ESIM Implementation with FastNLP on Stanford NLI, Mutli NLI, and Chinese NLI

Linyang He
School of Data Science
Fudan University
lyhe15@fudan.edu.cn

Abstract

Understanding entailment and contradiction is fundamental to understanding natural language, and inference about entailment and contradiction is a valuable testing ground for the development of semantic representations, therefore, Natural Language Inference(NLI) is quite a challenging task in NLP field. With the large NLI corpus provided(such as SNLI, MultiNLI, XNLI, etc), it is possible to train an effective neural network based inference models nowadays. In this project, we choose the Enhanced LSTM for Natural Language Inference(ESIM) as the base model. ESIM model is a carefully designed sequential inference model based on chain LSTMs. Besides, we use the FastNLP to help implement. FastNLP is a modular Natural Language Processing system based on PyTorch. At last, we test our improved ESIM model on Stanford Natural Language Inference corpus, Multi-Genre Natural Language Inference corpus and Chinese Natural Language Inference corpus. The best accuracy was achieved at 68.8% for MultiNLI. While for the Chinese NLI, our test score(95.21%) is way much higher than the current top performing model(82.38%) in the online CNLI2018 competition, which might be a new state-of-the-art.

1 Introduction

Natural language inference (NLI) and paraphrase detection are one of the main research topics in natural language processing. NLI has been addressed using a variety of techniques, including those based on symbolic logic, knowledge bases, and neural networks. I Natural language inference refers to a problem of determining entailment and contradiction between two statements and paraphrase detection focuses on determining sentence duplicity.

Specifically, natural language inference is concerned with determining whether a natural language hypothesis h can be inferred from a premise p. There are three relationships between the premise p and the hypothesis h: *Entailment*, *Neutral*, and *Contradiction*.$Entailment$ suggests that$h$ entails p, while *Contradiction* shows that p is contrary to h, and *Neutral* is in the between. Here're three examples.

1. p: *A man inspects the uniform of a figure in some East Asian country.*
   h: *The man is sleeping.*
   Relationship: *Contradiction*
2. p: *A soccer game with multiple males playing.*
   h: *Some men are playing a sport.*
   Relationship: *Entailment*
3. p: *A smiling costumed woman is holding an umbrella.*

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

h: *A happy woman in a fairy costume holds an umbrella.*
Relationship: *Neutral*

Recent years have seen advances in modeling natural language inference. Important contributions on the creation of the richness of the annotated dataset for NLI include the Stanford Natural Language Inference(SNLI) dataset [4](Bowman et al., 2015), the Multi-Genre NLI(MultiNLI) corpus(Williams et al., 2018), and the Cross-Lingual NLI (XNLI) corpus (Conneau et al., 2018). In this project, we use Stanford NLI, which is also widely used in previous work. The corpus has 570,000 human-written English sentence pairs manually labeled by multiple human subjects. This makes it feasible to train more complex inference models. Besides, we want to see if the very model works well in the English corpus would work well in Chinese. Therefore, other than SNLI, we use a Chinese Natural Language Inference corpus provided by Beijing Language and Culture University as the dataset.

Neural network models, which often need relatively large annotated data to estimate their parameters, have shown to achieve the state of the art on SNLI. As for the contributions on the neural models, papers published on this topic use recurrent neural networks (Wang & Jiang, 2016, Wang et al., 2017), convolutional neural networks (Mou et al., 2016) or feed-forward neural networks with soft-alignment and attention (Parikh et al., 2016.).

The ESIM model we choose in this project is a hybrid network model with RNN and soft-alignment(or self attention). It is provided by Chen et al., 2016[1]. While some previous well performed models use quite complex neural network structures to achieve their best results, ESIM, instead, provides a comparatively less complicated yet excellently-performing model, which suggests that the potentials of such sequential inference approaches have not been fully exploited yet. In this project, we modify the number of the MLP layer from one three. This help us achieve quite high test accuracy.

The remaining part of this report is organized as follows: Section 2 will talk about other people's work brifely. Section 3 will focus on some background knowledge needed for this project, including how to use FastNLP, what is BiLSTM, how self-attention mechanism works and so on. Section 4 focuses on the ESIM model. Section 5 describes the experimental details, including word embedding dataset(GloVe[3], SGNS), data pre-processing and processing, experimental parameter settings, training and test details. Section 6 will be the experimental results and analysis. The final part of this article is some ideas for future work.

2 Related Work

Generally speaking, for NLI, the idea allows neural models to pay attention to specific areas of the sentences. Much more advanced networks have been developed since Bowman et al provided the SNLI dataset. Among them, more relevant to ESIM are the approaches proposed by Parikh et al. (2016) and Munkhdalai and Yu (2016b), which are among the best performing-models.

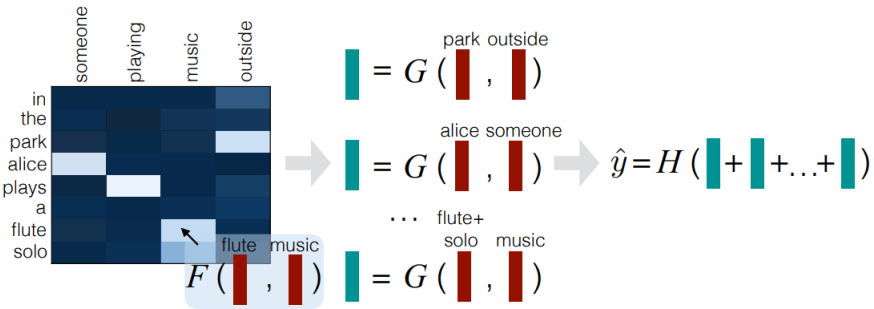


Figure 1: A Decomposable Attention Model for Natural Language Inference, Parikh et al. (2016)

As the Figure 1 shows, Parikh et al. (2016) propose a relatively simple but very effective decomposable model. The model decomposes the NLI problem into subproblems that can be solved separately. The figure below depicts the core model architecture which is composed of three layers: attention, comparison and aggregation.

Attention layer is implemented using a feed-forward neural network F that is applied to both questions separately. Outputs of the neural network are then normalized using a softmax function and soft-aligned to the second sentence. In the comparison level, what the neural network does is to compare soft-aligned sentence matrices. Similarly to the previous step, a feed-forward neural network is used and inputs to the network are concatenated sentence matrices respectively. The last part of the core model architecture is the aggregation layer. All this layer does is a column-wise sum over the output of the comparison network so that they could obtain a fixed-size representation of every sentence.

Also, Munkhdalai and Yu (2016b) proposed much more complicated networks that considered sequen tial LSTM-based encoding, recursive neural networks, and complex combinations of attention models, which provided around 0.5% gain over the results of Parikh et al. (2016).

Besides, although ESIM was the state-of-the-art back to 2016, there are many other new models have gained higher testing scores than ESIM did from then. We choose the current state-of-the-art model, Densely-Connected Recurrent and Co-Attentive Network (Seonhoon Kim et al., 2018) to study their model. As the figure2 shows, each layer of the model uses concatenated information of attentive features as well as hidden features of all the preceding recurrent layers. Therefore, it can preserve the original and the co-attentive feature information from the bottommost word embedding layer to the uppermost recurrent layer. This from-beginning-to-end attention mechanism help them set the new state-of-the-art.

It was not quite clear if the potential of sequential models like RNN had been exploited for NLI when Chen et al., 2016 revisited the problem. This is also why they built the ESIM model.

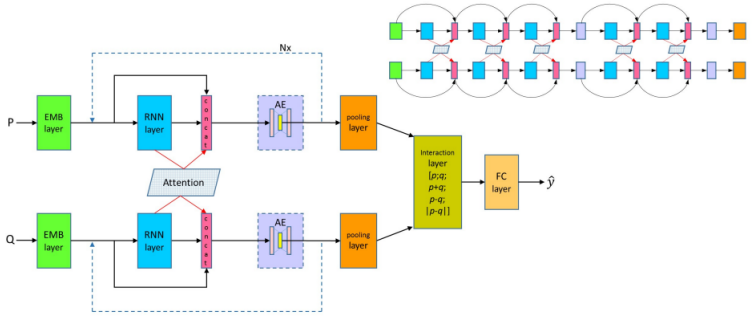


Figure 2: Densely-Connected Recurrent and Co-Attentive Network , Seonhoon Kim et al. '18

3 Background

3.1 FastNLP

We use FastNLP to help build our NLP model. FastNLP is provided by Natural Language Processing Group, Fudan University. It is a modular Natural Language Processing system based on PyTorch, built for fast development of NLP models. A deep learning NLP model in FastNLP is the composition of three types of modules: *encoder*, *aggregator, and decoder.* It can help the users to build some basic natural language models quite easily and fast.

In this project, we mainly use FastNLP to preprocess the dataset(fastNLP.core.dataset), build a vocabulary for the datast(fastNLP.core.vocabulary), loader the pretrained word embedding vectors(fastNLP.io.embed_loader), train the model with the trainer(fastNLP.core.trainer), and test the model(fastNLP.core.tester). We found some bugs in fastNLP. Also we believe that there should be some functions in some models. See more details in Appendix.

3.2 BiLSTM

In ESIM, we will use BiLSTM both in the input encoding and inference composition stage. BiLSTM or Bidirectional LSTM. The structure and the connections of a bidirectional LSTM are represented in figure 4. LSTM is a kind of RNN. Briefly, when modeling a sequence, an LSTM employs a set of

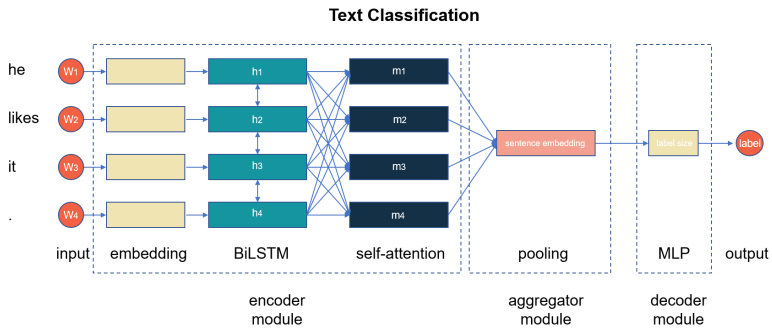


Figure 3: FastNLP for Text Classification

soft gates together with a memory cell to control message flows, resulting in an effective modeling of tracking long-distance information/dependencies in a sequence.

As in BiLSTM, there are two type of connections, one going forward in time, which helps learn from previous representations and another going backwards in time, which helps learn from future representations. Forward propagation is done in two steps: Move from left to right, starting with the initial time step we compute the values until we reach the final time step. Move from right to left, starting with the final time step we compute the values until we reach the initial time step. Also, those two states' output are not connected to inputs of the opposite direction states. Therefore,

$$h = \text{BiLSTM}(X)$$

is equivalent to

$$h_1 = \text{LSTM}_1(X),$$

$$h_2 = \text{LSTM}_2(X),$$

$$h = \text{cat}(h_1, h_2).$$

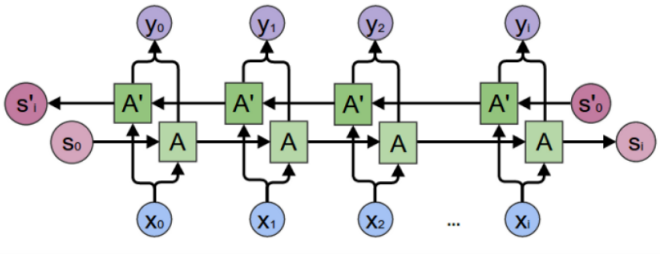


Figure 4: BiLSTM

3.3 Attention Mechanism

Broadly speaking, a neural attention mechanism equips a neural network with the ability to focus on a subset of its inputs (or features), in other words, it selects specific inputs. As the Figure 5 shows, the attention in Seq2Seq model is actually a vector. It helps the decoder to decide which part of encoder is input for current time step. In this way, the decoder pays "attention" to the specific time step in the encoder. While for the alignment, the words in the left pay "attention" to some certian words above - which are also the important words(white area).

Suppose a is a attention vector, for soft attention, it multiplies features with a soft mask of values between zero and one, therefore, $a \in [0, 1]^k$. While for hard attention, those values are constrained to be exactly zero or one, namely $a \in \{0, 1\}^k$.

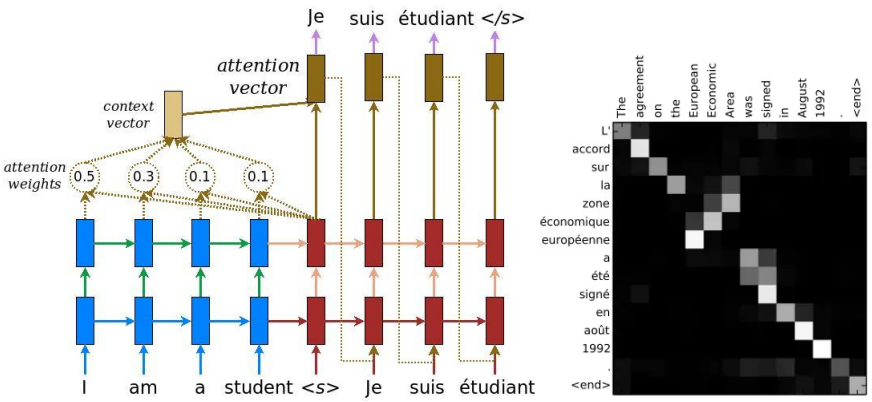


Figure 5: Seq2Seq with attention(left), alignment (right)

3.4 Soft Alignment

Soft attention or soft alignment, is usually used in semantic relatedness measures. In this approach, we can compare the neural word embeddings to compute the relatedness between words across both the sentences, thus producing the soft alignment matrix. The word embedding could be pretrained word vectors or output of some embedding layer. In ESIM, as we will talk as below, we use the hidden states(output) of the BiLSTM to compute soft alignment.

4 ESIM

Enhanced Sequential Inference Model (**ESIM**) are composed of the following major components: input encoding, local inference modeling, and inference composition, as showing in the left part of Figure 6. Considering that Chen et al. spent about 40 hours on Nvidia-Tesla K40M to train tree-LSTM(right part of Figure 6), which means that we might need more than 120 hours to train the tree-ESIM model (the ESIM model that takes Chen et al. about 6 hours costs us more than 18 hours), we don't think that we have enough computing resources and time to train this model. So in this project, we focus on the ESIM.

As for the notation, we denote the two sentences as $\mathbf{a} = (\mathbf{a}_1, ..., \mathbf{a}_{l_a})$ and $\mathbf{b} = (\mathbf{b}_1, ..., \mathbf{b}_{l_b})$, where $\mathbf{a}$ is a premise and $\mathbf{b}$ a hypothesis. The $\mathbf{a}$ or $\mathbf{b}_j \in \mathbb{R}^l$ is an embedding of l -dimensional vector, which can be initialized with pretrained word vectors. The goal is to predict a label y that indicates the logic relationship between$\mathbf{a}$ and $\mathbf{b}$.

4.1 Input Encoding

In this layer, we use BiLSTM to embed the word vectors in premise and hypothesis. We denotethe hidden (output) state given by the BiLSTM as $\bar{\mathbf{a}}_\mathbf{i}$ at time i with respect to the input sequence $\mathbf{a}$. The same is applied to $\bar{\mathbf{b}}_\mathbf{j}$:

$$\bar{\mathbf{a}}_\mathbf{i} = \text{BiLSTM}(\mathbf{a}, i), \forall i \in [1, ..., l_a],$$

$$\bar{\mathbf{b}}_\mathbf{j} = \text{BiLSTM}(\mathbf{b}, j), \forall j \in [1, ..., l_b].$$

In some way, BiLSTM turns the general word vectors into new embeddings, which introduce a lot of useful information about the context. The new embeddings could represent not just the word itself, but also the relationship among words in the same sentence(premise or hypothesis).

4.2 Local Inference Modeling

4.2.1 Locality of inference

Basically, this layer is the most important part in the whole model to determine the overall inference between premise and hypothesis. The model in this layer could help collect local inference for words

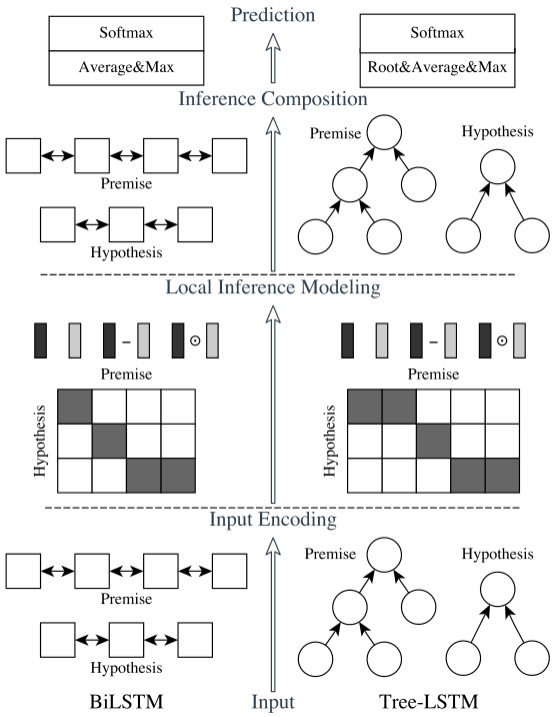


Figure 6: A high-level view of the hybrid neural inference networks

and their context. The soft alignment layer caculates the attention weights as the similarity with respect to a hidden state pair $< \bar{\mathbf{a}}_\mathbf{i}, \bar{\mathbf{b}}_j >$between a premise and a hypothesis as

$$e_{ij} = \bar{\mathbf{a}}_i^T \bar{\mathbf{b}}_j.$$

Rather than using a feedforward neural network like in Parikh et al. (2016) to map the original word representation for computing e_{ij}, ESIM chooses BiLSTM, which encodes the information in premise and hypothesis quite well and achieves better performance. We believe this might introduce the correlation information between words from different sentences(premise and hypothesis).

4.2.2 Local inference collected over sequences

Local inference is used to computer the local relevance between a premise and a hypothesis. For the hidden state of a word in a premise, i.e., $\bar{\mathbf{a}}_\mathbf{i}$ (encoding both the word itself and its context, we can define a new embedding vector denoting the word in premise as the summary of relevant semantics in the hypothesis which is a kind of weighted $\bar{\mathbf{b}}_\mathbf{j}$:

$$\tilde{\mathbf{a}}_i = \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{ik})} \bar{\mathbf{b}}_j, \forall i \in [1, ..., l_a],$$

and it is the same for the new defined hypothesis word vector:

$$\tilde{\mathbf{b}}_j = \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{kj})} \bar{\mathbf{a}}_i, \forall i \in [1, ..., l_a],$$

In this stage, the embedding vectors encode not just the word itself, but also the context information in the same sentence, and the relevance information of the other sentence.

4.2.3 Enhancement of local inference information

Up to now, we have two types of new embedding vectors, $\bar{\mathbf{a}}_\mathbf{i}$ and$\tilde{\mathbf{a}}_i$. To enhance the local inference, we should make maximum use of these vectors. We compute the difference and the element-wise product for the tuple $< \bar{\mathbf{a}}, \tilde{\mathbf{a}} >$ as well as for $< \bar{\mathbf{b}}, \tilde{\mathbf{b}} >$.We expect that such operations could help sharpen local inference information betwen elements in the tuples, thus capture the inference relationships such as contradiction. The difference and element-wise product are then concatenated with the original vectors,

$$\mathbf{m}_a = [\bar{\mathbf{a}}; \tilde{\mathbf{a}}; \bar{\mathbf{a}} - \tilde{\mathbf{a}}; \bar{\mathbf{a}} \odot \tilde{\mathbf{a}}]$$

$$\mathbf{m}_b = [\bar{\mathbf{b}}; \tilde{\mathbf{b}}; \bar{\mathbf{b}} - \tilde{\mathbf{b}}; \bar{\mathbf{b}} \odot \tilde{\mathbf{b}}]$$

$\mathbf{m_a}$and$\mathbf{m}_b$would be then fed into the inference composition layer to predict the final inference label.

4.3 Inference Composition

To obtain the overall inference relationship between the premise and the hypothesis, we build a composition layer to compose the enhanced local inference information$\mathbf{m_a}$and$\mathbf{m}_b$.

4.3.1 The composition layer

Still using BiLSTM here, we could compose local inference information sequentially. More specifically, we use a 1-layer feedforward neural network with the ReLU activation.

4.3.2 Pooling

We consider both average and max pooling here and concatenate all these pooled vectors to form a final length fixxing vector v. v can be computed as followsing:

$$\mathbf{v}_{a,\text{ave}} = \sum_{i=1}^{l_a} \frac{\mathbf{v}_{a,i}}{l_a}, \mathbf{v}_{a,\text{max}} = \max_{i=1}^{l_a} \mathbf{v}_{a,i}$$

$$\mathbf{v}_{b,\text{ave}} = \sum_{j=1}^{l_b} \frac{\mathbf{v}_{b,j}}{l_b}, \mathbf{v}_{b,\text{max}} = \max_{j=1}^{l_b} \mathbf{v}_{b,j}$$

$$\mathbf{v} = [\mathbf{v}_{a,\text{ave}}; \mathbf{v}_{a,\text{max}}; \mathbf{v}_{b,\text{ave}}; \mathbf{v}_{b.\text{max}}]$$

4.3.3 Prediction and our improvement

We feed $\mathbf{v}$ into a final multilayer perceptron (MLP) classifier. In Chen et al's model, the MLP has a hidden layer with tanh as the activation function and softmax as the output layer. We believe that the MLP would be a good layer for fine-tuning. After different experiments, we find that a three-layer MLP with ELU as the activation function could provide a great test score.

5 Experiment Setup

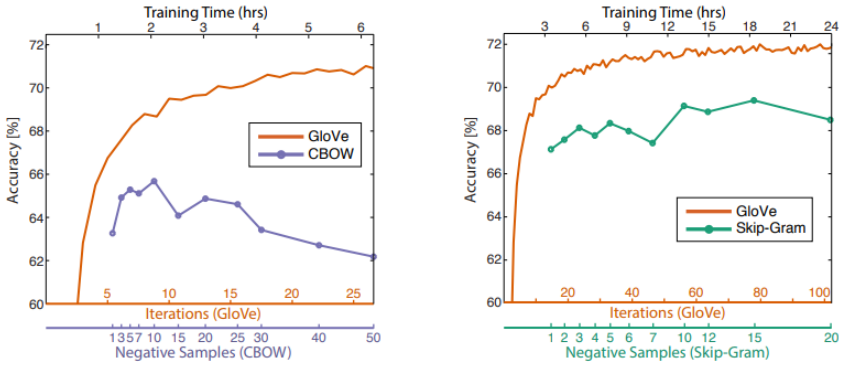
5.1 Data

In our experiment, we test ESIM both on English and Chinese corpus. For English experiments, we use GloVe as the pretrained embedding weights. For the NLI corpus in English experiments, we test both on Staford NLI and Multi-Genre NLI. For the Chinese experiment, we use the pretrained SGNS Wikipedia embedding introduced in Analogical Reasoning on Chinese Morphological and Semantic Relations. (Shen Li et al., 2018) For the NLI corpus in Chinese experiment, we use Chinese Natrual Language Inference corpus provided by Beijing Language and Culture University. [1]

[1] See more details on https://github.com/blcunlp/CNLI

5.1.1 GloVe

GloVe, or Global Vectors for Word Representation, is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Usually, we use the pre-trained GloVe word embedding provided by Stanford University. From Figure 7, we could find that GloVe performs better than the traditional word2vec embedding models CBoW and Skip-Gram.



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 7: GloVe vs Word2Vec

5.1.2 Chinese Word Embedding

We use the Wikipedia Chinese word vector mentioned in Analogical Reasoning on Chinese Morphological and Semantic Relations.[2](Shen Li et al., 2018)[2]

Corpus	Size	#tokens	$\|V\|$	Description
Wikipedia	1.3G	223M	2129K	Wikipedia data obtained from https://dumps.wikimedia.org/
Baidubaike	4.1G	745M	5422K	Chinese wikipedia data from https://baike.baidu.com/
People's Daily News	3.9G	668M	1664K	News data from People's Daily (1946-2017) http://data.people.com.cn/
Sogou news	3.7G	649M	1226K	News data provided by Sogou Labs http://www.sogou.com/labs/
Zhihu QA	2.1G	384M	1117K	Chinese QA data from https://www.zhihu.com/, including 32137 questions and 3239114 answers
Combination	14.8G	2668M	8175K	We build this corpus by combining the above corpora

Figure 8: Chinese Word Vector

There are many different pretrained word vectors based on different corpora incuding Wikipedia, Baidubaike, Sogou News, People's Daily, Zhihu QA, Financial News, Weibo, etc. Also, different word vectors have different performances as the Figure 9 shows. Howover, considering the importing speed, we choose the minimal one - Wikipedia. Besides, the data format of Chinese word vector is the same as GloVe. Therefore, we could use fastNLP to load the embedding vector directly.

[2]Download the pretrained Chinese word vectors here https://github.com/Embedding/Chinese-Word-Vectors

	CA_translated			CA8									
	Cap.	Sta.	Fam.	A	AB	Pre.	Suf.	**Mor.**	Geo.	His.	Nat.	Peo.	**Sem.**
Wikipedia 1.2G	.597	.771	.360	.029	.018	.152	.266	.180	.339	.125	.147	.079	.236
Baidubaike 4.3G	.706	.966	.603	.117	.162	.181	**.389**	.222	.414	**.345**	.236	**.223**	.327
People's Daily 4.2G	**.925**	**.989**	.547	.140	.158	**.213**	.355	**.226**	**.694**	.019	.206	.157	**.455**
Sogou News 4.0G	.619	.966	.496	.057	.075	.131	.176	.115	.432	.067	.150	.145	.302
Zhihu QA 2.2G	.277	.491	**.625**	**.175**	**.199**	.134	.251	.189	.146	.147	**.250**	.189	.181
Combination 15.9G	**.872**	**.994**	**.710**	**.223**	**.300**	**.234**	**.518**	**.321**	**.662**	**.293**	**.310**	**.307**	**.467**

Figure 9: Different word vectors' performances on CA_translated and CA8.

5.1.3 Stanford NLI

The SNLI corpus is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference (NLI), also known as recognizing textual entailment (RTE). The aim is to serve both as a benchmark for evaluating representational systems for text, especially including those induced by representation learning methods, as well as a resource for developing NLP models of any kind.

Actually, there are 5 labels in each NLI line. If any one of the three labels was chosen by at least three of the five annotators, it was chosen as the *gold label*. If there was no such consensus, which occurred in about 2% of cases, we assigned the placeholder label '-'. In our project, we just use the *gold label*. There's an SNLI example in the Introduction part already, so we won't give another example here.

5.1.4 Multi-Genre NLI

The Multi-Genre Natural Language Inference (MultiNLI) corpus is a crowd-sourced collection of 433k sentence pairs annotated with textual entailment information. The corpus is modeled on the SNLI corpus, but differs in that covers a range of genres of spoken and written text, and supports a distinctive cross-genre generalization evaluation.

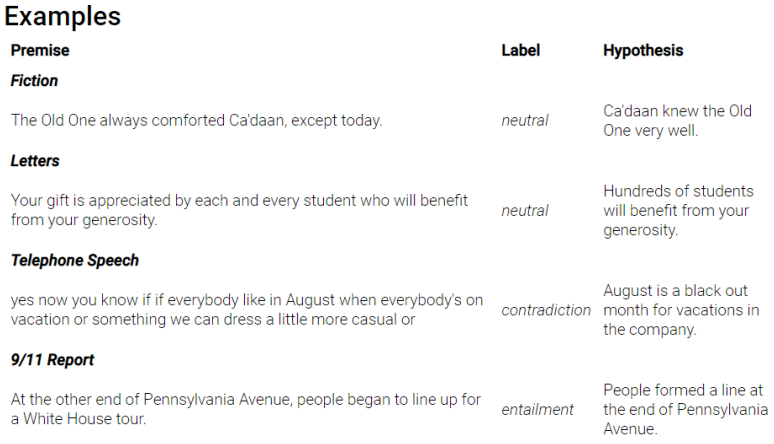


Figure 10: Multi-Genre NLI Corpus

As we can see from examples in Figure 10, different from the SNLI, in which the sentences are derived from only a single text genre image captions, there are many different genres here: *GOVERNMENT, SLATE, TELEPHONE, TRAVEL, FICTION, and 9/11 REPORT, FACE-TO-FACE, LETTERS, OUP,VERBATIM.*

Genre	#Examples Train	Dev.	Test
SNLI	*550,152*	*10,000*	*10,000*
FICTION	77,348	2,000	2,000
GOVERNMENT	77,350	2,000	2,000
SLATE	77,306	2,000	2,000
TELEPHONE	83,348	2,000	2,000
TRAVEL	77,350	2,000	2,000
9/11	0	2,000	2,000
FACE-TO-FACE	0	2,000	2,000
LETTERS	0	2,000	2,000
OUP	0	2,000	2,000
VERBATIM	0	2,000	2,000
MultiNLI Overall	**392,702**	**20,000**	**20,000**

Figure 11: Multi-Genre NLI Corpus

All of the genres appear in the test and development sets, but only five are included in the training set(the first five). Models thus can be evaluated on both the matched test examples, which are derived from the same sources as those in the training set, and on the mismatched examples. The dev dataset is split into matched and dismatched part. In this project, we test on both the matched and mismatched.

5.1.5 Chinese NLI

CNLI dataset is firstly provided for the Chinese Natural Language Inference (CNLI) share task on The Seventeenth China National Conference on Computational Linguistics, CCL 2018[3]

Both the train and dev set in CNLI are tab-separated format. Each line in the train (or dev) file corresponds to an instance, and it is arranged as*: sentence-id, premise, hypothesis, label.* Here are some examples.

1. p: 一个小男孩正在打一个黄色的球。
   h: 一个小男孩在读书。
   Relationship: *Contradiction*
2. p: 一个小男孩在铁轨上行走。
   h: 一个小男孩坐火车。
   Relationship: *Neutral*
3. p: 一对夫妇正在观看*3D*电影。
   h: 有些人正在看电影。
   Relationship: *Entailment*

5.2 Experiment

Considering that all the datasets provide training, dev and test set, we don't have to split the dataset on our own. As for the parameters setting for the training stage, we keep the same as Chen et al. did in the original paper. The parameters setting are as follows:

- batch size: 32
- learning rate: initially 0.0004
- optimizer: Adam method with 0.9 as the first momentum and 0.999 the sencond
- embeddding dimension: 300
- hidden size in BiLSTM: 300
- num of linear layer in MLP: 3

[3]Task3, see more details on http://www.cips-cl.org/static/CCL2018/call-evaluation.html

- dropout rate: 0.5
- activate function: ELU
- hidden size in MLP: 300
- padding index: 0
- out of vocabulary index: 1

Besides, there's a built-in SNLI model in FastNLP. The forward function of this model is: forward(*premise, hypothesis, premise_len, hypothesis_len*), where the *premise_len* is a mask Tensor of *premise*, recording if the word is a padding. However, we are not quite sure how to get a mask Tensor as the input with the only usage of FastNLP, considering the padding function is encapsulated in FastNLP.

Also, we noticed that there's no loading pretrained word embedding function in the built-in SNLI model. So we add another set of experiments where we don't do word embedding. Instead, we use the index of each word in the dataset as the input.

6 Result

6.1 Result for English Experiment

Considering the dataset is very large, training is quite time consuming. Hence, we just let the ESIM model on SNLI run more than 40 epochs. For those models on MultiNLI, we just run about 10 epochs.

Here's the result for SNLI.

Model	dev score	test score
our ESIM	83.66	82.72
our ESIM, no emb	89.62	89.70
ESIM, Chen et al.	92.60	88.00
Densely-Connected Recurrent and Co-Attentive Network*	95.0	90.10

Table 1: Result for SNLI, *current state-of-the-art

Here's the result for MultiNLI. Notice that MultiNLI isn't avalaible yet when Chen et al. proposed ESIM in 2016.

Model	dev score	test score, matched	test score, mismatched
Most Frequent Class	-	36.50	35.60
CBoW	-	65.20	64.60
BiLSTM	-	67.50	67.10
ESIM, Bowman et al.	-	**72.40**	**72.90**
Our ESIM	**68.62**	**68.78**	**68.79**
Our ESIM, no emb	67.03	66.72	67.88

Table 2: Result for MultiNLI

Noticed that out ESIM doesn't perform well as Boman et al.'s ESIM does. This is because that we don't have enough time to train the model on the MultiNLI dataset. Nonetheless, out owe-trained model still beats all the three other models. This suggests that ESIM indeed is a strong model in the Natural Language Inference field.

6.2 Result for Chinese Experiment

You can see this page to view the current evaluation result in the CNLI2018 competetion.

The current highest test score in the CNLI2018 competetion is 0.8238. Our test score is way much higher than the current top1 model. Therefore, we could say that ESIM indeed works quite well not just for English corpus, but also on Chinese NLI task.

Team	Model	Test Score
Our Team	**ESIM**	**95.21**
Out Team	**ESIM, no emb**	**92.10**
water	cnn+lstm	82.38
zzunlp	nlpc	78.28
BaiduZhizhu	Excailibur	76.92
GDUFSER	-	76.18
ray_li	-	74.25
INTSIG_AI	-	73.03
Yonseiiii	decom-att	72.42
Baseline	ESIM	72.22
_503	bi	68.48
Hiter	DAM	60.90

Table 3: Result for CNLI

According to all the experiments above, we foud that no matter what the dataset is, or no matter what the language is, word embedding can always enhance the performance. This also suggests that the FastNLP built-in SNLI model might consider to add this function.

From the result above, we could notice that all the models' performance on Multi-Genre NLI corpus are not much impressive, which suggests there's still a lot of potential of this specific task to be exploited.

7 Feature Work

According to time and resources limitation, we could not train a Tree-ESIM in this project. But we are extremely interested in encoding syntactic information in the ESIM model. Moreover, there are a banch of ways to introducing syntactic information into a NLP model such as multi-task learning with POS, etc. We want to investigate if all forms of syntactic information adding would help enhance the performance, or just the tree structured LSTM based on a parsing tree would work.

References

[1] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, Vancouver, July 2017. ACL. 1

[2] Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143. Association for Computational Linguistics, 2018. 5.1.2

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 1

[4] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. 1