Speeding up Context-based Sentence Representation Learning with Non-autoregressive Convolutional Decoding

Abstract

Context plays an important role in human language understanding, thus it may also be useful for machines learning vector representations of language. In this paper, we explore an asymmetric encoder-decoder structure for unsupervised context-based sentence representation learning. We carefully designed experiments to show that neither an autoregressive decoder nor an RNN decoder is required. After that, we designed a model which still keeps an RNN as the encoder, while using a non-autoregressive convolutional decoder. We further combine a suite of effective designs to significantly improve model efficiency while also achieving better performance. Our model is trained on two different large unlabelled corpora, and in both cases the transferability is evaluated on a set of downstream NLP tasks. We empirically show that our model is simple and fast while producing rich sentence representations that excel in downstream tasks.

Introduction

Learning distributed representations of sentences is an important and hard topic in both the deep learning and natural language processing communities, since it requires machines to encode a sentence with rich language content into a fixed-dimension vector filled with real numbers. Our goal is to build a distributed sentence encoder learnt in an unsupervised fashion by exploiting the structure and relationships in a large unlabelled corpus.

Numerous studies in human language processing have supported that rich semantics of a word or sentence can be inferred from its context BIBREF0 , BIBREF1 . The idea of learning from the

co-occurrence BIBREF2 was recently successfully applied to vector representation learning for words in BIBREF3 and BIBREF4 .

A very recent successful application of the distributional hypothesis BIBREF0 at the sentence-level is the skip-thoughts model BIBREF5 . The skip-thoughts model learns to encode the current sentence and decode the surrounding two sentences instead of the input sentence itself, which achieves overall good performance on all tested downstream NLP tasks that cover various topics. The major issue is that the training takes too long since there are two RNN decoders to reconstruct the previous sentence and the next one independently. Intuitively, given the current sentence, inferring the previous sentence and inferring the next one should be different, which supports the usage of two independent decoders in the skip-thoughts model. However, BIBREF6 proposed the skip-thought neighbour model, which only decodes the next sentence based on the current one, and has similar performance on downstream tasks compared to that of their implementation of the skip-thoughts model.

In the encoder-decoder models for learning sentence representations, only the encoder will be used to map sentences to vectors after training, which implies that the quality of the generated language is not our main concern. This leads to our two-step experiment to check the necessity of applying an autoregressive model as the decoder. In other words, since the decoder's performance on language modelling is not our main concern, it is preferred to reduce the complexity of the decoder to speed up the training process. In our experiments, the first step is to check whether “teacher-forcing” is required during training if we stick to using an autoregressive model as the decoder, and the second step is to check whether an autoregressive decoder is necessary to learn a good sentence encoder. Briefly, the experimental results show that an autoregressive decoder is indeed not essential in learning a good sentence encoder; thus the two findings of our experiments lead to our final model design.

Our proposed model has an asymmetric encoder-decoder structure, which keeps an RNN as the encoder

and has a CNN as the decoder, and the model explores using only the subsequent context information as the supervision. The asymmetry in both model architecture and training pair reduces a large amount of the training time.

The contribution of our work is summarised as:

The following sections will introduce the components in our “RNN-CNN” model, and discuss our experimental design.

RNN-CNN Model

Our model is highly asymmetric in terms of both the training pairs and the model structure. Specifically, our model has an RNN as the encoder, and a CNN as the decoder. During training, the encoder takes the INLINEFORM0 -th sentence INLINEFORM1 as the input, and then produces a fixed-dimension vector INLINEFORM2 as the sentence representation; the decoder is applied to reconstruct the paired target sequence INLINEFORM3 that contains the subsequent contiguous words. The distance between the generated sequence and the target one is measured by the cross-entropy loss at each position in INLINEFORM4 . An illustration is in Figure FIGREF4 . (For simplicity, we omit the subscript INLINEFORM5 in this section.)

1. Encoder: The encoder is a bi-directional Gated Recurrent Unit (GRU, BIBREF7). Suppose that an input sentence INLINEFORM0 contains INLINEFORM1 words, which are INLINEFORM2 , and they are transformed by an embedding matrix INLINEFORM3 to word vectors. The bi-directional GRU takes one word vector at a time, and processes the input sentence in both the forward and backward directions; both sets of hidden states are concatenated to form the hidden state matrix INLINEFORM7 , where INLINEFORM8 is the dimension of the hidden states INLINEFORM9 (INLINEFORM10).

2. Representation: We aim to provide a model with faster training speed and better transferability than existing algorithms; thus we choose to apply a parameter-free composition function, which is a concatenation of the outputs from a global mean pooling over time and a global max pooling over time, on the computed sequence of hidden states INLINEFORM0 . The composition function is represented as DISPLAYFORM0

where INLINEFORM0 is the max operation on each row of the matrix INLINEFORM1 , which outputs a vector with dimension INLINEFORM2 . Thus the representation INLINEFORM3 . 3. Decoder: The decoder is a 3-layer CNN to reconstruct the paired target sequence INLINEFORM4 , which needs to expand INLINEFORM5 , which can be considered as a sequence with only one element, to a sequence with INLINEFORM6 elements. Intuitively, the decoder could be a stack of deconvolution layers. For fast training speed, we optimised the architecture to make it possible to use fully-connected layers and convolution layers in the decoder, since generally, convolution layers run faster than deconvolution layers in modern deep learning frameworks.

Suppose that the target sequence INLINEFORM0 has INLINEFORM1 words, which are INLINEFORM2 , the first layer of deconvolution will expand INLINEFORM3 , into a feature map with INLINEFORM4 elements. It can be easily implemented as a concatenation of outputs from INLINEFORM5 linear transformations in parallel. Then the second and third layer are 1D-convolution layers. The output feature map is INLINEFORM6 , where INLINEFORM7 is the dimension of the word vectors.

Note that our decoder is not an autoregressive model and has high training efficiency. We will discuss the reason for choosing this decoder which we call a predict-all-words CNN decoder.

4. Objective: The training objective is to maximise the likelihood of the target sequence being generated from the decoder. Since in our model, each word is predicted independently, a softmax layer is applied

after the decoder to produce a probability distribution over words in INLINEFORM0 at each position, thus the probability of generating a word INLINEFORM1 in the target sequence is defined as: DISPLAYFORM0

where, INLINEFORM0 is the vector representation of INLINEFORM1 in the embedding matrix INLINEFORM2 , and INLINEFORM3 is the dot-product between the word vector and the feature vector produced by the decoder at position INLINEFORM4 . The training objective is to minimise the sum of the negative log-likelihood over all positions in the target sequence INLINEFORM5 : DISPLAYFORM0

where INLINEFORM0 and INLINEFORM1 contain the parameters in the encoder and the decoder, respectively. The training objective INLINEFORM2 is summed over all sentences in the training corpus.

Architecture Design

We use an encoder-decoder model and use context for learning sentence representations in an unsupervised fashion. Since the decoder won't be used after training, and the quality of the generated sequences is not our main focus, it is important to study the design of the decoder. Generally, a fast training algorithm is preferred; thus proposing a new decoder with high training efficiency and also strong transferability is crucial for an encoder-decoder model.

CNN as the decoder

Our design of the decoder is basically a 3-layer ConvNet that predicts all words in the target sequence at once. In contrast, existing work, such as skip-thoughts BIBREF5 , and CNN-LSTM BIBREF9 , use autoregressive RNNs as the decoders. As known, an autoregressive model is good at generating sequences with high quality, such as language and speech. However, an autoregressive decoder seems

to be unnecessary in an encoder-decoder model for learning sentence representations, since it won't be used after training, and it takes up a large portion of the training time to compute the output and the gradient. Therefore, we conducted experiments to test the necessity of using an autoregressive decoder in learning sentence representations, and we had two findings.

Finding I: It is not necessary to input the correct words into an autoregressive decoder for learning sentence representations.

The experimental design was inspired by BIBREF10 . The model we designed for the experiment has a bi-directional GRU as the encoder, and an autoregressive decoder, including both RNN and CNN. We started by analysing the effect of different sampling strategies of the input words on learning an auto-regressive decoder.

We compared three sampling strategies of input words in decoding the target sequence with an autoregressive decoder: (1) Teacher-Forcing: the decoder always gets the ground-truth words; (2) Always Sampling: at time step INLINEFORM0 , a word is sampled from the multinomial distribution predicted at time step INLINEFORM1 ; (3) Uniform Sampling: a word is uniformly sampled from the dictionary INLINEFORM2 , then fed to the decoder at every time step.

The results are presented in Table TABREF10 (top two subparts). As we can see, the three decoding settings do not differ significantly in terms of the performance on selected downstream tasks, with RNN or CNN as the decoder. The results show that, in terms of learning good sentence representations, the autoregressive decoder doesn't require the correct ground-truth words as the inputs.

Finding II: The model with an autoregressive decoder performs similarly to the model with a predict-all-words decoder.

With Finding I, we conducted an experiment to test whether the model needs an autoregressive decoder at all. In this experiment, the goal is to compare the performance of the predict-all-words decoders and that of the autoregressive decoders separate from the RNN/CNN distinction, thus we designed a predict-all-words CNN decoder and RNN decoder. The predict-all-words CNN decoder is described in Section SECREF2 , which is a stack of three convolutional layers, and all words are predicted once at the output layer of the decoder. The predict-all-words RNN decoder is built based on our CNN decoder. To keep the number of parameters of the two predict-all-words decoder roughly the same, we replaced the last two convolutional layers with a bidirectional GRU.

The results are also presented in Table TABREF10 (3rd and 4th subparts). The performance of the predict-all-words RNN decoder does not significantly differ from that of any one of the autoregressive RNN decoders, and the same situation can be also observed in CNN decoders.

These two findings indeed support our choice of using a predict-all-words CNN as the decoder, as it brings the model high training efficiency while maintaining strong transferability.

Mean+Max Pooling

Since the encoder is a bi-directional RNN in our model, we have multiple ways to select/compute on the generated hidden states to produce a sentence representation. Instead of using the last hidden state as the sentence representation as done in skip-thoughts BIBREF5 and SDAE BIBREF11 , we followed the idea proposed in BIBREF12 . They built a model for supervised training on the SNLI dataset BIBREF13 that concatenates the outputs from a global mean pooling over time and a global max pooling over time to serve as the sentence representation, and showed a performance boost on the SNLI task. BIBREF14 found that the model with global max pooling function provides stronger transferability than the model with a global mean pooling function does.

In our proposed RNN-CNN model, we empirically show that the mean+max pooling provides stronger transferability than the max pooling alone does, and the results are presented in the last two sections of Table TABREF10 . The concatenation of a mean-pooling and a max pooling function is actually a parameter-free composition function, and the computation load is negligible compared to all the heavy matrix multiplications in the model. Also, the non-linearity of the max pooling function augments the mean pooling function for constructing a representation that captures a more complex composition of the syntactic information.

Tying Word Embeddings and Word Prediction Layer

We choose to share the parameters in the word embedding layer of the RNN encoder and the word prediction layer of the CNN decoder. Tying was shown in both BIBREF15 and BIBREF16 , and it generally helped to learn a better language model. In our model, tying also drastically reduces the number of parameters, which could potentially prevent overfitting.

Furthermore, we initialise the word embeddings with pretrained word vectors, such as word2vec BIBREF3 and GloVe BIBREF4 , since it has been shown that these pretrained word vectors can serve as a good initialisation for deep learning models, and more likely lead to better results than a random initialisation.

Study of the Hyperparameters in Our Model Design

We studied hyperparameters in our model design based on three out of 10 downstream tasks, which are SICK-R, SICK-E BIBREF17 , and STS14 BIBREF18 . The first model we created, which is reported in Section SECREF2 , is a decent design, and the following variations didn't give us much performance change except improvements brought by increasing the dimensionality of the encoder. However, we think

it is worth mentioning the effect of hyperparameters in our model design. We present the Table TABREF21 in the supplementary material and we summarise it as follows:

1. Decoding the next sentence performed similarly to decoding the subsequent contiguous words.

2. Decoding the subsequent 30 words, which was adopted from the skip-thought training code, gave reasonably good performance. More words for decoding didn't give us a significant performance gain, and took longer to train.

3. Adding more layers into the decoder and enlarging the dimension of the convolutional layers indeed sightly improved the performance on the three downstream tasks, but as training efficiency is one of our main concerns, it wasn't worth sacrificing training efficiency for the minor performance gain.

4. Increasing the dimensionality of the RNN encoder improved the model performance, and the additional training time required was less than needed for increasing the complexity in the CNN decoder. We report results from both smallest and largest models in Table TABREF16 .

Experiment Settings

The vocabulary for unsupervised training contains the 20k most frequent words in BookCorpus. In order to generalise the model trained with a relatively small, fixed vocabulary to the much larger set of all possible English words, we followed the vocabulary expansion method proposed in BIBREF5 , which learns a linear mapping from the pretrained word vectors to the learnt RNN word vectors. Thus, the model benefits from the generalisation ability of the pretrained word embeddings.

The downstream tasks for evaluation include semantic relatedness (SICK, BIBREF17), paraphrase

detection (MSRP, BIBREF19), question-type classification (TREC, BIBREF20), and five benchmark sentiment and subjective datasets, which include movie review sentiment (MR, BIBREF21 , SST, BIBREF22), customer product reviews (CR, BIBREF23), subjectivity/objectivity classification (SUBJ, BIBREF24), opinion polarity (MPQA, BIBREF25), semantic textual similarity (STS14, BIBREF18), and SNLI BIBREF13 . After unsupervised training, the encoder is fixed, and applied as a representation extractor on the 10 tasks.

To compare the effect of different corpora, we also trained two models on Amazon Book Review dataset (without ratings) which is the largest subset of the Amazon Review dataset BIBREF26 with 142 million sentences, about twice as large as BookCorpus.

Both training and evaluation of our models were conducted in PyTorch, and we used SentEval provided by BIBREF14 to evaluate the transferability of our models. All the models were trained for the same number of iterations with the same batch size, and the performance was measured at the end of training for each of the models.

Related work and Comparison

Table TABREF16 presents the results on 9 evaluation tasks of our proposed RNN-CNN models, and related work. The “small RNN-CNN” refers to the model with the dimension of representation as 1200, and the “large RNN-CNN” refers to that as 4800. The results of our “large RNN-CNN” model on SNLI is presented in Table TABREF19 .

Our work was inspired by analysing the skip-thoughts model BIBREF5 . The skip-thoughts model successfully applied this form of learning from the context information into unsupervised representation learning for sentences, and then, BIBREF29 augmented the LSTM with proposed layer-normalisation

(Skip-thought+LN), which improved the skip-thoughts model generally on downstream tasks. In contrast, BIBREF11 proposed the FastSent model which only learns source and target word embeddings and is an adaptation of Skip-gram BIBREF3 to sentence-level learning without word order information. BIBREF9 applied a CNN as the encoder, but still applied LSTMs for decoding the adjacent sentences, which is called CNN-LSTM.

Our RNN-CNN model falls in the same category as it is an encoder-decoder model. Instead of decoding the surrounding two sentences as in skip-thoughts, FastSent and the compositional CNN-LSTM, our model only decodes the subsequent sequence with a fixed length. Compared with the hierarchical CNN-LSTM, our model showed that, with a proper model design, the context information from the subsequent words is sufficient for learning sentence representations. Particularly, our proposed small RNN-CNN model runs roughly three times faster than our implemented skip-thoughts model on the same GPU machine during training.

Proposed by BIBREF30 , BYTE m-LSTM model uses a multiplicative LSTM unit BIBREF31 to learn a language model. This model is simple, providing next-byte prediction, but achieves good results likely due to the extremely large training corpus (Amazon Review data, BIBREF26) that is also highly related to many of the sentiment analysis downstream tasks (domain matching).

We experimented with the Amazon Book review dataset, the largest subset of the Amazon Review. This subset is significantly smaller than the full Amazon Review dataset but twice as large as BookCorpus. Our RNN-CNN model trained on the Amazon Book review dataset resulted in performance improvement on all single-sentence classification tasks relative to that achieved with training under BookCorpus.

Unordered sentences are also useful for learning representations of sentences. ParagraphVec BIBREF32 learns a fixed-dimension vector for each sentence by predicting the words within the given sentence.

However, after training, the representation for a new sentence is hard to derive, since it requires optimising the sentence representation towards an objective. SDAE BIBREF11 learns the sentence representations with a denoising auto-encoder model. Our proposed RNN-CNN model trains faster than SDAE does, and also because we utilised the sentence-level continuity as a supervision which SDAE doesn't, our model largely performs better than SDAE.

Another transfer approach is to learn a supervised discriminative classifier by distinguishing whether the sentence pair or triple comes from the same context. BIBREF33 proposed a model that learns to classify whether the input sentence triplet contains three contiguous sentences. DiscSent BIBREF34 and DisSent BIBREF35 both utilise the annotated explicit discourse relations, which is also good for learning sentence representations. It is a very promising research direction since the proposed models are generally computational efficient and have clear intuition, yet more investigations need to be done to augment the performance.

Supervised training for transfer learning is also promising when a large amount of human-annotated data is accessible. BIBREF14 proposed the InferSent model, which applies a bi-directional LSTM as the sentence encoder with multiple fully-connected layers to classify whether the hypothesis sentence entails the premise sentence in SNLI BIBREF13 , and MultiNLI BIBREF36 . The trained model demonstrates a very impressive transferability on downstream tasks, including both supervised and unsupervised. Our RNN-CNN model trained on Amazon Book Review data in an unsupervised way has better results on supervised tasks than InferSent but slightly inferior results on semantic relatedness tasks. We argue that labelling a large amount of training data is time-consuming and costly, while unsupervised learning provides great performance at a fraction of the cost. It could potentially be leveraged to initialise or more generally augment the costly human labelling, and make the overall system less costly and more efficient.

Discussion

In BIBREF11 , internal consistency is measured on five single sentence classification tasks (MR, CR, SUBJ, MPQA, TREC), MSRP and STS-14, and was found to be only above the “acceptable” threshold. They empirically showed that models that worked well on supervised evaluation tasks generally didn't perform well on unsupervised ones. This implies that we should consider supervised and unsupervised evaluations separately, since each group has higher internal consistency.

As presented in Table TABREF16 , the encoders that only sum over pretrained word vectors perform better overall than those with RNNs on unsupervised evaluation tasks, including STS14. In recent proposed log-bilinear models, such as FastSent BIBREF11 and SiameseBOW BIBREF37 , the sentence representation is composed by summing over all word representations, and the only tunable parameters in the models are word vectors. These resulting models perform very well on unsupervised tasks. By augmenting the pretrained word vectors with a weighted averaging process, and removing the top few principal components, which mainly encode frequently-used words, as proposed in BIBREF38 and BIBREF39 , the performance on the unsupervised evaluation tasks gets even better. Prior work suggests that incorporating word-level information helps the model to perform better on cosine distance based semantic textual similarity tasks.

Our model predicts all words in the target sequence at once, without an autoregressive process, and ties the word embedding layer in the encoder with the prediction layer in the decoder, which explicitly uses the word vectors in the target sequence as the supervision in training. Thus, our model incorporates the word-level information by using word vectors as the targets, and it improves the model performance on STS14 compared to other RNN-based encoders.

BIBREF38 conducted an experiment to show that the word order information is crucial in getting better results on supervised tasks. In our model, the encoder is still an RNN, which explicitly utilises the word order information. We believe that the combination of encoding a sentence with its word order information

and decoding all words in a sentence independently inherently leverages the benefits from both log-linear models and RNN-based models.

Conclusion

Inspired by learning to exploit the contextual information present in adjacent sentences, we proposed an asymmetric encoder-decoder model with a suite of techniques for improving context-based unsupervised sentence representation learning. Since we believe that a simple model will be faster in training and easier to analyse, we opt to use simple techniques in our proposed model, including 1) an RNN as the encoder, and a predict-all-words CNN as the decoder, 2) learning by inferring subsequent contiguous words, 3) mean+max pooling, and 4) tying word vectors with word prediction. With thorough discussion and extensive evaluation, we justify our decision making for each component in our RNN-CNN model. In terms of the performance and the efficiency of training, we justify that our model is a fast and simple algorithm for learning generic sentence representations from unlabelled corpora. Further research will focus on how to maximise the utility of the context information, and how to design simple architectures to best make use of it.

Supplemental Material

Table TABREF21 presents the effect of hyperparameters.

Decoding Sentences vs. Decoding Sequences

Given that the encoder takes a sentence as input, decoding the next sentence versus decoding the next fixed length window of contiguous words is conceptually different. This is because decoding the subsequent fixed-length sequence might not reach or might go beyond the boundary of the next

sentence. Since the CNN decoder in our model takes a fixed-length sequence as the target, when it comes to decoding sentences, we would need to zero-pad or chop the sentences into a fixed length. As the transferability of the models trained in both cases perform similarly on the evaluation tasks (see rows 1 and 2 in Table TABREF21), we focus on the simpler predict-all-words CNN decoder that learns to reconstruct the next window of contiguous words.

Length of the Target Sequence TT

We varied the length of target sequences in three cases, which are 10, 30 and 50, and measured the performance of three models on all tasks. As stated in rows 1, 3, and 4 in Table TABREF21 , decoding short target sequences results in a slightly lower Pearson score on SICK, and decoding longer target sequences lead to a longer training time. In our understanding, decoding longer target sequences leads to a harder optimisation task, and decoding shorter ones leads to a problem that not enough context information is included for every input sentence. A proper length of target sequences is able to balance these two issues. The following experiments set subsequent 30 contiguous words as the target sequence.

RNN Encoder vs. CNN Encoder

The CNN encoder we built followed the idea of AdaSent BIBREF41 , and we adopted the architecture proposed in BIBREF14 . The CNN encoder has four layers of convolution, each followed by a non-linear activation function. At every layer, a vector is calculated by a global max-pooling function over time, and four vectors from four layers are concatenated to serve as the sentence representation. We tweaked the CNN encoder, including different kernel size and activation function, and we report the best results of CNN-CNN model at row 6 in Table TABREF21 .

Even searching over many hyperparameters and selecting the best performance on the evaluation tasks (overfitting), the CNN-CNN model performs poorly on the evaluation tasks, although the model trains much faster than any other models with RNNs (which were not similarly searched). The RNN and CNN are both non-linear systems, and they both are capable of learning complex composition functions on words in a sentence. We hypothesised that the explicit usage of the word order information will augment the transferability of the encoder, and constrain the search space of the parameters in the encoder. The results support our hypothesis.

The future predictor in BIBREF9 also applies a CNN as the encoder, but the decoder is still an RNN, listed at row 11 in Table TABREF21 . Compared to our designed CNN-CNN model, their CNN-LSTM model contains more parameters than our model does, but they have similar performance on the evaluation tasks, which is also worse than our RNN-CNN model.

Dimensionality

Clearly, we can tell from the comparison between rows 1, 9 and 12 in Table TABREF21 , increasing the dimensionality of the RNN encoder leads to better transferability of the model.

Compared with RNN-RNN model, even with double-sized encoder, the model with CNN decoder still runs faster than that with RNN decoder, and it slightly outperforms the model with RNN decoder on the evaluation tasks.

At the same dimensionality of representation with Skip-thought and Skip-thought+LN, our proposed RNN-CNN model performs better on all tasks but TREC, on which our model gets similar results as other models do.

Compared with the model with larger-size CNN decoder, apparently, we can see that larger encoder size helps more than larger decoder size does (rows 7,8, and 9 in Table TABREF21).

In other words, an encoder with larger size will result in a representation with higher dimensionality, and generally, it will augment the expressiveness of the vector representation, and the transferability of the model.

Experimental Details

Our small RNN-CNN model has a bi-directional GRU as the encoder, with 300 dimension each direction, and the large one has 1200 dimension GRU in each direction. The batch size we used for training our model is 512, and the sequence length for both encoding and decoding are 30. The initial learning rate is INLINEFORM0 , and the Adam optimiser BIBREF40 is applied to tune the parameters in our model.

Results including supervised task-dependent models

Table TABREF26 contains all supervised task-dependent models for comparison.