Recognizing Textual Entailment using Deep Learning Techniques

Han Yang

Supervisor: Marta R. Costa-jussà & Lluís Padró Cirera

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONATECH

July 6th, 2017

Outline

- Introduction
- Background
- Model Development
- Result and Analysis
- Conclusion

Introduction

Introduction - Textual Entailment

Natural language inference (NLI) or Textual Entailment (TE) in natural language processing refers to the problem of determining a directional relation between two text fragments.

Premise	**Hypothesis**	**Relation**
A soccer game with multiple males playing.	Some men are playing a sport	Entailment
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	Contradiction
A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	Neutral

Table 1: Example of NLI/TE task

Introduction - RepEval2017 shared task

Corpus	Size	Contributor	Time
Stanford Natural Language Inference (SNLI)	570K	Bowman et al.	2015
Multi-Genre NLI (MNLI)	433K	Adina Williams et al.	2017

Table 2: Corpus for NLI task

RepEval2017 (co-located with EMNLP 2017) shared task meant to evaluate NLI models based on sentence encoders that transform sentences into fixed-length vector representations and reason using those representations.

The main task in this thesis project will be a three-class classification problem over sentence pairs upon MNLI corpus.

Introduction - Contributions

In this thesis project, we use the model introduced by Adina Williams et al. [1] as the baseline model for the NLI task. The baseline model is consisted with a word embedding layer and a BiLSTM encoder.

We augment the baseline BiLSTM model and propose our **Character-level Intra Attention Network (CIAN)**. In our CIAN model, we use the character-level convolutional network to replace the standard word-level embedding layer, and we use the intra attention layer to capture the intra-sentence semantics.

Background

Background - Neural Networks

Simple feed-forward network

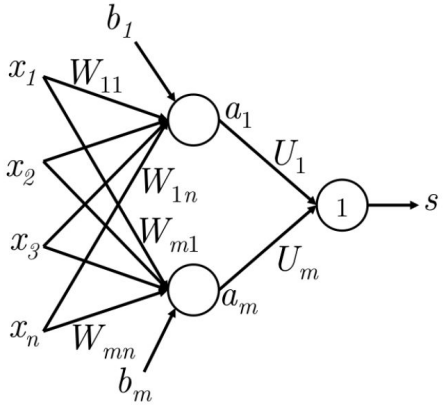


Figure 1: A simple feed-forward network [2]

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix} = \sigma(z) = \sigma(Wx + b)$$

Training with backpropagation

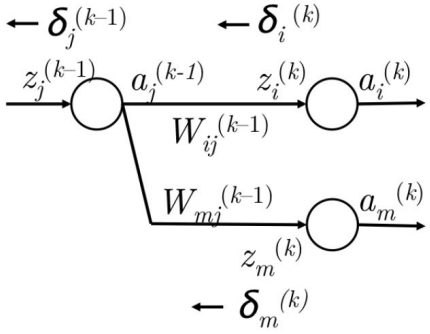


Figure 2: Step of backpropagation [2]

$$\delta_j^{(k-1)} = f'(z_j^{(k-1)}) \Sigma_i \delta_i^{(k)} W_{ij}^{(k-1)}$$

Background - Sequence encoder

The aim of an encoder is to learn a representation (encoding) from a set of text sequences. Those encoding could then be used for NLP tasks (NLI here).

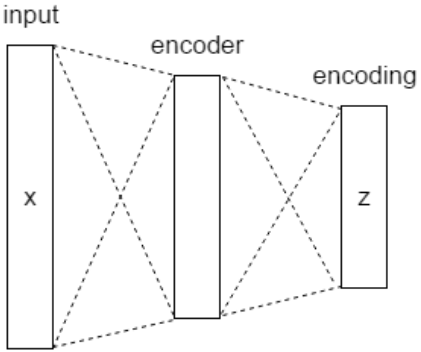


Figure 3: Architecture of sequence encoder

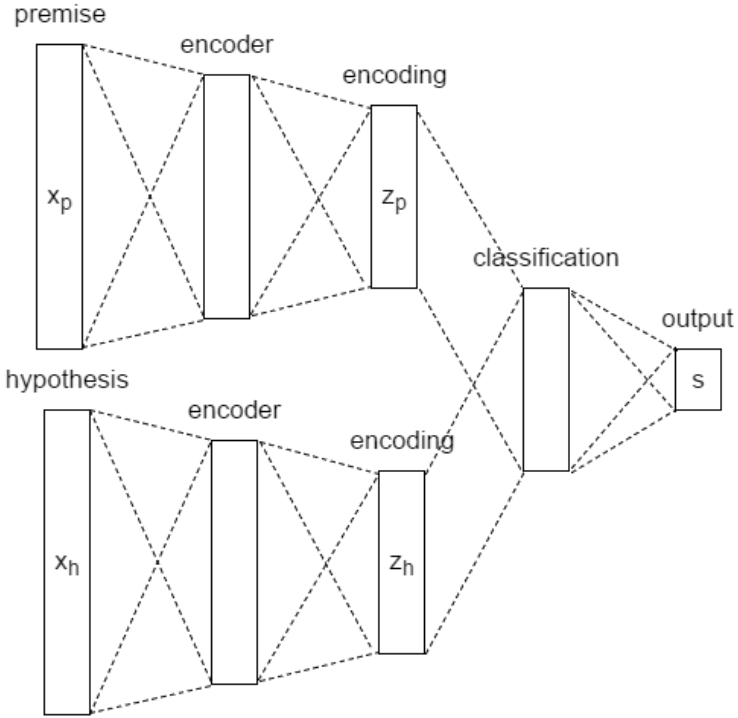
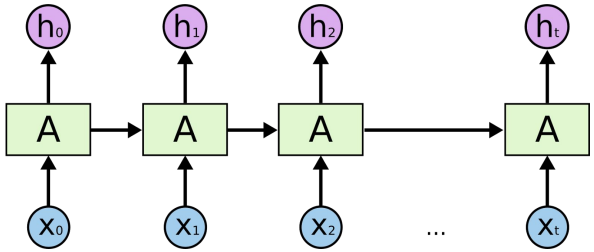


Figure 4: Architecture of sequence encoder for NLI task

Background - Sequence encoder (RNN)

Recurrent neural networks (RNN) recursively concatenate each word with its previous memory, until the whole information of a sentence has been derived.

Elman network, a simple recurrent network:



$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

Figure 5: Architecture of a recurrent network [3]

Background - Sequence encoder (RNN)

Long short-term memory (LSTM) is proposed in 1997 by Hochreiter et al. [4], which solves the problem of RNN:

- Gradient vanishing
- Long-term dependencies

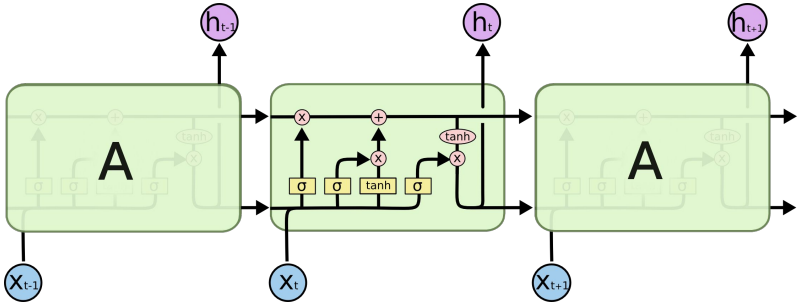


Figure 6: Architecture of a LSTM network [3]

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \quad \text{(Input gate)}$$

$$f_t = s(W^{(f)}x_t + U^{(f)}h_{t-1}) \quad \text{(Forget gate)}$$

$$o_t = s(W^{(o)}x_t + U^{(o)}h_{t-1}) \quad \text{(Output gate)}$$

$$\hat{c}_t = tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \quad \text{(New memory cell)}$$

$$c_t = ft \circ c_{t-1} + i_t \circ \hat{c}_t \quad \text{(Final memory cell)}$$

$$h_t = o_t \circ tanh(c_t) \quad \text{(Hidden state)}$$

Background - Sequence encoder (CNN)

Convolutional neural networks (CNN) concatenate the sentence information by applying multiple convolving filters over the sentence.

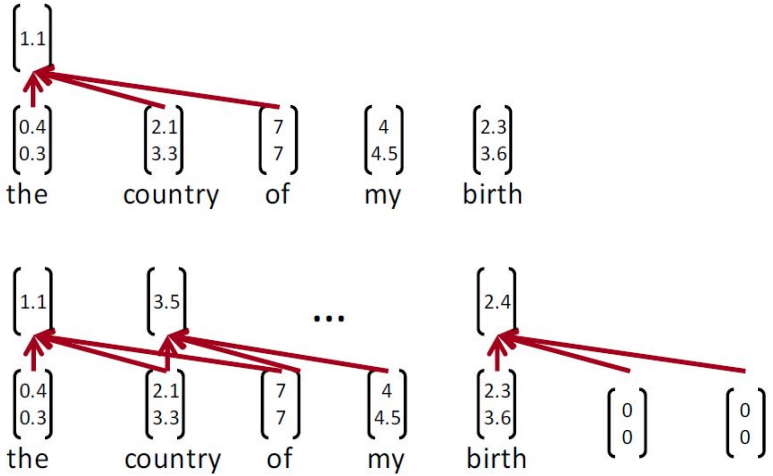


Figure 7: Convolutional computation over a sentence [2]

x: input vector; w: filter matrix

h: widtn of filter (3 here)

$$c_{[i]} = f\,(w^T x_{i:i+h-1} + b)$$

$$c = [c_{[1]}, c_{[2]}, \ldots, c_{[n-h+1]}]$$

Background - Sequence encoder (CNN)

- Multiple-Filters: use multiple filters with different width to recognize a different kind of word combination.
- Maxpooling: keep the information of the most important features

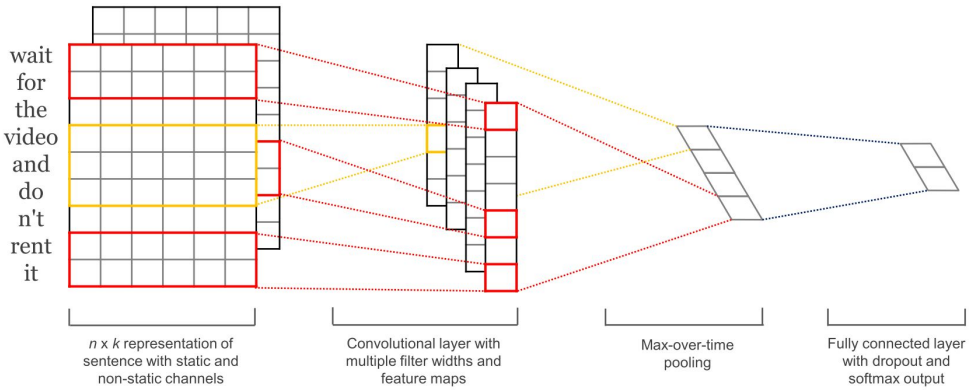


Figure 8: Convolutional computation over a sentence with multiple-filters and maxpooling [5]

Model Development

Model Development - Baseline

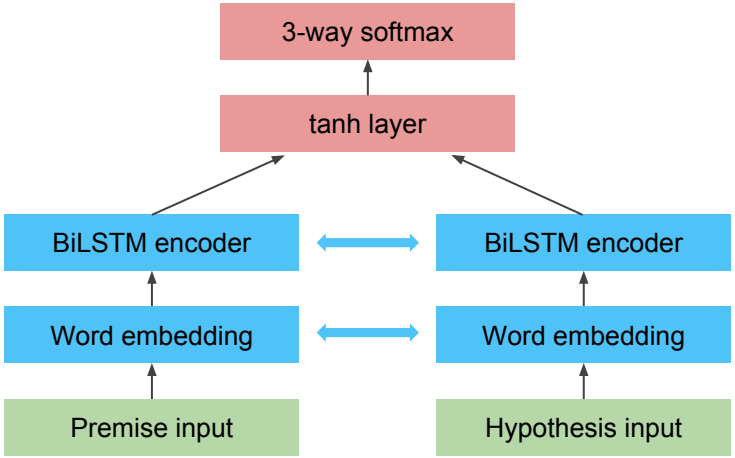


Figure 9: Architecture of baseline model

The BiLSTM model from Adina Williams et al. [1] is used as the baseline model:

- word embeddings are initialized with the 300D pretrained GloVe vectors (840B token version, 2.2M vocab, Pennington et al. 2014) [6]
- use BiLSTM as the sentence encoder, the sentence representation vector is computed with an average pooling over all BiLSTM hidden states

Model Development - CharCNN

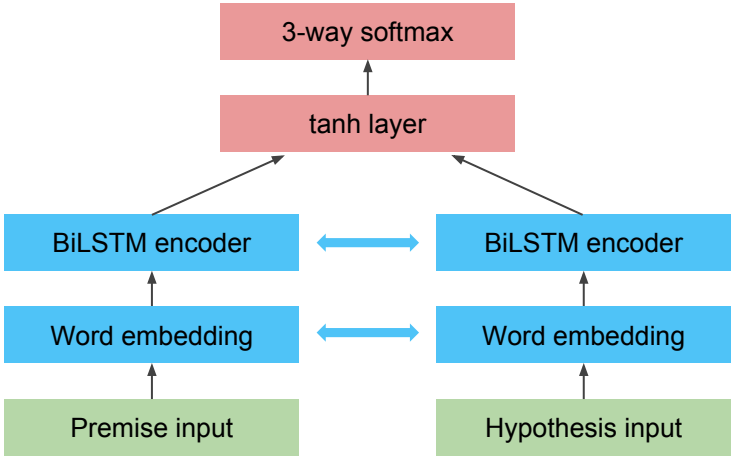


Figure 9: Architecture of baseline model

In the baseline BiLSTM model, the input x_t to the encoder layer at time t are pre-trained word embeddings. Those pre-trained word embeddings can boost the performance of the model. However, it is limited to the finite-size of the pre-trained embedding vocabulary.

Here we take place of the word embedding layer with a character-level convolutional network (CharCNN) introduced by Kim et al. [7] in 2016.

Model Development - CharCNN

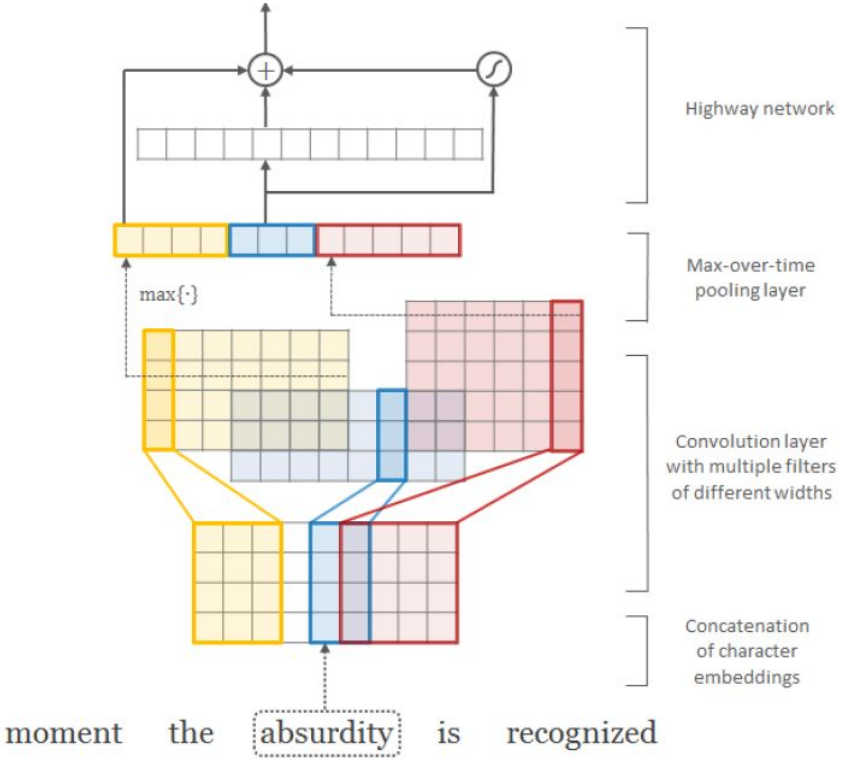


Figure 10: Architecture of CharCNN [7]

Convolution layer:

- input is character embedding
- 7 filters with width from 1 to 7
- each filter followed with a max pooling

Highway network:

- $z = t \odot g(W_H y + b_H) + (1 - t) \odot y$
- adaptively carrying some dimensions of the input directly to the output

Model Development - CharCNN

Advantage:

- Model is able to exploit rich semantic and orthographic features of words, which is computed from characters of corresponding word.

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
LSTM-Word	*although*	*your*	*conservatives*	*jonathan*	*advertised*	–	–	–
	letting	*her*	*we*	*robert*	*advertising*	–	–	–
	though	*my*	*guys*	*neil*	*turnover*	–	–	–
	minute	*their*	*i*	*nancy*	*turnover*	–	–	–
LSTM-Char (before highway)	*chile*	*this*	*your*	*hard*	*heading*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

Table 3: Nearest neighbors of OOV words [7]

Model Development - Intra sentence attention

In the baseline BiLSTM model, the sentence representation s as the output of BiLSTM is an average pooling over the BiLSTM hidden states [h_0, h_1, … , h_n]. However, this has its bottleneck as we intuitively know that not all words contribute equally to the sentence representation.

To augment the performance of RNN based encoder, the concept of Intra Attention mechanism introduced by Z. Yang et al. [8] in 2016 is implemented here.

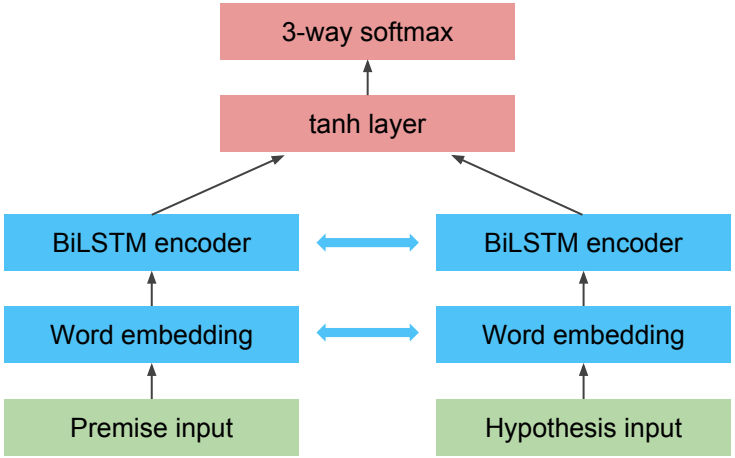


Figure 9: Architecture of baseline model

Model Development - Intra sentence attention

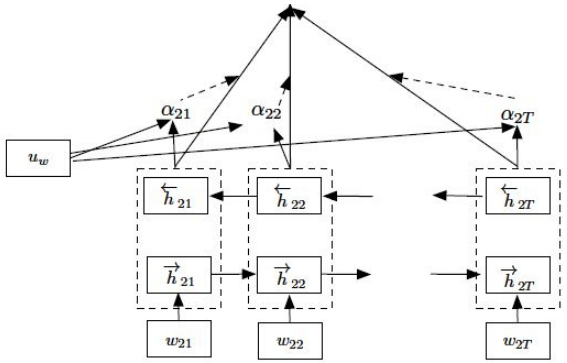


Figure 11: Architecture of intra attention [8]

$$u_t = \tanh(W_\omega h_t + b_\omega)$$

$$\alpha_t = \frac{\exp(u_t^T u_\omega)}{\sum_t \exp(u_t^T u_\omega)}$$

$$s = \sum_t \alpha_t h_t$$

1. Feed the hidden states h_t through a one-layer MLP to get u_t as the hidden representation.
2. Use a softmax function to catch the normalized importance weight matrix α_t.
3. Sentence vector s is computed by a weighted sum of all hidden states h_t with the weight α_t.

Advantages:

- the ability to efficiently encode long sentences
- enhance the interpretability of the model by visualizing the attention weights α_t

Model Development - CIAN

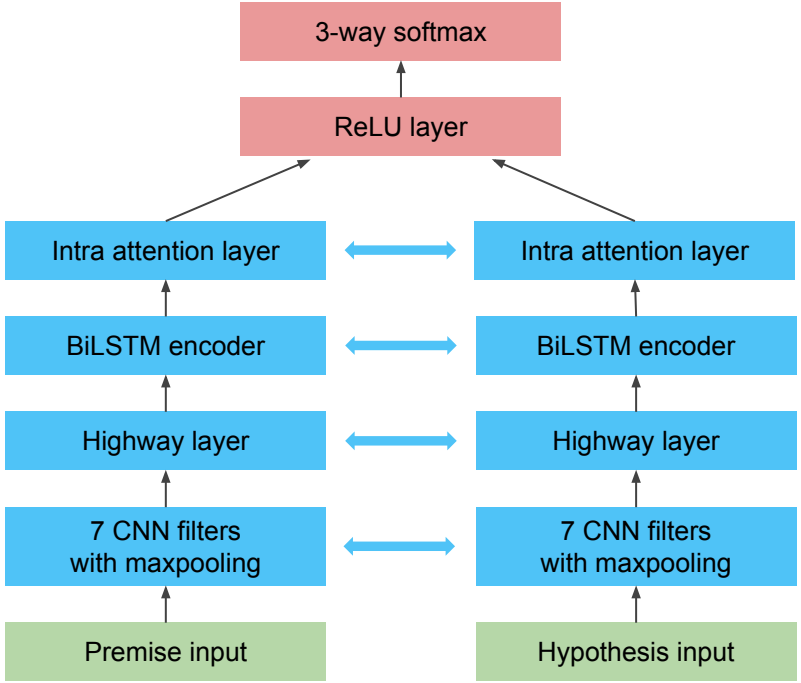


Figure 12: Architecture of CIAN model

Layer	Output Shape	Number of Parameters
Input layer	(50, 15)	0
CNN layer with Maxpooling	(50, 1100)	83035
Highway layer	(50, 1100)	4844400
BiLSTM layer	(50, 600)	3362400
Intra Attention layer	(600)	186000
ReLU layer	(300)	1449000
3-way Softmax layer	(3)	1803
Total parameters: 9,931,038		

Table 4: Detail information for CIAN model

Result and Analysis

Result and Analysis - Test Accuracy

Model	Matched	Mismatched
BiLSTM	67.0	67.6
CIAN	67.9	68.2

Table 5: Test accuracy

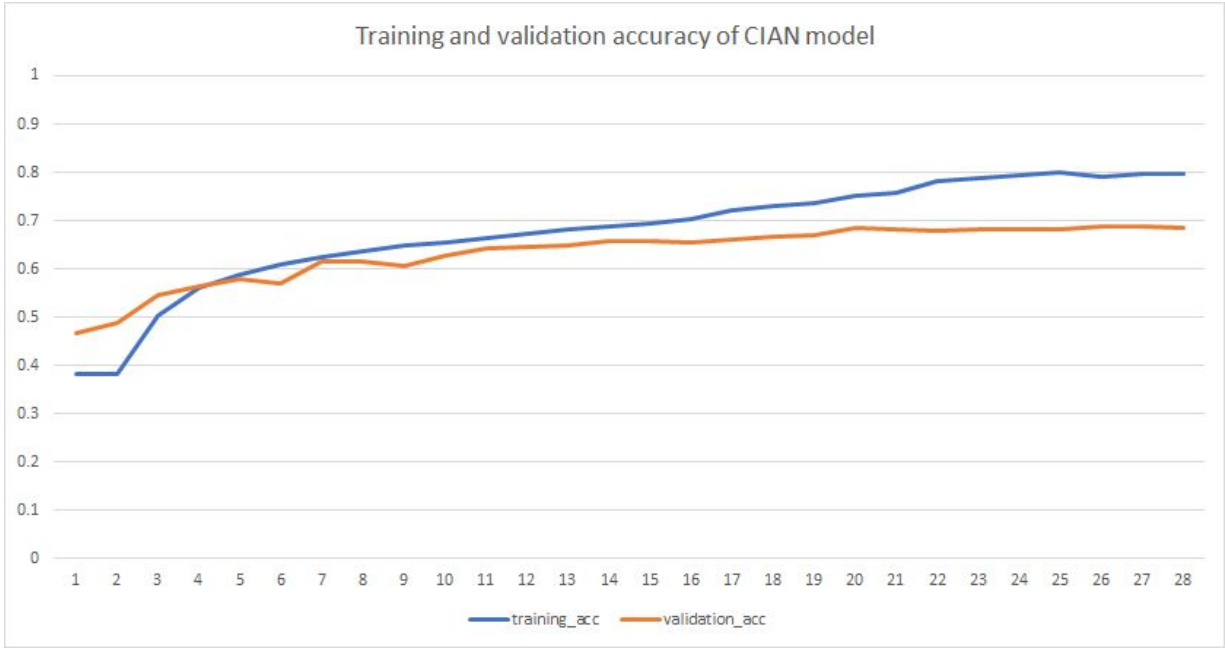


Figure 13: Training and validation accuracy

Result and Analysis - Error Analysis

	Matched		Mismatched	
Tag	**BiLSTM**	**CIAN**	**BiLSTM**	**CIAN**
CONDITIONAL	100	48	100	62
WORD_OVERLAP	**50**	**79**	**57**	**62**
NEGATION	71	71	69	70
ANTO	**67**	**82**	**58**	**70**
LONG_SENTENCE	**50**	**68**	**55**	**63**
TENCE_DIFFERNCE	64	65	71	72
ACTIVE/PASSIVE	**75**	**87**	**82**	**90**
PARAPHRASE	78	88	81	89
QUANTITY/TIME_REASONING	50	47	46	44
COREF	84	67	80	72
QUANTIFIER	64	63	70	69
MODAL	66	66	64	70
BELIEF	74	71	73	70

Table 6: Error analysis

Result and Analysis - Visualization

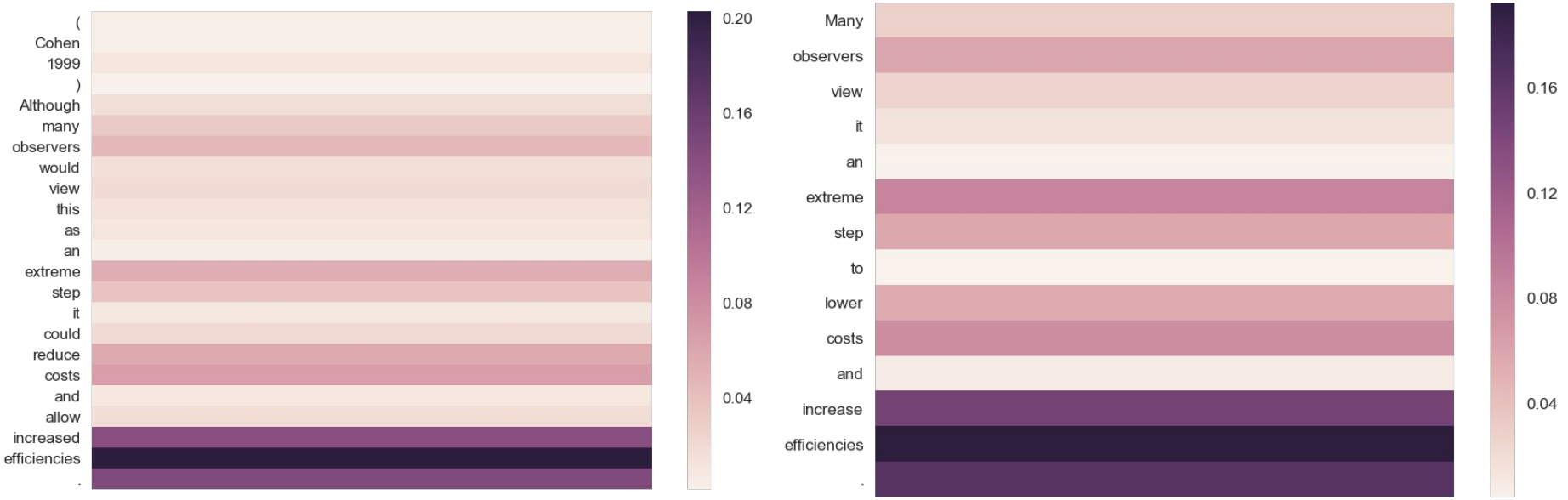


Figure 14: PairID 192997e, label Entailment

Result and Analysis - Visualization

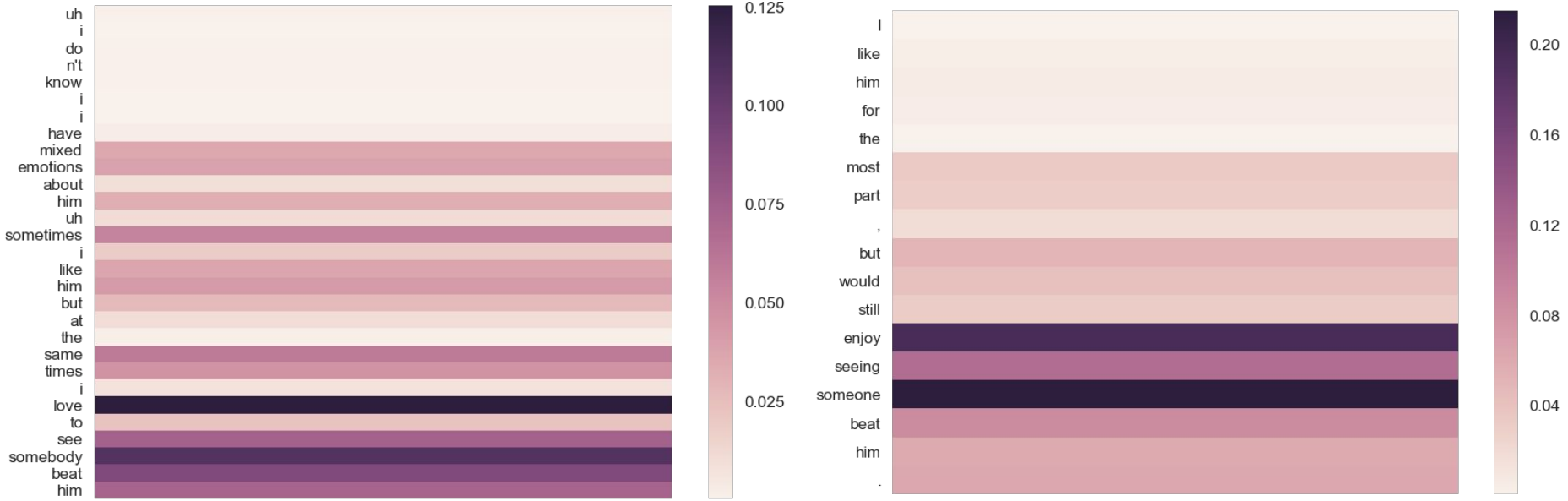


Figure 15: PairID 254941e, label Entailment

Conclusion

Conclusion

In this thesis project, we use the BiLSTM baseline model to solve the NLI task upon MNLI corpus. The main contributions are the following augments on the baseline model:

- use the character-level convolutional network to replace the standard word-level embedding layer, which captures rich semantic and orthographic features of words.
- use the intra attention layer to capture the intra-sentence semantics, which also provides high interpretability of the model.

As it's an end-to-end neural network that does not need any specific pre-processing or outside data like pre-trained word embeddings. It can be easily applied to other encoder architecture tasks such as language modeling, sentiment analysis and question answering.

Reference

[1] Williams, A., Nangia, N., & Bowman, S. R. (2017). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. arXiv preprint arXiv:1704.05426.

[2] Richard Socher et al., “Natural Language Processing with Deep Learning”, Stanford University (2017).

[3] Olah, Christopher. "Understanding lstm networks." GITHUB blog, posted on August 27 (2015): 2015.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[5] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[6] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In EMNLP (Vol. 14, pp. 1532-1543).

[7] Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016, March). Character-aware neural language models. In Thirtieth AAAI Conference on Artificial Intelligence.

[8] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In Proceedings of NAACL-HLT (pp. 1480-1489).

Thank you for listening!