RNN on Machine Reading Comprehension

- Bi-Directional Attention Flow model -

Implementor

NGUYEN Hong Thinh

Vietnam National University, Hanoi
University of Engineering and Technology
Faculty of Electronics and Telecommunications

Contents

Chapter 1

Introduction

Although end-to-end deep neural network have gained popularity in the last few years and have been successful in several Natural Language Processing (NLP) tasks such as sequence labeling [6], the task of reading comprehension remains a challenging task for NLP researchers. Reading or machine comprehension is a special task of Question Answering (QA), where the machine is given a query about a given context and is required to predict the answer. Such problems gain significant popularity over the years not only because of their vast applications, but also theoretical values for language and neural network research. This task is challenging because the system must be able to model complex interactions between the question and the context paragraph and must be able to do several complex skills such as coreference resolution, commonsense reasoning, causal relations, and spatiotemporal relations. To solve this, usually some attention mechanism is adopted to focus on only a small portion of the context. This task also requires modeling of the interaction between query and context. To motivate this line of research, Stanford NLP group released the SQuAD dataset [8], which consists of 100K question-answer pairs, along with a context paragraph for each pair. There is also a public leader board available. The state of the art is already very competitive, as there are many methods that are approaching human level performance.

An Example of SQuAD dataset context One of the most famous people born in Warsaw was Maria Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the Nobel Prize. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.

Example 1.1 (A pair of questions/answers from the above context)
What was Maria Curie the first female recipient of?
Ground Truth Answers: Nobel Prize

Example 1.2 (Another pair of questions/answers)
What year was Casimir Pulaski born in Warsaw?
Ground Truth Answers: 1745

Example 1.3 (Another pair of questions/answers)
Who was one of the most famous people born in Warsaw?
Ground Truth Answers: Maria Skłodowska-Curie

Example 1.4 (Another pair of questions/answers)
How old was Chopin when he moved to Warsaw with his family?
Ground Truth Answers: seven months old

Problem Definition We define our problem as the following: Given word sequence of context with length T, $p = \{p_1, p_2, ..., p_T\}$ and question with length J, $q = \{q_1, q_2, ..., q_J\}$, the model needs to learn a function $f : (p, q) \rightarrow \{a_s, a_e\}$, with the condition $1 \leq a_s \leq a_e \leq T$. $\{a_s, a_e\}$ is a pair of scalar indices pointing to the start position and end position respectively in the context p, indicating the answer to the question q.

The report is constructed as following: chapter 2 presents several popular neural network structures that have been using in NLP field: recurrent neural network, long short term memory, attention mechanism... Chapter 3 will forcus on BiDAF, the most popular model of the leaderboard of SQuAD. Finally, chapter 4 will conclude the report with some remarks.

Chapter 2

Background

2.1 Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones. Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist. These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. The RNN handles the variable-length sequence by having a recurrent hidden state whose activation at each time is dependent on that of the previous time.

2.2 Long Short-Term Memory

In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning... Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much much better than the standard version. Almost all exciting results based on recurrent neural networks

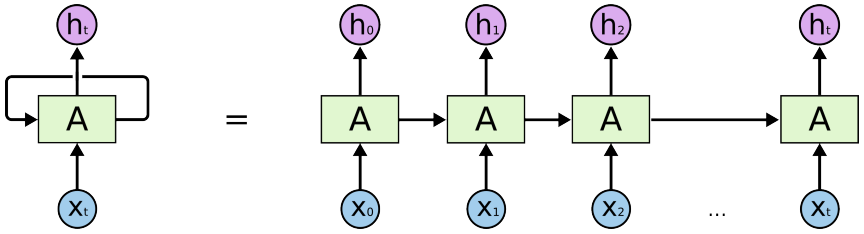


Figure 2.1: Recurrent Neural Networks unrolled

are achieved with them.

2.2.1 The Problem of Long-Term Dependencies

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the [...]," we don't need any further context – it's pretty obvious the next word is going to be "sky". In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.

But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in Hanoi...I speak fluent [...]." Recent information suggests that the next word is probably the name of a language (which is *Vietnamese*), but if we want to narrow down which language, we need the context of Hanoi, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information. In theory, RNNs are absolutely capable of handling such "long-term dependencies." A human could carefully pick parameters for them to solve toy problems of this form. In practice, however, RNNs don't seem to be able to learn them. In fact, it has been observed by, e.g., Bengio et al. [2] that it is difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish (most of the time) or explode (rarely, but with severe effects).

One of the solution is to design a more sophisticated activation function than a usual activation function, consisting of affine transformation followed by a simple element-wise nonlinearity by using gating units. The earliest attempt in this direction resulted in an activation function, or a recurrent unit, called a long short-term memory (LSTM) unit.

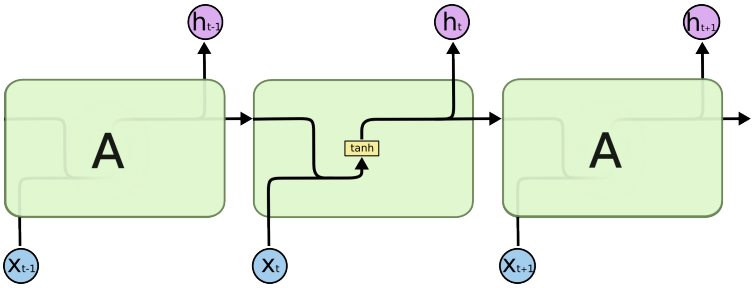


Figure 2.2: a standard RNN

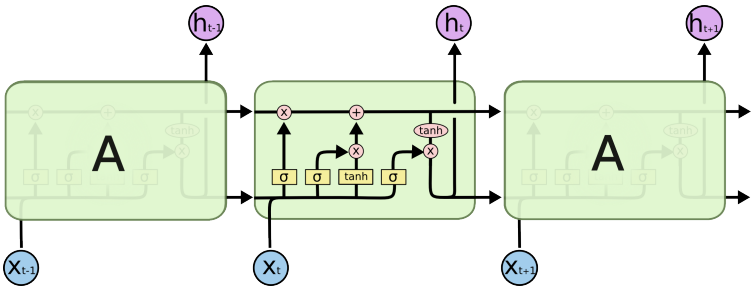


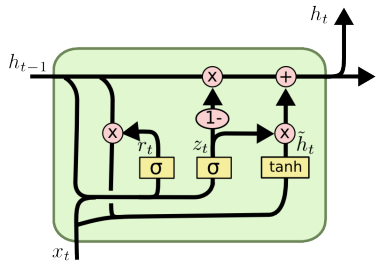
Figure 2.3: a LSTM

2.2.2 Long Short-Term Memory Networks

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber [5], and were refined and popularized by many people. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer as in figure 2.2.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



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

Figure 2.4: a Gated Recurrent Unit

2.3 Gated Recurrent Unit

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. [3] to make each recurrent unit to adaptively capture dependencies of different time scales. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cells. It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

GRU is relatively new, and its performance is on par with LSTM, but computationally more efficient (less complex structure as pointed out). So we are seeing it being used more and more. Chung et al. [4] made an excellent and comprehensive comparison between LSTM and GRU.

2.4 Attention Mechanism

Attention Mechanisms in Neural Networks are (very) loosely based on the visual attention mechanism found in humans. Human visual attention is well-studied and while there exist different models, all of them essentially come down to being able to focus on a certain region of an image with high "resolution" while perceiving the surrounding image in "low resolution", and then adjusting the focal point over time.

Attention mechanism was first introduced in Neural Machine Translation (NMT) [1], where it generate a translation word based on the whole original sentence states, not just the last state. The idea is to let every step of an RNN pick information to look at from some larger collection of information. For example, if you are using an RNN to create a caption describing an image, it might pick a part of the image to look at for every word it outputs. The idea is then applied to reading comprehension, allowing the model to select a subset of context paragraph and a subset of question that are most relevant. That way, the model can use the most

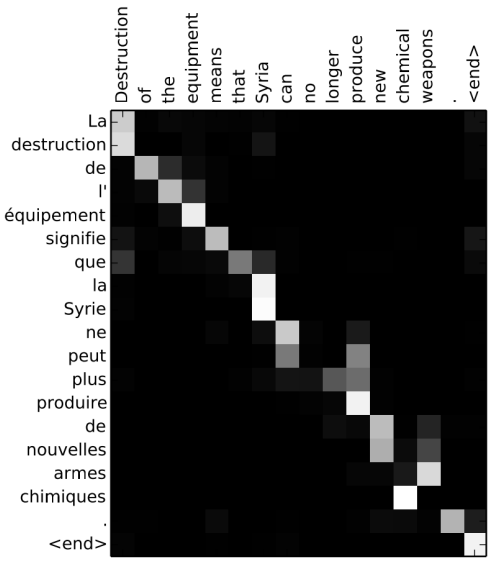


Figure 2.5: the attention weight matrix when translating a sentence

relevant information to give a better answer.

Classically, most NMT systems work by encoding the source sentence (e.g. a German sentence) into a vector using a Recurrent Neural Network, and then decoding an English sentence based on that vector, also using a RNN. The decoder is supposed to generate a translation solely based on the last hidden state from the encoder. This vector must encode everything we need to know about the source sentence. It must fully capture its meaning. In more technical terms, that vector is a sentence embedding. In fact, if you plot the embeddings of different sentences in a low dimensional space using PCA for dimensionality reduction, you can see that semantically similar phrases end up close to each other.

Still, it seems somewhat unreasonable to assume that we can encode all information about a potentially very long sentence into a single vector and then have the decoder produce a good translation based on only that. Let's say your source sentence is 50 words long. The first word of the English translation is probably highly correlated with the first word of the source sentence. But that means decoder has to consider information from 50 steps ago, and that information needs to be somehow encoded in the vector. Recurrent Neural Networks are known to have problems dealing with such long-range dependencies. In theory, architectures like LSTMs should be able to deal with this, but in practice long-range dependencies are still problematic. For example, researchers have found that reversing the source sequence (feeding it backwards into the encoder) produces significantly better results because it shortens the path from the decoder to the relevant parts of the encoder. Similarly, feeding an input sequence twice also seems to help a network

to better memorize things.

With an attention mechanism we no longer try encode the full source sentence into a fixed-length vector. Rather, we allow the decoder to "attend" to different parts of the source sentence at each step of the output generation. Importantly, we let the model learn what to attend to based on the input sentence and what it has produced so far. So, in languages that are pretty well aligned (like English and German) the decoder would probably choose to attend to things sequentially. Attending to the first word when producing the first English word, and so on. A big advantage of attention is that it gives us the ability to interpret and visualize what the model is doing.

Chapter 3

Bi-Directional Attention Flow model (BiDAF)

In this chapter, we cover BiDAF [9], the most popular model of the leaderboard of SQuAD. BIDAF is a multi-stage hierarchical process that represents the context at different levels of granularity and uses a bi-directional attention flow mechanism to achieve a query aware context representation without early summarization.

BIDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation. The attention mechanism offers some improvements to the previously popular attention paradigms. First, the attention layer is not used to summarize the context paragraph into a fixed-size vector. Instead, the attention is computed for every time step, and the attended vector at each time step, along with the representations from previous layers, is allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization. Second, the authors used a memory-less attention mechanism, that is the attention at each time step is a function of only the query and the context paragraph at the current time step and does not directly depend on the attention at the previous time step. This simplification leads to the division of labor between the attention layer and the modeling layer. It forces the attention layer to focus on learning the attention between the query and the context, and enables the modeling layer to focus on learning the interaction within the query-aware context representation (the output of the attention layer). It also allows the attention at each time step to be unaffected from incorrect attendances at previous time steps. The experiments show that memory-less attention gives a clear advantage over dynamic attention. Third, attention mechanisms are used in both directions, query-to-context and context-to-query, which provide complimentary information to each other.

BiDAF model is a hierarchical multi-stage process and consists of six layers (Figure 3.1):

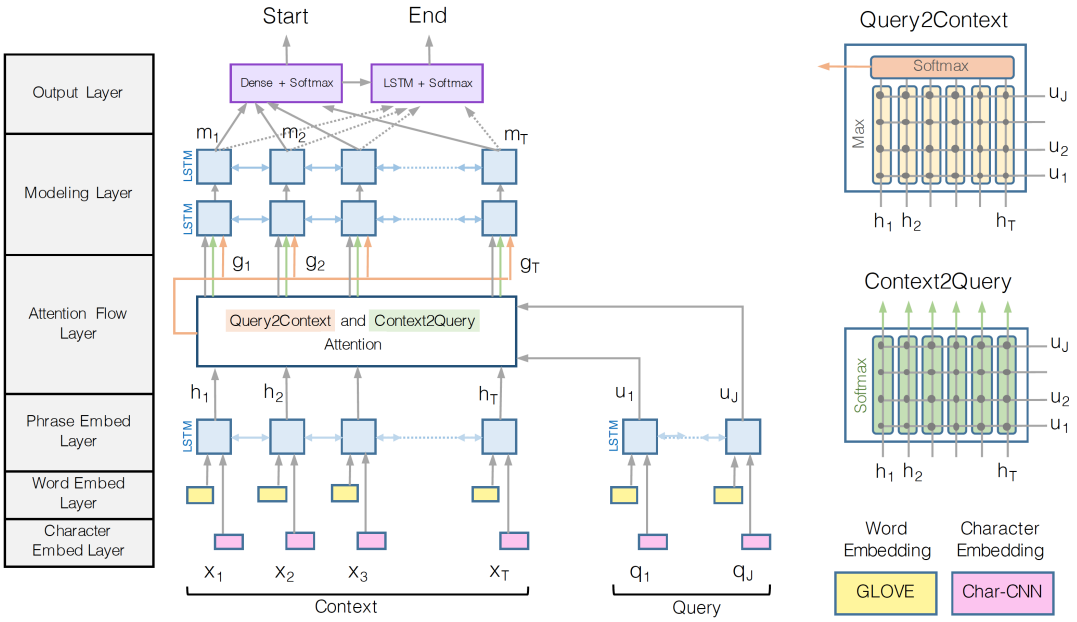


Figure 3.1: Bi-Directional Attention Flow model

- Character Embedding Layer maps each word to a vector space using character-level CNNs.
- Word Embedding Layer maps each word to a vector space using a pre-trained word embedding model.
- Contextual Embedding Layer utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context.
- Attention Flow Layer couples the query and context vectors and produces a set of queryaware feature vectors for each word in the context.
- Modeling Layer employs a Recurrent Neural Network to scan the context.
- Output Layer provides an answer to the query.

Character Embedding Layer Character embedding layer is responsible for mapping each word to a high-dimensional vector space. Let $\{x_1, ...x_T\}$ and $\{q_1, ...q_J\}$ represent the words in the input context paragraph and query, respectively. The characterlevel embedding of each word is obtained via Convolutional Neural Networks (CNN). Characters are embedded into vectors, which can be considered as 1D inputs to the CNN, and whose size is the input channel size of the CNN. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word.

Word Embedding Layer Word embedding layer also maps each word to a high-dimensional vector space. The pre-trained word vectors, GloVe [7], is often used to obtain the fixed word embedding of each word. The concatenation of the character and word embedding vectors is passed to a two-layer Highway Network [10]. The outputs of the Highway Network are two sequences of d-dimensional vectors, or more conveniently, two matrices: $X \in R^{d \times T}$ for the context and $Q \in R^{d \times J}$ for the query.

Contextual Embedding Layer a Long Short-Term Memory Network (LSTM) is used on top of the embeddings provided by the previous layers to model the temporal interactions between words. The outputs of the two LSTMs is concatenated. Hence we obtain $H \in R^{2d \times T}$ from the context word vectors X, and $U \in R^{2d \times T}$ from query word vectors Q.

Attention Flow Layer Attention flow layer is responsible for linking and fusing information from the context and the query words. Unlike previously popular attention mechanisms, the attention flow layer is not used to summarize the query and context into single feature vectors. Instead, the attention vector at each time step, along with the embeddings from previous layers, are allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization.

The inputs to the layer are contextual vector representations of the context H and the query U. The outputs of the layer are the query-aware vector representations of the context words, G, along with the contextual embeddings from the previous layer.

Modeling Layer The input to the modeling layer is G, which encodes the query-aware representations of context words. The output of the modeling layer captures the interaction among the context words conditioned on the query. This is different from the contextual embedding layer, which captures the interaction among context words independent of the query. We use two layers of bi-directional LSTM, with the output size of d for each direction. Hence we obtain a matrix $M \in R^{2d \times T}$, which is passed onto the output layer to predict the answer. Each column vector of M is expected to contain contextual information about the word with respect to the entire context paragraph and the query.

Chapter 4

Conclusion

In this report, we have presented in details the popular Bi-Directional Attention Flow model which represents the context at different level and combined the context-to-query and query-to-context direction attention. All neccesary background knowledge of general Recurrent Neural Network is also discussed. Currently, at least 8 out of top-40 of SQuAD leaderboards directly use BiDAF as the main component. Table 4.1 compares human performance with some BiDAF-based models on the dataset SQuAD.

Model	EM	F1
Human Performance	82.304	91.221
BiDAF + Self Attention + ELMo (ensemble)	81.003	87.432
BiDAF + Self Attention + ELMo (single model)	78.580	85.833
SEDT+BiDAF (ensemble)	73.723	81.530
BiDAF (ensemble)	73.744	81.525
BiDAF (single model)	67.974	77.323

Table 4.1: BiDAF-related models Performance

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014). arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[3] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[4] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[5] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[6] Xuezhe Ma and Eduard H. Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *CoRR* abs/1603.01354 (2016). arXiv: 1603.01354. URL: http://arxiv.org/abs/1603.01354.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[8] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

[9] Min Joon Seo et al. "Bidirectional Attention Flow for Machine Comprehension". In: *CoRR* abs/1611.01603 (2016). arXiv: 1611.01603. URL: http://arxiv.org/abs/1611.01603.

[10] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).