Recurrent Neural Network Encoder with Attention for Community Question Answering

Wei-Ning Hsu
wnhsu@mit.edu

Yu Zhang
yzhang87@mit.edu

Abstract

We apply a general recurrent neural networks (RNNs) encoder framework to address the community question answering (CQA) selection tasks. Our approach does not rely on any linguistic tools and can be applied to different languages or domains. Further improvements are also observed after we extend the RNN encoders with neural attention mechanism that encourages reasoning over the entire sequences. To deal with practical issues such as data sparsity and imbalanced labels, we also apply various techniques such as transfer learning and multitask learning. Our experiments on SemEval-2016 CQA task show 10% improvement on MAP compared to the information retrieval-based approach and paralleled results to a strong handcrafted feature-based method. Code can be found here[1].

1 Introduction

Community question answering (CQA) is a paradigm which provides forums for users to ask or answer questions on any topic with merely any restrictions. In the past decade, these websites have attracted a great number of users, and have accumulated a large collection of question-comment threads generated by these users. However, the low restriction results in high variance of answer quality, which makes it time-consuming to search for useful information from existing content. It would be valuable to automate the procedure of ranking related questions and comments for users given a new question or when looking for solutions from comments of an existing question.

Automation of the above procedure can be divided into three tasks: question-comment relevance (Task A), question-question relevance (Task B), and question-external comment relevance (Task C). One might think that classic retrieval models like language models for information retrieval (Zhai and Lafferty, 2004) could solve these. However, a big challenge for CQA tasks is that users are used to expressing similar meanings with different words, which creates gaps when matching questions based on common words. Other challenges include informal usage of language, highly diverse content of comments, and length variation of both questions and comments.

To overcome these issues, most of previous work (e.g. SemEval 2015(Nakov et al., 2015)) relied heavily on additional features and reasoning capabilities. In (Rocktäschel et al., 2015), a neural attention based model was proposed for automatically recognizing entailment relations between pairs of natural language sentences. In this study, we first modify this model for all three CQA tasks. We also extend this framework into a jointly trained model when the external resources are available, e.g. how to select external comment when we also know the question which the external comment replies to (Task C).

Our ultimate objective is to classify relevant questions and comments without complicated handcrafted features. By applying recurrent neural network-based encoders, we avoid heavy engineered features and learn the representation automatically. In addition, attention mechanism enhances encoders with the ability to attend to the past output directly. This becomes more helpful when encoding longer sequences since we no more need to compress entire information into a fixed-length representation.

However, in our view, existing CQA corpus with golden labels are generally too small for training a decent end-to-end neural network. To address this, we also apply transfer learning through pretrain the recurrent systems on other

[1] https://github.mit.edu/wnhsu/rnn_enc

corpora as well as generate more instances from existing CQA corpus.

2 Method

In this section, we first discuss long short-term memory (LSTM) units and its extension with attention mechanism. Next, we explain how we can encode a pair of sentences into a dense vector for predicting relationship using LSTM with attention mechanism. Finally, we propose our models to predict question-question similarity, question-comment similarity and question-external comment similarity.

2.1 LSTM

LSTMs have shown great success in many different fields. An LSTM unit contains a memory cell with self-connections as well as three multiplicative gate to control the information flow. Given input vector x_t, previous hidden outputs h_{t-1}, and previous cell state c_{t-1}, LSTM units are operated as follows:

$$X = \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \tag{1}$$

$$i_t = \sigma(\mathbf{W_{iX}}X + \mathbf{W_{ic}}c_{t-1} + \mathbf{b_i}) \tag{2}$$

$$f_t = \sigma(\mathbf{W_{fX}}X + \mathbf{W_{fc}}c_{t-1} + \mathbf{b_f}) \tag{3}$$

$$o_t = \sigma(\mathbf{W_{oX}}X + \mathbf{W_{oc}}c_{t-1} + \mathbf{b_o}) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(\mathbf{W_{cX}}X + \mathbf{b_c}) \tag{5}$$

$$h_t = o_t \odot tanh(c_t) \tag{6}$$

Above i_t, f_t, o_t are input gate, forget gate, and output gate respectively. Sigmoid function $\sigma()$ is a soft gate function controlling the amount of information to flow. Ws and bs in bold are parameters to learn in the model.

2.2 Neural Attention

Traditional RNN encoder-decoder approach (Sutskever et al., 2014) first encodes a sequence of arbitrary length into a fixed-length dense vector, which can further be used as input to other classification models or to initialize the hidden state of the secondary decoder. This could be problematic when RNN fail to compress all the necessary information into a single vector.

Neural attention model (Bahdanau et al., 2014) (Cho et al., 2014) is recently proposed to alleviate this issue by enabling the network to attend to past outputs when decoding. Thus, the encoder no longer need to squeeze entire sequence into one vector; instead, it encodes information into a sequence of vectors and chooses a subset of the vectors adaptively when decoding.

2.3 Predicting Relationship of Object Pair with Attention Model

In our tasks, the pair of objects are (question, question) or (question, comment), and relationship is relevant/irrelevant. Left part of Figure 1 shows one intuitive way to predict relationship using RNNs. The parallel LSTM tries to encode two objects independently and concatenate their representations as input to a feed-forward neural network (FNN) with softmax output for classification.

The representations of two objects are generated independently in this manner. However, we are more interested in the relationship instead of object representations alone. Therefore, we design an serialized LSTM-encoder model in the right part of Figure 1 similar to that in (Rocktäschel et al., 2015), but in addition allowing augmented feature input to the FNN classifier.

Figure 2 illustrates our framework in detail. The first LSTM reads one object, and passes information through hidden units to the second LSTM. The second LSTM then reads the other object and generates the representation of this pair after finish reading. We build another feed-forward neural network that takes this representation as input to classify the relationship of this pair.

By augmenting attention mechanism to the encoder, we allow the second LSTM to attend to the sequence of output vectors from the first LSTM, and hence generate a weighted representation of first object according to both objects. Let h_N be the last output of second LSTM and $M = [h_1, h_2, \cdots, h_L]$ be the sequence of output vectors of the first object. The weighted representation of the first object is

$$h' = \sum_{i=1}^{L} \alpha_i h_i \tag{7}$$

The weight is computed by

$$\alpha_i = \frac{exp(a(h_i, h_N))}{\sum_{j=1}^{L} exp(a(h_j, h_N))} \tag{8}$$

where $a()$ is the importance model that produces higher score to (h_i, h_N) if h_i is useful to determine the object pair's relationship. We parametrize this model using another FNN. Note that in our framework, we also allow using other augmented feature

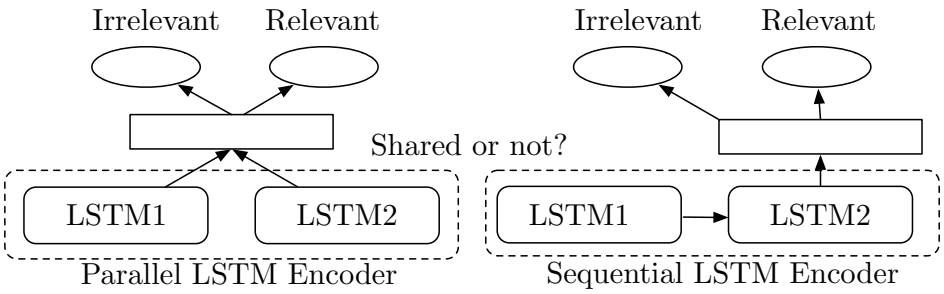


Figure 1: RNN encoder for related question/comment selection

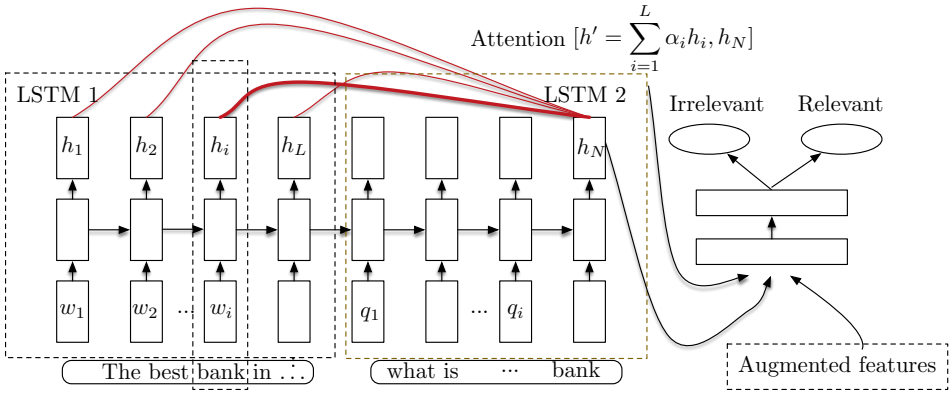


Figure 2: Neural attention for related question/comment selection

(e.g. the ranking score from IR system) to enhance the classifier. So the final input to the classifier will be h_N and h' as well as augmented features.

2.4 Modeling Question-External Comment Relationship

For task C, in addition to an original question (oriQ) and an external comment (relC), the question which relC commented on is also given (relQ). To incorporate this extra information, we proposed a multitask learning framework which jointly learns to predict the relationships of the three pairs (oriQ/relQ, oriQ/relC, relQ/relC).

Figure 3 shows our framework: the lower part are three separate serialized LSTM-encoders for three pairs respectively, whereas the upper part is an FNN that takes as input the concatenation of the outputs of three encoders, and then predicting relationships of all three pairs. More specifically, on top of the last hidden layer are three softmax layers where each one is designated to predict the relationship of one pair.

For the overall loss function, we put together three loss functions using a heuristic weight vector β in the sense of allocating higher weight to the main task, which is oriQ-relC relationship prediction here, as follows:

$$\mathcal{L} = \beta_1 \mathcal{L}_1 + \beta_2 \mathcal{L}_2 + \beta_3 \mathcal{L}_3 \quad (9)$$

By doing so, we expect the related tasks could improve the main task by using the commonality among all tasks.

3 Experiments

We evaluate our approach on all three selection tasks. We use the CQA datasets provided by Semeval 2016 task. The CQA data is organized as follows: there are 267 original questions, each question has 10 related question and each related question has 10 comments. Therefore, for task A, there are a total number of 26700 question-comment pairs). For task B, there are 2670 question-question pairs. For task C, there are 26700 question-comment pairs. The test dataset includes 50 questions, 500 related questions and 5000 comments which is non-overlapped with training set. To evluate the performance, we use mean average score (MAP) and F1 score.

Baseline System: Figure 4 shows our baseline systems. The IR-based system is scored

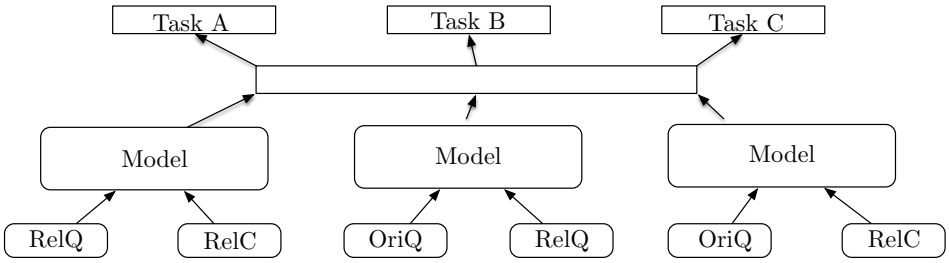


Figure 3: Joint learning for external comment selection

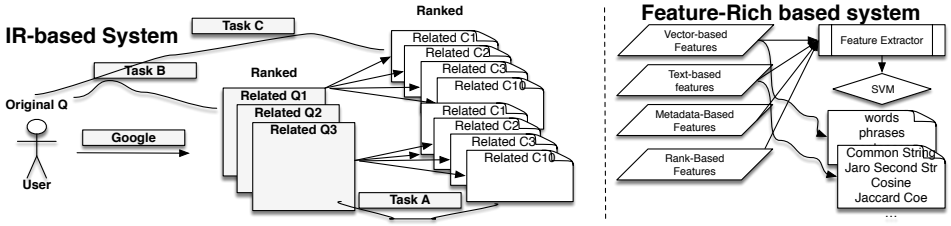


Figure 4: IR-based system and feature-rich based system

by Google search engine. For each question-comment pair or question-question pairs, we use Google's rank to calculate the MAP. There is no training using the target data but we can expect Google already use lots of external resource to give these ranks. The feature-riched system is proposed by (Belinkov et al., 2004) in SemEval-2015. In this approach, they compute *text-based*, *vector-based*, *metadata-based* and *rank-based features* from the pre-processed data. The features are used for a linear SVM for comment selection. As we can see, this system already includes traditional handcrafted features and some RNN-based features (vector-based). It also includes the information from IR system (ranked-based). So we believe it is a strong base to compare with our model.

RNN encoder: our system is based on Theano (Bastien et al., 2012; Bergstra et al., 2010). Table 1 gives a list of hyper-parameters we tried. As suggested by (Greff et al., 2015), the hyper-parameters for LSTM can be tuned independently. We tune each parameter separately on a dev set (split from the training set) and simply pick the best one. Our experiments shows that using word embedding from Google-News only gives slightly gain, but fixing the embedding will degrades the performance a lot. Also, using separate parameters for LSTMs is better than sharing. For the optimization method, AdaDelta converged faster but AdaGrad gives better performance. Note that all the parameter is tuned on Task A and we simply apply it to Task B and C. This is for saving the computation power and also Task A is more well-defined compared to B and C (in terms of dataset size and label balance).

Embedding	**init** or random, fix or **update**
Two LSTM	shared or **not**
#cells for LSTM	64, **128**, 256
# nodes for MLP	128, **256**
Optimizer	**AdaGrad**, AdaDelta, SGD
learning rate	0.001,0.01,0.1
Regularizer	**Dropout**, L2 regularization
Dropout rate	0.0, 0.2, 0.3, **0.4**, 0.5
L2	0, 0.001, **0.0001**, 0.00001

Table 1: The hyper-parameters we tuned. The bolder one is the final parameter set we picked.

3.1 Preliminary Results

Table 2 shows the initial results using RNN encoder for different tasks. We can observe that the attention model always gets better results than the RNN without attention, especially for task C. However, it can be observed our model get very low F1 score. For task B, it is even worse than the random baseline. The reason is for task B, there are only 2670 pairs for train which is highly limited to train a reasonable neural network. For task C, the problem is highly imbalanced data. Because the comments is not directly commented on the original question, there are more than 90% com-

	Task A		Task B		Task C	
Model	MAP	F1	MAP	F1	MAP	F1
Random	0.4860	0.5004	0.5595	0.4691	0.1383	0.1277
Parallel LSTM	0.6123	0.6091	0.5553	0.4087	0.2413	0.0057
Seq LSTM	0.6175	0.6063	0.5620	0.4299	0.2356	0.0115
w/ Attention	0.6239	0.6323	0.5723	0.4334	0.2837	0.1449

Table 2: The RNN encoder results for different tasks

ments which is labeled as irrelevant to the original question. The low F1 (with high precision and low recall) means our system is tend to output false to every comments. In the following section, we will investigate how to improve the performance for task B and C.

3.2 Initialize the Parameters from More Robust Models

One way to enhance the model trained on limited data is using external data to pretrain the neural network. In this study, we tried two different dataset to augment the results.

- Cross-domain: Stanford natural language inference (SNLI) corpus (Bowman et al., 2015). This corpus has huge amount of cleaned premise and hypothesis pairs. But the problem is that it is a different task. The relation between premise and hypothesis may be similar to the relation between question and comment, but it is different.

- In-domain: since task A seems has reasonable performance and the network is also well-trained, we could use it directly to initialize task B.

To utilize the data, we first train the model on each auxiliary data (SNLI or Task A) and then remove the softmax layer. After that, we retrain the network using target data of which softmax layer is randomly initialized.

For task A, the SNLI can not improve MAP or F1 scores. Actually it slightly hurts the performance. It is probably because the domain is different. Further investigation is needed: for example, we could only take the parameter for embedding layers. For task B, the SNLI gives slight improvement on MAP (0.2%) and Task A could give (1.2%) on top of that. No improvement is observed on F1. For task C, pretrained by task A is also better than the one pretrained by SNLI (task A is 1% better than the baseline and SNLI is almost the same).

In summary, the in-domain pretraining seems better, but overall, the improvement is less than we expected especially for task B, which only has very limited target data. We will not make a conclusion here since more investigation is needed for this section.

3.3 Multitask Learning

As we mentioned in Section 2.4, we also applied a multitask learning framework which jointly learns to predict the relationships of three tasks. We set 0.8 for the main task (task C) and 0.1 for the other auxiliary tasks. The MAP score has not improved but F1 goes up to 0.1617. This is probably because other tasks have more balanced labels which improves the shared parameters for task C.

3.4 Augmented data

There are tons of external question-answer pairs that could be used in our tasks. For example:

- WebQuestion: was introduced by the authors of SEMPRE system (Berant et al., 2013) and it contains 3,778 train and 2,032 test.

- TREC: was introduced for the purpose of evaluating information retrieval QA system (Voorhees and Tice, 2000) and it contains 962 train and 517 test

- WikiAnswers: is a set of questions that were randomly sampled from a crawl of WikiAnswers and it contains 1334 train and 7310 test

- The SimpleQuestions dataset [2]: consists of a total of 108,442 questions written in natural language by human English-speaking annotators.

All of them are positive example for our task and we can easily sample negative examples from it.

[2] http://fb.ai/babi.

But initial experiments shows that the system is very easy to overfit into these obvious negative examples. We think the reason is our negative examples is non-informative for our task which just introduce some noise. So next step we will try to use the confidence score from our model to filter the negative example, e.g. only use the more confused negative examples.

Since the external data seems to hurt the performance, we try to use the in-domain pairs to enhance task B and task C.

For task B, if relative question 1 (rel1) and relative question 2 (rel2) are all relevant to the same original question, then we add a positive sample (rel1, rel2, 1). If one of rel1 and rel2 is irrelevant and the other is relevant, we add a negative sample (rel1, rel2, 0). After doing this, the samples of task B has increased from 2670 to 11810. By applying this method, the MAP score increased slightly from 0.5723 to 0.5789 but the F1 score got huge improvement, from 0.4334 to 0.5860.

For task C, we could use task A's data directly. The observation is very similar, slightly improvement on MAP but huge improvement on F1 score from 0.1449 to 0.2064.

3.5 Augmented features

Note that we have the original rank from IR system. To further enhance the system, we take the one hot vector of ranking as a additional feature into the MLP classifier. Table 3 shows the results. We can see huge improvement for task B and C. The F1 score for task A is slightly degrades but MAP also been improved. This might be because task A already have lots of training data.

3.6 Compare to Other Systems

Table 4 gives the final comparison between different models (we only list the MAP score because it is the official score for the challenge). Since the two baseline did not use any additional data, in this table our system also use the provided training data only. For task A, we can see if we have enough training data our single system already performed better than a very strong feature-rich based system. For task B, since only limited training data is given, both feature-rich based system and our system are worse than the IR system. For task C, our system also got comparable results with feature-rich based system. If we do a simple system combination (average the rank score) between our system and IR system, the combined system will give huge gain on task B and task C[3]. This implied our system is complimentary with the IR system. Thus, in future, we could also integrate with some useful features from the feature-rich based system into our RNN encoder.

3.7 Qualitative Analysis of Attention Mechanism

In addition to quantitative analysis, it is natural to qualitatively evaluate the performance of attention mechanism by visualizing the weight distribution of each instance. We randomly picked several instances from the test set in task A, of which the sentence lengths are more moderate for demonstration. These examples are shown in Figure 5 and categorized into short, long, and noisy sentences for discussion. Darker patch refers to larger weight relatively to other words in the same sentence.

3.7.1 Short Sentences

Figure 5a illustrates two examples whose questions are relatively short in the set. The comments corresponding to these questions are *"the doha international airport..."* and *"...snorkeling two days ago off the coast of dukhan..."*. We can observe that our model successfully learns to focus on the most representative part of the question pertaining to classifying the relationship, which is *"place can ... visited in qatar"* for the first example and *"place for snorkeling"* for the second example.

3.7.2 Long Sentences

In Figure 5b, we investigate two examples with longer questions, which contain 63 words both. Interestingly, the distribution of weights does not become more uniform; on the contrary, our model still focuses on a small number of hot words, for example, *"hectic driving in doha ... car insurance ... quite costly"* and *"puppy dog for ... mall"*. Additionally, some words appear frequently but carry little information for classification are assigned very small weights, such as *I/we/my*, *is/am*, *like*, and *to*.

3.7.3 Noisy Sentence

As we mentioned before, some contents from community question answering are noisy. Figure 5c is an example with excessive usage of question marks. Again, our model exhibits its robust-

[3]The feature-rich based system already combined with IR system)

	Task A		Task B		Task C	
Model	MAP	F1	MAP	F1	MAP	F1
w/ Attention	0.6239	0.6323	0.5723	0.4334	0.2837	0.1449
w/ Attention + aug features	0.6385	0.6218	0.6585	0.5382	0.3236	0.1963

Table 3: Results on augmented feature

	Task A	Task B	Task C
Model	MAP	MAP	MAP
IR	0.538	0.714	0.307
Attention	0.639	0.659	0.324
Feature-Rich & IR	0.632	0.685	0.339
Attention & IR	0.639	0.717	0.394

Table 4: Compared with other systems

ness by allocating very low weights to the noise symbol and therefore excludes the noninformative contents.

4 Conclusion

In this paper, we first demonstrated that a general RNN encoder framework can be applied to community question answering tasks. Next, by adding neural attention mechanism, we showed quantitatively and qualitatively that attention can help improve the RNN encoder framework. Further, to deal with a more realistic scenario, we expanded the framework to incorporate metadata as augmented inputs to the FNN classifier, and pretrained models on larger datasets, which increases both stability and performance. Our model is consistently comparable with traditional feature-rich based system and superior to IR-based system when having enough data.

It is also found that our model is complimentary with IR-based system, which uses a huge amount of external resource but trained for general purposes. By combining the two systems, it approaches the best results comparing to both feature-rich system and IR-based system in all three tasks.

Future work will proceed in two directions: first, we can enrich existing system by incorporating more available metadata and preprocessing data with morphological normalization and OOV mapping; second, we can reinforce our model by carrying out word-by-word and history-aware attention mechanism instead of attending only when reading the last word.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Yonatan Belinkov, Mitra Mohtarami, Scott Cyphers, and James Glass. 2004. Vectorslu: A continuous word vector approach to answer selection in community question answering systems. In *ACM Trans. Inf. Syst*.

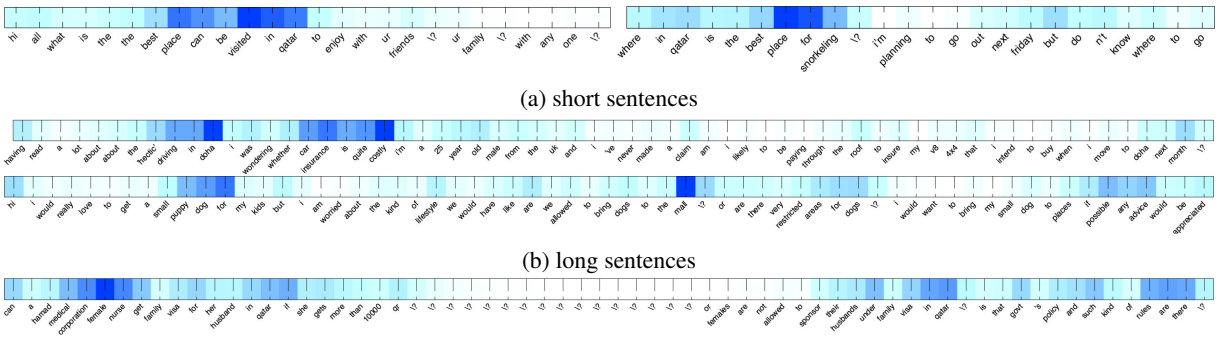
J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

S. Bowman, G. Angeli, C. Potts, and C. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

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*.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber.



(a) short sentences

(b) long sentences

(c) noisy sentences

Figure 5: visualization of attention mechanism on short sentences, long sentences and noisy sentences.

2015. LSTM: A search space odyssey. *CoRR*, abs/1503.04069.

R. Nakov, L. Marquez, W. Magdy, A. Moschitti, and J. Glass. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proc. SamEval*, pages 282–287.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

E. Voorhees and D. Tice. 2000. Building a question answering test collection. In *SIGIR*.

C. Zhai and J. Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. In *ACM Trans. Inf. Syst*.