. Pointing makes it possible to deliver information and to associate context to a particular object without knowing how to call it. This was approached by pointer networks [3]. Pointer networks are sequence-to-sequence models where the output is discrete tokens corresponding to positions in an input sequence.

Some NLP tasks, like machine translation, can be seen as a task of predicting target text given some context, where some of the target words already appear in the context. In this case, a model [1] can learn to point at a word in the context and also learn when to do that.

The model proposed by [1] predicts a target word at each time-step. To do that, first it determines the word generation source, i.e. whether to choose a word from a predefined shortlist or copy from the context. For the shortlist source, a softmax function is applied to compute the conditional probability distribution of the words in the shortlist. To copy from the context, a pointer network can be trained to learn the most probable word to copy. Nevertheless, some words will still be labeled as unknown if they do not exist in neither the shortlist nor the context.

2 Related Work

There has been three general approaches to the rare/unknown word problem. The first relies on increasing the computation speed of the softmax layer (e.g. Hierarchical softmax) so it can support a larger vocab. However, this still does not solve the problem because: (1) for the unknown words, we can still see new words during test time, (2) rare words stay rare and without meaningful representations.

The second approach, like the method proposed in this paper, uses information from the input context. While the third approach uses characters instead of words as the basic block. Although in this approach, the problem of rare/unknown is almost non-existent, the computational requirements to train the model increase significantly since the length of the sequences increases.

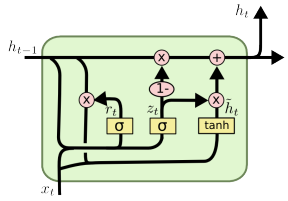
The attention-based pointing method was first introduced in pointer networks [3]. In pointer networks, the output space of the target sequence is limited by the input sequence. Instead of a regular softmax output layer with a fixed size, the outputs vary dynamically for each input sequence to maximize the attention probability of the input sequence. However, it does not have the ability to choose when to point, it always does. In this case, it can be considered as a special case of this model.

3 Neural Machine Translation Model with Attention

3.1 GRUs

A problem that arises when using vanilla RNNs is their limited memory capacity, caused by the vanishing (or exploding) gradient problem. A simple variant was introduced to control the information flow through time-steps using a gated mechanism and a cell state.

Gated Recurrent Units (GRUs) use an *update gate* z_t and a *reset gate* r_t to control the flow. The update gate decides how much of the past information to pass to the future, while the reset (sometime called *forget*) gate has an opposite functionality where it decides decide how much of the past information to forget.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 1: GRU cell[1]

3.2 Encoder-Decoder

The encoder-decoder "framework", first described in [4], is an approach to sequence-to-sequence learning tasks. The encoder -typically an RNN(-variant)- learns from the input sequence and outputs a context vector that contains a representation for the whole sequence. GRUs/LSTMs are typically used to handle long-term dependencies among other things.

The decoder -another RNN- starts from a fixed-length vector representation for a variable-length sequence and *generates* the output sequence one step at a time. At each time-step the output of the decoder depends on: the previous decoder output, the decoder hidden state and the context vector from the encoder.

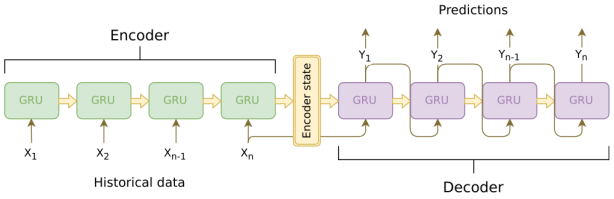


Figure 2: Encoder-Decoder architecture[2]

3.3 Attention Mechanism

The attention mechanism was first introduced for neural machine translation [2] as a method to learn to align and translate jointly. In the decoder of section 3.2, at each time-step t, the alignment mechanism first computes the "energies" e_{tj} which determines the contribution of annotation vector h_j to the t-th target word. The energies are learned through a non-linear mapping function f (e.g. MLP):

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

The outputs are then normalized using a softmax function, used to calculate the new context vector and then a deep output layer is used to compute the conditional distribution over words.

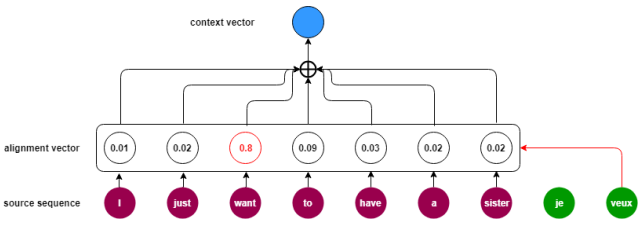


Figure 3: Attention Mechanism[3]

4 Pointer Softmax

In this section, we introduce the pointer softmax (PS) model, to approach the rare/unknown words problem. The model is an extension to the

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] https://github.com/Arturus/kaggle-web-traffic

[3] https://machinetalk.org/2019/03/29/neural-machine-translation-with-attention-mechanism/

NMT model presented in section 3, where it learns two key abilities jointly to generalize the pointing mechanism over arbitrary sequences: (1) learning *when* to use the pointing mechanism, (2) choosing any location from the source sequence whose length can vary.

To be able to do this, the model utilizes two softmax output layers, the *shortlist* softmax and the *location* softmax. The shortlist softmax is the same as the NMT model where it computes the conditional distribution over a list of vocab words. The location softmax, however, computes the conditional distribution over the input sequence, which has variable length over the examples.

To learn, at each time-step, whether to choose a word from the shortlist or the input sequence, a switching network is trained to make this decision. The switching network, that can be chosen to be a simple MLP, is conditioned on the context vector (from the encoder) and the previous decoder hidden state. It outputs a binary variable z_t which indicates the decision to consider the output of the shortlist softmax ($z_t = 1$) or the location softmax ($z_t = 0$).

The goal of the model is to maximize the following function:

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

where $\mathbf{y}$ is the target word sequence, $\mathbf{z}$ is the switching variable and $\mathbf{x}$ is the input context sequence. $\mathcal{T}_w$ 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, $\mathcal{T}_w \cup \mathcal{T}_l = \{1, 2, ..., T\}$ and $\mathcal{T}_w \cap \mathcal{T}_l = \phi$. We denote all previous observations at step t by $(y, z)_{<t}$.

4.1 Shortlist softmax

The conditional probability distribution over the vocab list predicted at each time-step.

$$p(w_t|z_t = 1, (y,z)_{<t}, \mathbf{x})$$

4.2 Location softmax

The conditional probability distribution over the input sequence predicted at each time-step.

$$p(l_t|z_t = 0, (y,z)_{<t}, \mathbf{x})$$

Note that for models using the attention mechanism, the calculated distributions can be reused directly instead of training a pointer network.

4.3 Switching network

The switching network can be modeled as a mapping f (e.g. MLP) with sigmoid output function that outputs the probability to choose l_t or w_t. More specifically, the network represents the conditional probability:

$$p(z_t|(y,z)_{<t}, \mathbf{x})$$

The network is conditioned on the context vector $\mathbf{c}_t$ and the decoder hidden state $\mathbf{h}_t$.

5 Experiments

In this section, we provide the experimental results comparing the baseline NMT model (section 3) to the pointer softmax model (section 4) on the machine translation task. The results are reported on two datasets and we report the BLEU score [4] which is a measure of the quality of a translation system [5].

Both models were trained using the Adam optimizer and the loss function was computed as the negative log-likelihood (NLL). PyTorch's *Dataset* and *Iterator* wrappers were used to load the datasets, and create train, validation and test sets iterators.

The first dataset is from the WMT 2016 multimodal task[5]. Vocabulary was created for both English (source) and German (target) languages using PyTorch's *Vocab* wrapper. After excluding tokens appearing only once in the dataset, we found 7,702 unique tokens for English and 9,595 unique tokens for German (two tokens are reserved for the unknown word and padding token). The dataset contains 29,000 training pairs, 1,014 validation pair and 1,000 test pairs.

The second dataset is from the IWSLT 2016 TED talk translation task [6]. Same procedures were followed to construct the vocab for the same languages. However, due to this dataset being significantly larger (and hardware limits), we extract sentences with 50 or less tokens. We also choose the first 7,702 and 9,595 unique tokens for English and German respectively (the same as the

[4] https://github.com/mjpost/sacreBLEU

[5] https://www.statmt.org/wmt16/multimodal-task.html

[6] https://sites.google.com/site/iwsltevaluation2016/

Dataset	WMT	ISWLT
NMT	86.7	81.37
NMT + PS	89.1	82.36

Table 1: Machine Translation Results[7]

first task), which will also allow us to measure the unknown word problem more effectively since the dataset has originally 108,806 and 188,584 unique tokens for these languages.

During training we notice that Pointer Softmax converges slightly faster than NMT, however the difference is negligible. We also notice in Table 1 PS scores better than NMT with an average of 1.5 points in the BLEU score. We can see a better overall performance on the first dataset since it contains shorter sentences and less vocab, which makes it easier to translate.

6 Conclusion

The idea of combining encoder-decoder models with pointer networks under a single objective is a successful idea. Backed by psychological observations, *pointing* at unknown objects indeed helps convey information about them. Using this interesting mechanism improved many sequence-to-sequence learning tasks, where the model *learns* to point to the unknown word and copy it from context to target.

We notice improved results on two datasets for the machine translation. And in addition to being an extra simple computation (over the NMT model), all this makes using this new model feasible and desirable.

References

[1] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou and Yoshua Bengio. 2016. Pointing the Unknown Words. abs/1603.08148.

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

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

[4] Ilya Sutskever, Oriol Vinyals and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. abs/1409.3215.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.

[7] BLEU scores reported