Research on attention memory networks as a model for learning natural language inference

Zhuang Liu and **Degen Huang** and **Jing Zhang** and **Kaiyu Huang**
School of Computer Science and Technology
Dalian University of Technology, Dalian, P.R. China
huangdg@dlut.edu.cn
{zhuangliu,zhangjingqf}@mail.dlut.edu.cn
huangkaiyucs@foxmail.com

Abstract

Natural Language Inference (NLI) is a fundamentally important task in natural language processing that has many applications. It is concerned with classifying the logical relation between two sentences. In this paper, we propose attention memory networks (AMNs) to recognize entailment and contradiction between two sentences. In our model, an attention memory neural network (AMNN) has a variable sized encoding memory and supports semantic compositionality. AMNN captures sentence level semantics and reasons relation between the sentence pairs; then we use a Sparsemax layer over the output of the generated matching vectors (sentences) for classification. Our experiments on the Stanford Natural Language Inference (SNLI) Corpus show that our model outperforms the state of the art, achieving an accuracy of 87.4% on the test data.

1 Introduction

Natural Language Inference (NLI) refers to the problem of determining entailment and contradiction relationships between two sentences. The challenge in Natural Language Inference, also known as Recognizing Textual Entailment (RTE), is to correctly decide whether a sentence (called a hypothesis) entails or contradicts or is neutral in respect to another sentence (referred to as a premise). Provided with a premise sentence, the task is to judge whether the hypothesis can be inferred (Entailment) or the hypothesis cannot be true (Contradiction) or the truth is unknown (Neutral). Few examples are illustrated in Table 1.

NLI is the core of natural language understanding and has wide applications in NLP, e.g., automatic text summarization (Yan et al., 2011a; RuiYan et al., 2011b); and question answering (Harabagiu and Hickl, 2006). Moreover, NLI is also related to other tasks of sentence pair modeling, including relation recognition of discourse units (YangLiu et al., 2016), paraphrase detection (Hu et al., 2014), etc.

Bowman released the Stanford Natural Language Inference (SNLI) corpus for the purpose of encouraging more learning centered approaches to NLI (Bowman et al., 2015). Published SNLI corpus makes it possible to use deep learning methods to solve NLI problems. So far proposed work based on neural networks for text similarity tasks including NLI have been published in recent years (Hu et al., 2014; Wang and Jiang, 2015; Rocktaschel et al., 2016; Yin et al., 2016);. The core of these models is to build deep sentence encoding models, for example, with convolutional networks (LeCun et al., 1990) or long short-term memory networks (Hochreiter and Schmidhuber, 1997) with the goal of deeper semantic encoders. Recurrent neural networks (RNNs) equipped with internal short memories, such as long short-term memories (LSTMs) have achieved a notable success in sentence encoding. LSTMs are powerful because it learns to control its short term memories. However, the short term memories in LSTMs are a part of the training parameters. This imposes some practical difficulties in training and modeling long sequences with LSTMs.

In this paper, we proposed a deep learning framework for natural language inference, which mainly consists of two layers. As we can see from the fig-

Proceedings of the Workshop on Structured Prediction for NLP, pages 18–24,
Austin, TX, November 5, 2016.

Premise	Hypothesis	Label
A person throwing a yellow ball in the air.	The ball sails through the air.	Entailment
A person throwing a yellow ball in the air.	The person throws a square.	Contradiction
A person throwing a yellow ball in the air.	The ball is heavy.	Neutral

Table 1: Three NLI examples from SNLI. Relations between a Premise and a Hypothesis: Entailment, Contradiction, and Neutral (irrelevant).

ure 1, from top to bottom are: (A) The sentence encoding layer (Figure 1a); (B) The sentence matching layer (Figure 1b). In the sentence encoding layer, we introduce an attention memory neural network (AMNN), which has a variable sized encoding memory and naturally supports semantic compositionality. The encoding memory evolves over time, whose size can be altered depending on the length of input sequences. In the sentence matching layer, we directly model the relation between two sentences to extract relations between premise and hypothesis, and dont generate sentence representations. In addition, we introduce the Sparsemax (Yin and Schutze, 2015) , a new activation function similar to the traditional Softmax, but is able to output sparse probability distributions; then, we present a new smooth and convex loss function, Sparsemax loss function, which is the Sparsemax analogue of the logistic loss. We will explain the two layers in detail in the following subsection.

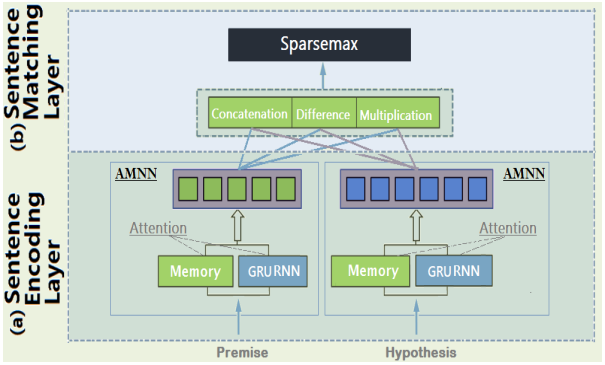

Figure 1: High-level architectures of attention memory neural networks. (a) The sentence encoding layer: Individual sentence modeling via AMNN. (b) The sentence matching layer: Sentence pair modeling, after which a Sparsemax layer is applied for output.

2 Proposed Approach

In our model, we adopt a two-step strategy to classify the relation between two sentences. Concretely, our model comprises two parts:

- The sentence encoding layer (Figure 1a). This part is mainly a sentence semantic encoder, aiming to capture general semantics of sentences.
- The sentence matching layer (Figure 1b). This part mainly introduces how vector representations are combined to capture the relation between the premise and hypothesis for classification.

2.1 The sentence encoding layer: AMNN

In this layer, we introduce an attention memory neural network (AMNN), which implements an attention controller and a variable sized encoding memory, and naturally supports semantic compositionality. AMNN has four main components: Input, Output and Attention memory modules, and an encoding memory. We then examine each module in detail and give intuitions about its formulations.

Suppose we are given an set $\{X^i, Y^i\}_{i=1}^N$, where the input X^i is a sequence $w_1^i, w_2^i, ..., w_{T_i}^i$ of tokens, and Y^i can be an output sequence. The encoding memory $M \in S^{d \times l}$ has a variable number of slots, where d is the embedding dimension and l is the length of the input sequence. Each memory slot vector $m_t \in S^d$ corresponds to the vector representation of w_t. In particular, the memory is initialized with the raw embedding vector at time t=0. As Attention memory module reads more input content in time, the initial memory evolves over time and refines the encoded sequence.

Input module reads an embedding vector. Attention memory module looks for the slots related to the input by computing semantic similarity between each memory slot and the hidden state. We calculate the similarity by the dot product and transform the similarity scores to the fuzzy key vector by normalizing with Softmax function. Since our key vector is

fuzzy, the slot to be composed is retrieved by taking weighted sum of the all slots. In this process, our memory is analogous to the soft attention mechanism. We compose the retrieved slot with the current hidden state and map the resulting vector to the encoder output space. Finally, we write the new representation to the memory location pointed by the key vector.

In our recurrent network, we use a gated recurrent network (Cho et al., 2014a; Chung et al., 2014). We also explored the more complex LSTM (Hochreiter and Schmidhuber, 1997) but it performed similarly and is more computationally expensive. Both work much better than the standard *tanh* RNN and we postulate that the main strength comes from having gates that allow the model to suffer less from the vanishing gradient problem (Hochreiter and Schmidhuber, 1997).

Concretely, let $v_l \in R^l$ and $v_d \in R^d$ be vectors, and given a input function f_{input}^{GRU}, a output function f_{output}^{GRU}, and the key vector output a_t, the output state h_t and the encoding memory M_t in time step t as

$$o_t = f_{input}^{\mathbf{GRU}}(x_t) \tag{1}$$

$$a_t = Softmax(o_t^T M_{t-1}) \tag{2}$$

$$m_t = a_t^T M_{t-1} \tag{3}$$

$$h_t = f_{output}^{\mathbf{GRU}}(o_t, m_t) \tag{4}$$

$$\begin{aligned} M_t = M_{t-1}(1 - (a_t \otimes v_d)^T) \\ +(h_t \otimes v_l)(a_t \otimes v_d)^T \end{aligned} \tag{5}$$

where the input function f_{input}^{GRU} and the output function f_{output}^{GRU} are neural networks, also are the training parameters in the model. We abbreviate the above computation with $M_t = GRU(x_t, M_{t-1})$. Equation (1) is a matrix of ones, $\otimes$ denotes the outer product which duplicates its left vector *l* or *d* times to form a matrix. The function f_{input}^{GRU} sequentially maps the word embeddings to the internal space of the memory w_{t-1}. Then Equation (2),(3),(4),and (5) retrieves a memory slot m_t that is semantically associated with the current input word w_t, and combines the slot m_t with the input w_t, and then transforms the composition vector to the encoding memory and rewrites the resulting new representation into the slot location of the memory space. The slot location (ranging from 1 to *d*) is defined by a key vector a_t which the Input module emits by attending over the memory slots. In GRU (x_t, M_{t-1}), the slot that was retrieved is erased and then the new representation is located. Attention memory module performs this iterative process until all words in the input sequence is read, and performs the input and output operations in every time step. The encoding memories $\{M\}_{t=1}^T$ and output states $\{h\}_{t=1}^T$ are further used for the tasks.

2.2 The sentence matching layer

Combining sentences encoding: In this part, we introduce how vector representations of individual sentences are combined to capture the relation between the premise and hypothesis. Three matching methods were applied to extract relations.

- Concatenation of the two representations
- Element-wise product
- Element-wise difference

This matching architecture was first used by (Mou et al., 2015) The first matching method follows the most standard procedure of the Siamese architectures, while the latter two are certain measures of similarity or closeness. This matching process is further concatenated (Figure 1b), given by

$$V_c = [(V_p V_h; V_p - V_h; V_p \odot V_h] \tag{6}$$

where V_p and V_h are the sentence vectors of the premise and hypothesis, respectively; $\odot$ denotes element-wise product; semicolons refer to column vector concatenation. V_c is the generated matching vector of the matching layer.

We would like to point out that, with subsequent linear transformation, element-wise difference is a special case of concatenation. If we assume the subsequent transformation takes the form of $W[V_p V_h]^T$, where W=$[W_1 W_2]$ is the weights for concatenated sentence representations, then element-wise difference can be viewed as such that $W_0(V_p - V_h)$ W_0

is the weights corresponding to element-wise difference). Thus, our third heuristic can be absorbed into the first one in terms of model capacity. However, as will be shown in the experiment, explicitly specifying this heuristic significantly improves the performance, indicating that optimization differs, despite the same model capacity. Moreover, word embedding studies show that linear offset of vectors can capture relationships between two words (Mikolov et al., 2013b), but it has not been exploited in sentence-pair relation recognition. Although element-wise distance is used to detect paraphrase in (He et al., 2015), it mainly reflects similarity information. Our study verifies that vector offset is useful in capturing generic sentence relationships, akin to the word analogy task.

Sparsemax Transformation: In this part, we introduce the Sparsemax transformation, which has similar properties to the traditional Softmax, but is able to output sparse probability distributions. This transformation was first used by Andre (Martins and Astudillo, 2016). Let $\triangle^{K-1} := p \in R^K | \mathbf{1}^T p = 1, p \geq 0$ be the (K-1)-dimensional simplex. We are interested in functions that map vectors in R^K to probability distributions in $\triangle^{K-1}$. Such functions are useful for converting a vector of real weights (e.g., label scores) to a probability distribution (e.g. posterior probabilities of labels). The Sparsemax function, defined componentwise as:

$$\textbf{Sparsemax}(z) := \underset{p \in \triangle^{K-1}}{\operatorname{argmax}} \|p - z\|^2 \qquad (7)$$

Sparsemax has the distinctive feature that it can return sparse posterior distributions, that is, it may assign exactly zero probability to some of its output variables. This property makes it appealing to be used as a filter for large output spaces, to predict multiple labels, or as a component to identify which of a group of variables are potentially relevant for a decision, making the model more interpretable. Crucially, this is done while preserving most of the attractive properties of Softmax: we show that Sparsemax is also simple to evaluate, it is even cheaper to differentiate, and that it can be turned into a convex loss function.

We present the Sparsemax loss, a new loss function that is the Sparsemax analogue of logistic regression. We show that it is convex, everywhere differentiable, and can be regarded as a multi-class generalization of the Huber classification loss, an important tool in robust statistics (Zhang and Tong, 2004). We apply the Sparsemax loss to train multi-label linear classifiers. Finally, we use a Sparsemax layer over the output of a non-linear projection of the generated matching vector for classification.

3 Experiments

3.1 Dataset

To evaluate the performance of our model, we conducted our experiments on Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015). The dataset, which consists of 549,367/9,842/9,824 premise-hypothesis pairs for train/dev/test sets and target label indicating their relation. Each pair consists of a premise and a hypothesis, manually labeled with one the labels ENTAILMENT, CONTRADICTION, or NEUTRAL. We used the provided training, development, and test splits.

3.2 Hyper-Parameter Settings

In this section, we provide details about training the neural network. The model is implemented using open-source framework the TensorFlow (Abadi et al., 2015). The training objective of our model is cross-entropy loss, and we use mini-batch stochastic gradient descent (SGD) with the Rmsprop (Hinton, 2012) for optimization. We set the batch size to 128, the initial learning rate to 3e-4 and l_2 regularizer strength to 3e-5, and train each model for 60 epochs, and fix dropout rate at 0.3 for all dropout layers. In our neural layers, we used pretrained 300D Glove 840B vectors (Pennington et al., 2014) to initialize the word embedding. Out-of-vocabulary words in the training set are randomly initialized by sampling values uniformly from (0.02, 0.02). All of these embedding are not updated during training. Each hyper-parameter setting was run on a single machine with 10 asynchronous gradient-update threads, using Adagrad (Duchi et al., 2011) for optimization.

3.3 Results and Qualitative Analysis

Table 1 compares the results of our models with the previous art-of-the-state baseline results. We compare our models against several baselines, including

Method	Train acc. (%)	Test acc. (%)	Params.
Previous non-neural network results			
Lexicalized Classifier(Bowman et al., 2015)	99.7	78.2	-
Previous neural network results			
LSTM LSTM RNN encoders(Bowman et al., 2016)	83.9	80.6	3.0M
Tree-based CNN encoders (Mou et al., 2015)	83.3	82.1	3.5M
SPINN-NP encoders (Bowman et al., 2016)	89.2	83.2	3.7M
LSTM with attention (Rocktaschel et al., 2016)	85.3	83.5	252K
mLSTM (Wang and Jiang, 2015)	92.0	86.1	1.9M
LSTMN with deep attention fusion (Cheng et al., 2016)	88.5	86.3	3.4M
Decomposable attention model (Parikh et al., 2016)	90.5	86.8	582K
Our results			
AMNs-G (AMNN with GRU)	89.1	87.4	3.5M
AMNs-L (AMNN with LSTM)	89.3	87.0	3.2M

Table 2: Train/test accuracies on the SNLI dataset and the approximate number of trained parameters (excluding embeddings) for each approach. "-G" and "-L" denote GRU and LSTM, resp.

the strongest published non-neural network-based result from Bowman et al. (2015) and previous neural network models built around several types of sentence encoders. Here AMNs-G, AMNs-L denote the neural networks of AMNN *GRU RNN* and *LSTM RNN*, respectively. When we experimented with the AMNN model instead of some previous models, Tree-based CNN by (Mou et al., 2015), SPINN-NP by Bowman et al. 2016, LSTM with attention by Rocktaschel et al. 2016, as initial sentence representations of the premise and the hypothesis. As seen, both AMNs-G and AMNs-L further slightly improved the result. Our models are able to outperform the previous state-of-the-art in terms of the accuracy at test time, by approximately 0.6%.

4 Related Work

Language inference or entailment recognition can be viewed as a task of sentence pair modeling. Most neural networks in this field involve a sentence-level model, followed by one or a few matching modules. Our method is motivated by the central role played by sentence-level modeling (Yin and Schutze, 2015; Mou et al., 2016; Wan et al., 2015; Parikh et al., 2016; YangLiu et al., 2016; Rocktaschel et al., 2016) and previous approaches to semantic encoder (Graves et al., 2014; Weston et al., 2015; Sukhbaatar et al., 2015; Kumar et al., 2016; Bahdanau et al., 2015). (Yin and Schutze, 2015) and (Mou et al., 2016) apply convolutional neural networks (CNNs) as the individual sentence model, where a set of feature detectors over successive words are designed to extract local features. (Wan et al., 2015) and (YangLiu et al., 2016) build sentence pair models upon recurrent neural networks (RNNs) to iteratively integrate information along a sentence.

The neural counterpart to sentence similarity modeling, attention and external memory, which are the key part of our approach, was originally proposed and has been predominantly used to attempt to extend deep neural networks with an external memory (NTM) (Graves et al., 2014). NTM implements a centralized controller and a fixed-sized random access memory. The controller uses attention mechanisms to access the memory. The work of (Sukhbaatar et al., 2015) combines the soft attention with Memory Networks (MemNNs) (Graves et al., 2014). Although MemNNs are designed with non-writable memories, it constructed layered memory representations and showed promising results on both artificial and real question answering tasks. Another variation of MemNNs is Dynamic Memory Network (DMN) (Kumar et al., 2016) which is equipped with an episodic memory and seems to be flexible in different settings.

In contrast, our use of external memory is based on variable sized semantic encoder and our method use the attention mechanism to access the external memory. The size of the memory can be altered depending on the input length, i.e., we use a larger memory for long sequences and a smaller memory for short sequences. Our models are suitable for NLI and can be trained easily by any gradient descent optimizer.

5 Conclusion and future work

In this paper, we proposed attention memory networks (AMNs) to solve the natural language inference (NLI) problem. Firstly, we present the attention memory neural network (AMNN) that uses attention mechanism and has a variable sized semantic memory. AMNN captures sentence-level semantics; then we directly model the relation with combining two sentence vectors to aggregate information between premise and hypothesis. Finally, we introduce the Sparsemax, a new activation function similar to the traditional Softmax, but is able to output sparse probability distributions. We use the Sparsemax layer over the generated matching vector for output. The attention memory networks (AMNs) over the premise provides further improvements to the predictive abilities of the system, resulting in a new state-of-the-art accuracy for natural language inference on the Stanford Natural Language Inference corpus.

Our model can be easily adapted to other sentence-matching models. There are several directions for our future work: (1) Employ this architecture on other sentence matching tasks such as Text Summarization, Paraphrase Text Similarity and Question Answer etc. (2) Try more heuristics matching methods to make full use of the individual sentence vectors. (3) Extend AMNN to produce encoding memory and representation vector of entire documents.

Acknowledgments

We thank Kai-Wei Chang, Ming-Wei Chang, Alexander Rush, Vivek Srikumar, and the anonymous EMNLP reviewers for their helpful comments on drafts of this paper. This research is supported by National Natural Science Foundation of China(No.61672127No. 61173100).

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, O. Vinyals F. Viegas, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow.org*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In ICLR*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, , and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *In Proceedings of EMNLP*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *In Proceedings of ACL*.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, , and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR, abs/1409.1259*.

Junyoung Chung, Calar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Technical Report Arxiv report 1412.3555*.

John Duchi, Elad Hazan, , and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, (12):2121–2159.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. *In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 905–912.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 17–21.

G. Hinton. 2012. Lecture 6.5: rmsprop: divide the gradient by a running average of its recent magnitude. *coursera: Neural networks for machine learning*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. *In Advances in Neural Information Processing Systems*, pages 2042–2050.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. *CoRR, abs/1506*, 2016.

Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D.Jackel. 1990. Handwritten digit recognition with a back-propagation network. *In Advances in NIPS*, 1990.

Andre F. T. Martins and Ramon F. Astudillo. 2016. From softmax to sparsemax:a sparse model of attention and multi-label classification. *arXiv preprint arXiv:1602.02068v2*.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. *In NAACL-HLT*, pages 746–751.

Lili Mou, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. *In Proceedings of ACL (short papers)*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Recognizing entailment and contradiction by tree-based convolution. *In ACL 2016*.

Ankur P Parikh, Oscar Tackstrom, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Tim Rocktaschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. *In Proceedings of ICLR*.

RuiYan, XiaojunWan, JahnaOtterbacher, LiangKong, Xiaoming Li, , and Yan Zhang. 2011b. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. *In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 745–754.

Sainbayar Sukhbaatar, Jason Weston, and Rob Fergus et al. 2015. End-to-end memory networks. *In NIPS 2015*, pages 2431–2439.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2015. A deep architecture for semantic matching with multiple positional sentence representations. *arXiv preprint arXiv:1511.08277*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *In Proceedings of NAACL*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *In ICML 2015*.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011a. Timeline generation through evolutionary trans-temporal summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 433–443.

YangLiu, SujianLi, XiaodongZhang, and ZhifangSui. 2016. Implicit discourse relation classification via multi-task neural networks. *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Wenpeng Yin and Hinrich Schutze. 2015. Convolutional neural network for paraphrase identification. *In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *In Transactions of the Association of Computational Linguistics*.

Zhang and Tong. 2004. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, pages 56–85.