CS839 COURSE PROJECT

Named Entities Recognition (NER) and Relation Extraction (RE) For Diabetes Research Literature

December 19, 2018

Jifan Gao, Biomedical Data Science, M.S.
Peilin Yu, Computer Science, B.S.
Shuoxuan Dong, Computer Science, M.S.
Xiaotian Li, Civil Engeering Ph.D

Contents

1 Named Entity Extraction (NER) from diabetes research literature **4**

1.1 Related work 4

1.2 Approach 6

1.2.1 Data 6

1.2.2 Evaluation 7

1.2.3 Preprocessing 7

1.2.4 Model 8

1.3 Results 8

1.4 Discussion 9

2 Relation Extraction (RE) from diabetes research literature **10**

2.1 Related work 10

2.2 Approach 12

2.2.1 Data 12

2.2.2 Evaluation 13

2.2.3 Preprocessing 13

2.2.4 Model 14

2.3 Results 15

2.4 Discussion 15

Work Breakdown **18**

Introduction

ALIBABA'S NLP COMPETITION

Probabilistic Graphical Model (PGM) and Deep Learning (DL) have important applications in the area of Natural Language Processing (NLP) such as Named Entity Recognition (NER) and Relation Extraction (RE). In order to gain practical experience on PGM and deep learning, our initial plan is to attend an English-based NLP competition. However, such competition with suitable schedule is not available in autumn 2018. What we in fact attended is the Diabetes-Related Knowledge Base Construction Competition which is a Chinese-based NLP competition held by Alibaba Group. The competition has 2 stages where the topic of the first stage is Named Entity Recognition from diabetes literature and the topic of the second stage is Relation Extraction from these literature. The top 100 from the total 1629 participated teams are qualified for the second stage. On the final leaderboard, we finished at 72nd in the first stage and 39th in the second stage.

Our project consists of two main topic: Named Entity Recognition (NER) and Relation Extraction (RE). We implement a BiLSTM+CRF model for the NER task and this model achieves an F1 score of 0.717. For the RE task, we built a stacking model which combines outputs from one BiLSTM+Attention model and one PCNN+Attention model as well as some hand-crafted features. This stacking model finally achieves an F1 score of 0.625 on the leaderboard.

SIGNIFICANCE

In Stage 2, our stacking model outperforms single deep learning models meanwhile it runs much faster. Although a stacking approach is not "academically elegant", our success implementation of a stacking model may indicate its ability to solve real world problems in an

more efficient way.

For us as students, we've obtained practical experience from this project. In Stage 1, we apply a CRF layer on the top of a deep learning framework for a sequence tagging task. We apply HMMs learned in class to conduct word segmentation for a word-based model. In Stage 2, we implement and modify some of the deep learning models covered in the second half semester. Our theoretical knowledge from CS839 lecture is consolidated through this competition.

Chapter 1

Named Entity Extraction (NER) from diabetes research literature

1.1 Related work

Our model uses a random embedding at character-level and a BiLSTM + CRF model proposed by Huang et al [3]. In Huang's work, they compare varieties of LSTM based models for sequence tagging. The models inlcude LSTM network, LSTM with a CRF layer and a Bi-LSTM with a CRF layer model. The proposed Bi-LSTM CRF model produced highest accuracy on the benchmark dataset.

A RNN maintains a memory based on history information, which enables the model to predict the current output conditioned on long distance features. Long Short-Term Memory networks are the same as RNNs, except that the hidden layer updates are replaced by purpose-built memory cells. A LSTM memory cell is implemented as the following:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{1.1}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{1.2}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{1.3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{1.4}$$

$$h_t = o_t tanh(c_t) \tag{1.5}$$

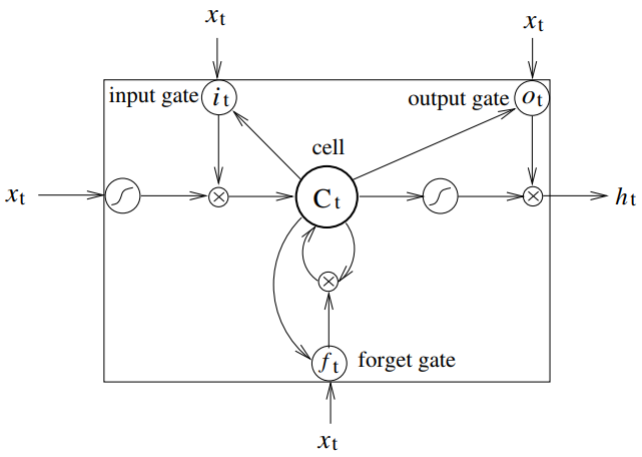

Figure 1.1: A LSTM cell[3]

The LSTM network is shown in Figure 1.2. In NER task, we have access to both past and future

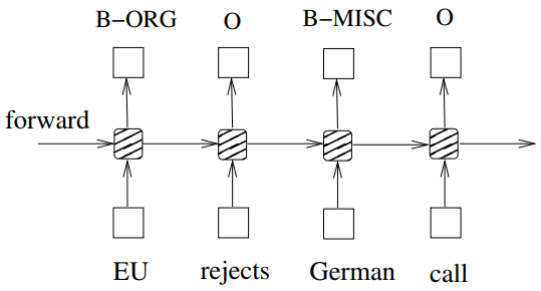

Figure 1.2: LSTM network[3]

input features for a given time, thus bidirectional LSTM (aka Bi-LSTM) network is a proper tool to utilize. We can efficiently use past features and future features for a specific time frame. A Bi-LSTM Conditional Random Field is beneficial for this named-entity recognition task. The LSTM tagger is typically sufficient for part-of-speech tagging, but a sequence model like the CRF is really essential for strong performance on NER. The CRF computes a conditional probability. Let y be a tag sequence and x an input sequence of words. Then we computer

$$P(y|x) = \frac{exp(Score(x,y))}{\Sigma_{y'} exp(Score(x,y'))} \tag{1.6}$$

The score is determined by defining some log potentials $log\phi_i(x,y)$ such that

$$Score(x,y) = \Sigma_i log\phi_i(x,y) \tag{1.7}$$

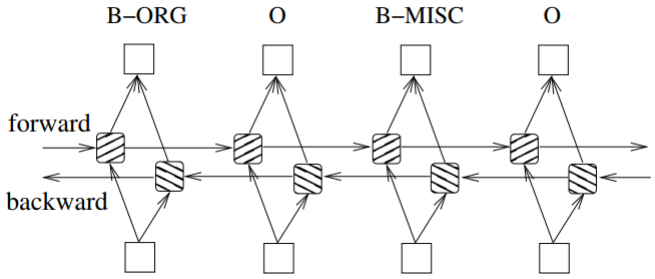

Figure 1.3: A Bi-LSTM-CRF model[3]

The CRF layer from Keras-contrib package is used for building the CRF on top of the Bi-LSTM model. We use "join mode" for training in the CRF layer. It does optimization by maximizing join likelihood, which is optimal in theory of statistics. Viterbi output is better than marginal output since we choose "join mode" for training.We first initialize the viterbi variables in log space. Then we let forward_var at step *i* holds the viterbi variables for step *i-1*. Then we transit it to STOP_TAG and follow the back pointers to decode the best path. At last, we pop off the start tag.

1.2 Approach

1.2.1 Data

We are provided with a training set that has tagged entities. The dataset is from diabetes related research literature in language of Chinese, with annotation labels as ground truth. The dataset contains 363 medical articles with 26906 sentences, which is 1396663 Chinese characters in total. There are 140028 training ground truth annotation labels in the format of "T14 Test 89 94 HBA1C".

Below is an example of the given information:

T14 Test 89 94 HBA1C

where "T14" is the index of the entity, "Test" is the class of the entity, "89 94" is the starting and ending position of the entity in the article, and "HBA1C" is the named entity.

The provided annotations can be divided into 15 classes include: five disease related classes: Disease, Reason, Symptom, Test, Test_value. Seven treatment related classes: Drug, Frequency, Amount, Method, Treatment, Operation, Side_Eff. Three human-body related classes:

Anatomy, Level, Duration.

1.2.2 Evaluation

The evaluation matrix is F1 score. If an entity is predicted with correct positions in an article, it is considered to be a true positive prediction no matter what type we predict. Then we would have:

$$precision = \frac{\text{number of true positive predictions}}{\text{number of total predictions}} \tag{1.8}$$

$$recall = \frac{\text{number of true positive predictions}}{\text{number of entities in ground truth}} \tag{1.9}$$

$$F1\_score = \frac{2 \times precision \times recall}{precision + recall} \tag{1.10}$$

1.2.3 Preprocessing

The training set includes 363 diabetes-related medical articles. The information of entities in each article is provided in addition. We firstly transform each article and its entity labels to tagged sentences. For example,

H	B	A	1	C	控	制	目	标	的	专	家	共	识
B-Test	I-Test	I-Test	I-Test	I-Test	0	0	0	0	0	0	0	0	0

where "HBA1C" is an entity and others are not.

Besides, we figure out an approach to correct false negative tags in the training set. For example, "HBA1C" is tagged as a "Test" in most articles. But in a few articles, it is not assigned to be a "Test". We can correct these "missing tags" but correcting in an arbitrary way raises another problem: there are in fact some noise in the given tags and therefore, the noise level will increase hugely. For example, in one article "2" is tagged as a “Test_value”. Tens of thousands of "2" will be labelled if we arbitrarily apply the correction mentioned above, which does not make sense. Here we fixed this issue by using a simple threshold:

$$p_{ent} = \frac{\text{how many times it is tagged}}{\text{how many times it is not tagged}} \tag{1.11}$$

If the p_{ent} of an entity is less than 0.8, we add tags to its unlabelled cases. Otherwise we do

nothing to this entity. The submitted results shows that such correction helps to increase the F1 score by about 0.03

1.2.4 Model

As is mentioned before, we re-implemented Huang's methods [3] which uses a random embedding at character-level and a BiLSTM+CRF model. The pipeline is realized using Keras with Tensorflow backend and hyperparameters are fine tuned.

1.3 Results

A F1 score of 0.717 is achieved and we finished at 72nd place in the final leaderboard of Stage 1, which guaranteed us to proceed to the next stage.

Rank	Participant	Organization	score	Best Submission Date
1	这次要好好搞一下啊	Shanghai University	0.763	2018-11-12
2	身体是革命的本钱	Harbin Institute of Technology	0.762	2018-11-12
3	wowjoy	华卓科技	0.762	2018-11-12
4	大雷wowjoy	下线中	0.762	2018-11-12
5 ↑1	S.S.S	Harbin Institute of Technology	0.761	2018-11-13
6 ↑29	保福寺僧团	Chinese Academy of Sciences	0.761	2018-11-13
7	SuperGuts	Jiangnan University	0.761	2018-11-13
8 ↑4	biu biu biu	Fudan University	0.760	2018-11-13
9 ↓4	苏小易	优易数据	0.760	2018-11-12
10 ↑18	kakalahm	Tsinghua University	0.758	2018-11-13
		.....		
72 ↓19	ggggfan	Other Overseas regions-YJR	0.713	2018-11-13

Figure 1.4: The final leaderboard of Stage 1

1.4 DISCUSSION

In addition to our character-based model, we designed a word-based model. Unlike English, Chinese words may contain multiple characters and English has a natural word boundary as segmentation marks - the space between words. It is, therefore, reasonable to consider a word-level NER model and we conjectured it to outperform our character-based approach if under similar parameter settings for the end model. We propose a pipeline to first conduct word segmentation with classic HMMs over corpus and then use canonical Word2Vec-based word embeddings. Finally, we would reuse our implementation of BiLSTM-CRF model to learn NER at a word level.

For the word segmentation, we implemented a bi-gram HMM-based method, a classic, simple yet effective approach widely recognized by the Chinese NLP communities. We define three categories of hidden states: Begin(B), Middle(M) and End(E). In essence, we wish to correctly infer and assign hidden states B,M,E to each observed state O_i, namely each character from the observed sequence of Chinese characters. To infer the most likely hidden sequence O, We follow the canonical approach for bi-gram HMMs and learn the most likely hidden assignments with **VITERBI** algorithm. With word-segmentation, we used a Word2Vec-based word embeddings to preprocess and prepared the corpus for the end NER model.

However, contrary to our intuition, initial experiments with word-segmentation models trained on mass Chinese corpus fail to yield a better result on the end BiLSTM-CRF. We conclude that the reason behind might be from the specificity of our corpus. Due to the limit of time, we are not able to conduct extensive experiments on the word-based models but we could expect future experiments with regard to the word segmentaiton model and the word embeddings.

For future extension, it would be reasonable to apply Snorkel [5], which is a effective system for modeling training data, in order to obtain more training data. Single entities will be used as candidate objects in Snorkel. In addition, using BERT [2], which is newly proposed transformer based model, in the sequence tagging problem of this competition seems to be reasonable if adequate computing recourse is available.

Chapter 2

Relation Extraction (RE) from diabetes research literature

2.1 RELATED WORK

The purpose of a Relation Extraction task is to detect and classify semantic connections between entities in text. Etzioni et al. [1] summarize all methods used for Relation Extraction into three categories: knowledge-based methods, supervised methods, and self-supervised methods. Our project uses a stacking model which combines two sentence-level supervised methods: Piecewise Convolutional Neural Networks (PCNN) [7] and Attention-Based Bidirectional Long Short-Term Memory Networks (BiLSTM+ATT) [8].

Zeng et al. [7]proposed Piecewise Convolutional Neural Networks for distant-supervised relation extraction in 2015. A Piecewise Convolutional Neural Network is similar with a Convolutional Neural Network except the pooling procedure. The authors argue that traditional pooling is not sufficient to capture the structural information between two entities. Since the input sentence can be divided into three segments based on the two named entities, they propose a piecewise max pooling procedure that returns the maximum value in each segment. Symbolically,

$$p_{ij} = max(\mathbf{c}_{ij}), 1 \leq i \leq n, 1 \leq j \leq 3 \tag{2.1}$$

where $\mathbf{c}_{ij}$ is the output vector from ith filter in the jth segment.

Thus, we can obtain a 3-dimensional vector $\mathbf{p}_i = \{p_{i1}, p_{i2}, p_{i3}\}$ from each filter. All vectors $\mathbf{p}_{1:n}$ are concatenated and fed into a non-linear function. Then we can use softmax classifier to

compute the confidence of each relation. The architecture sketch of a piecewise max pooling procedure is displayed in Figure 2.1.

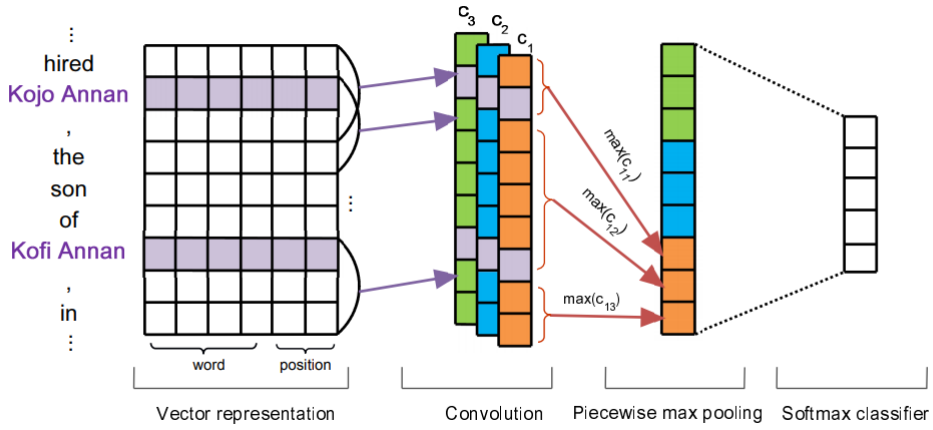

Figure 2.1: The architecture of PCNNs from Zeng et al [7]

The Attention-Based Bidirectional Long Short-Term Memory Networks [8] is introduced by Zhou et al. in 2016. As shown in Figure 2.2, the BiLSTM+Attention based model for relation extraction contains five components: Input layer, Embedding layer, BiLSTM layer, Attention layer, and Output layer. The purpose of using an attention layer is to capture the most important semantic information in a sentence. The attention layer produces a weight vector, and merge word/character-level features generated from BiLSTM into a sentence-level feature vector by multiplying the weight vector. Symbolically, let H be the output matrix consisting of output vectors $[h_1, h_2, ..., h_T]$ generated from BiLSTM, where T is the length of the input sentence. The representation r can be expressed as a weighted sum of these output vectors:

$$M = tanh(H) \tag{2.2}$$

$$\alpha = softmax(w^T M) \tag{2.3}$$

$$r = H\alpha^T \tag{2.4}$$

where $H \in \mathbb{R}^{d^w \times T}$, w is the trained weights vector, d^w is the dimension of the embedding vectors. The dimension of w, α, r is d^w, T, d^w respectively.

LightGBM is a boosting tree based algorithm which is welcomed by data scientists due to its high-accuracy. It is proposed by Ke et al. in 2016 and uses two novel techniques called Gradient-based One-Side Sampling and Exclusive Feature Bundling to deal with large number of data instances and large number of features respectively [4]. In this stage, we combine

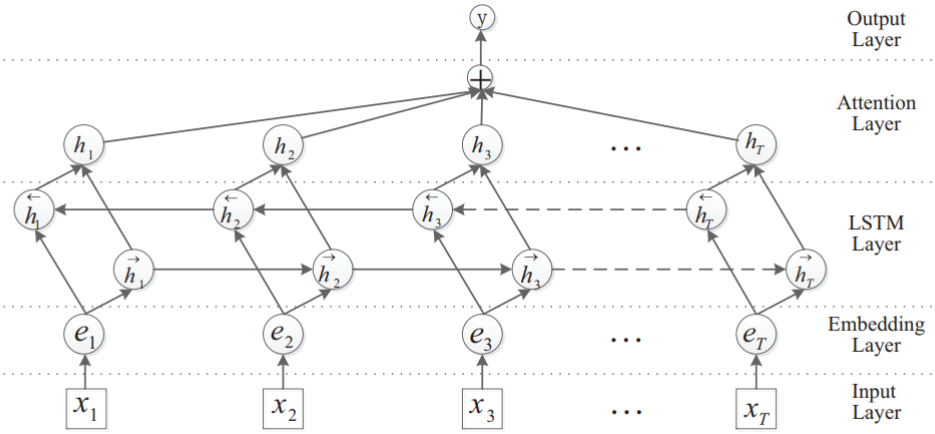

Figure 2.2: The architecture of LSTM+Attention from Zhou et al [8]

outputs from deep learning models and use these outputs to train a LightGBM classifier.

2.2 Approach

2.2.1 Data

The literature set for training and testing are the same as in the Stage 1. The entity types are consistent with Stage 1. The type and location of each entity is available in both training and testing set. In addition, the relationship between pairs of entities at certain locations are provided in the training. For example, between two entities whose indices are "T1" and "T5", types are "Disease" and "Test", locations in the corresponding literature are from 30th character to 35th character and 45th character to 50th character:

T1	Disease	30	35	2型糖尿病
T5	Test	45	50	HBA1C

there exists a "Test-Disease" relation:

R1	Test_Disease	Arg1:T1	Arg2:T5

where "R1" is the index, "Arg1" and "Arg2" are the indices of the involved entities.

There are 10 types of relations in the training set: "Test_Disease", "Symptom_Disease", "Treatment_Disease", "Drug_Disease", "Anatomy_Disease", "Frequency_Drug", "Duration_Drug", "Amount_Drug", "Method_Drug", and "SideEff_Drug". The aim of Stage 2 is to predict these relations from the testing set.

2.2.2 Evaluation

The evaluation matrix of this stage is F1 score either, which can be computed as:

$$precision = \frac{\text{number of corrected prediction}}{\text{number of total prediction}} \tag{2.5}$$

$$recall = \frac{\text{number of corrected prediction}}{\text{number of total relations in ground truth}} \tag{2.6}$$

$$F1\_score = \frac{2 \times precision \times recall}{precision + recall} \tag{2.7}$$

2.2.3 Preprocessing

Due to the limit of GPU memory, it is not feasible to embed a whole article or even long paragraphs. Therefore, the first step is to find sentence-level samples that contain pairs of entities where a relationship is possibly existing. We extract sentences containing two entities where the distance between the entities is less than 100 character. This procedure covers 85% of relations in the ground truth.

Furthermore, we continue to clean the extracted sentence based on the types of the entity pairs. In this dataset, each relation type indicates the entities' types. Therefore, we can exclude those sentences where the types of entities do not match one of the 10 given relations. For example, if a sentence contains two entities whose types are both "SideEff", we can throw away this sentence, because "SideEff_SideEff" is not contained in the ten given relations.

After this cleaning procedure, 26% of the relations contained in the cleaned samples are in fact correct compared with the ground truth. Note that the sample covers 85% of relations in the ground truth. That means, the precision of the cleaned samples is 26% and the recall is 85%. Thus, we can compute the F1 score as: $2 \times 0.85 \times 0.26/(0.85 + 0.26) = 0.398$. In other words, just by preprocessing we may achieve a F1 score around 0.40.

2.2.4 Model

Embedding

Since the character-based model outperforms the word-based model in Stage 1 on the same dataset, we continue using character embedding in Stage 2. In addition to the matrix of character indices, we also build two matrices which contains the relative distance from each character to two entities in the input sentence. Same as in Stage 1, we apply random embedding for each of the three metrics on character-level and then concatenate the three embedding metrics. The concatenated matrix will be fed into the deep learning model described in the next paragraph.

BiLSTM+Attention and PCNN+Attention

The pipeline of BiLSTM+ Attention is reimplemented according Zhou's paper [8]. For the PCNN+ Attention model, we first apply PCNN following Zeng's [7] procedure. Then we build an attention layer on the top of the output of piecewise max pooling.

Stacking

As shown in section 2.3, neither of the deep learning models achieves a F1 score over 0.60 on local validation. Therefore, we build a stacking model with an expectation of boosting the performance from single models. In order to maintain features' independency, BiLSTM+Attention and PCNN+Attention are trained on different training samples. The sketch of the stacking model is shown in Figure 2.3. In the training process we use 30% data to train a PCNN+Attention model and then another 30% to train BiLSTM+Attention model. We create 5 features from each sentence including: types of entities, sentence length, distance between entities, number of [‘ ! ’, ‘ 。 ’, ‘ ? ‘] between entities, and sum number of spaces and line-breaks between entities. Together with these features, the probability outputs of BiLSTM+Attention and PCNN+Attention models are treated as features to train a LightGBM classifier on the remaining 40% data.

In the testing process, the sample is firstly fed into the BiLSTM+Attention and PCNN+ Attention to generate features. These features are then concatenated with the five hand-crafted features and fed into the LightGBM classifier for the final prediction.

Note: In this section, "train" means "train and validate on a 9:1 split".

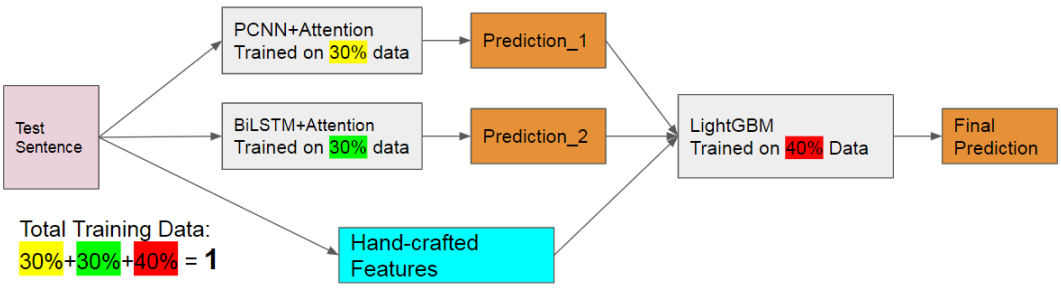

Figure 2.3: The stacking model

2.3 Results

The local validation scores and the online test scores are displayed on Table 2.2. Since there are only two submission chances for scoring on the final leaderboard, we choose PCNN+Attention model, which is the best performer among single models, and the stacking model for submission.

Table 2.1: Results

Method	Local Validation	Leaderboard	Ranking
BiLSTM+ATT	0.521	-	-
PCNN+ATT	0.544	0.553	-
LGB: 5000 epochs	0.619	0.625	39th
LGB: 50000 epochs	0.733	?	?

The local validation scores are quite consistent with online test scores on the leaderboard. The best local score comes from the stacking model with 50000 epochs on LightGBM. However, due to a stupid careless mistake, the stacking model with only 5000 epoch, which is far from convergence, is submitted for the final score. This model achieved an F1 score of 0.625 and 39th place on the leaderboard. A screen-shot of top teams and our ranking is displayed as Figure 2.4

2.4 Discussion

In this stage, we re-implement the BiLSTM+Attention model and added an attention layer on the top of a PCNN model. Both of the models achieve reasonable performance but there

Rank	Participant	Organization	score	Best Submission Date
1	身体是革命的本钱	Harbin Institute of Technology	0.787	2018-12-06
2	SuperGuts	Jiangnan University	0.773	2018-12-05
3 ↑4	megemini	无	0.770	2018-12-05
4 ↓1	yoyoyo	Peking University	0.768	2018-12-06
5 ↓1	Tangguo	成都小象科技	0.761	2018-12-05
6 ↓1	今晚上山打老虎	打虎联盟	0.760	2018-12-05
7 ↑1	一贫如洗	Chinese Academy of Sciences	0.759	2018-12-06
8 ↓2	大白大白	Hunan University	0.755	2018-12-05
9 ↑1	气味儿3	Zhejiang University	0.751	2018-12-06
10 ↓1	苏小易	优易数据	0.749	2018-12-04
		.....		
39 ↓2	ggggfan	Other Overseas regions-YJR	0.625	2018-12-06

Figure 2.4: The final leaderboard of Stage 2

is still a gap from top teams. In order to further improve our scores, we design a stacking model which combines deep learning outputs and five hand-crafted features. In the stacking model, the single BiLSTM+Attention and PCNN+Attention model are actually not expected to achieve high validation scores with 30% training data. In fact, a small number of epoch is used for training deep learning models in the stacking procedure in order to prevent overfitting. The purpose of training BiLSTM+Attention and PCNN+Attention model on different data is to maintain the independency among features and make the stacking model less prone to overfitting. The best stacking model achieved a f1 score of 0.733 locally but we could not figure out how it would be on the leaderboard. However, even a stacking model that is far from convergence achieved an F1 score of 0.625, which is a huge increase from single deep learning models. Beside its high performance, the stacking model requires less time for training compared with single deep learning models (We don't have statistics to support this point of view because we didn't save the log data that contains computation time before Alibaba released our instance). The shortcomings of the stacking model include that it is not an end-to-end model and there are more hyperparameters that need to be considered, such as how the proportion of data that is assigned to train each single model.

The main limit of our model is that it can only detect a binary relation within a sentence. Some of the relations which exists across sentences are lost during the procedure of preprocessing

described in 2.2.3. Song et al. [6] proposed graph-state LSTM model to capture relations among n entities across multiple sentences. In the future, we will try to implement such types of networks and attempt to propose some modification.

Besides, a potential issue, which is not mentioned in Zeng's paper [7], has been found in PCNN during implementation. If one entity locates at the starting position, or locates at the ending position, or is adjacent to the other, the sentence is divided into two parts instead of three (see Figure 2.5). Our solution is simply filling a zero to the squeezed segment. More reasonable results are worth to be explored.

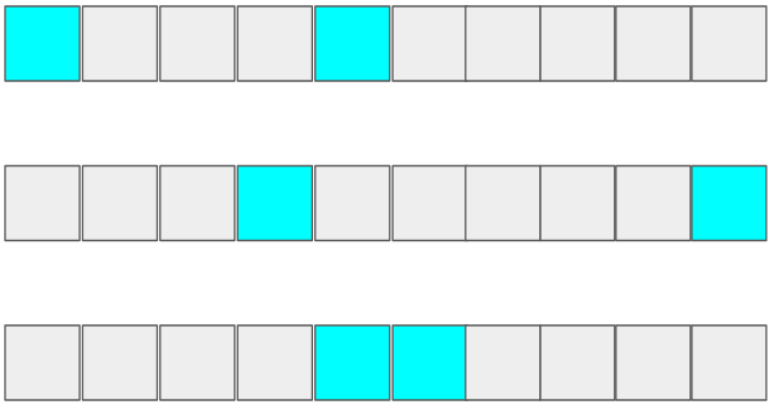

Figure 2.5: Special cases in PCNN, where a sentence is divided to two segments

Work Breakdown

Table 2.2: Work Breakdown

Work	Person
NER-Data Exploration and Visualization	Xiaotian Shuoxuan
NER-Feature Enginerring	Xiaotian Shuoxuan Peilin
NER-Model Implementation	Jifan Xiaotian Peilin
NER-Parameter Tuning	Shuoxuan Jifan
NER-Word-based Model	Peilin
RE-Data Exploration and Visualization	Jifan Xiaotian Shuoxuan
RE-Feature Enginerring	Jifan Xiaotian Shuoxuan
RE-Model Implementation	Jifan Peilin
RE-Parameter Tuning	Jifan Shuoxuan
RE-Stacking Model Design	Jifan

ACKNOWLEDGEMENT

A thank you to Professor Rekatsinas for an amazing semester! It has been an exciting opportunity to learn deep about PGMs and more applications in deep learning and data management. We really appreciate your feedback about out project along the way. Have a great trip and happy holiday!

Bibliography

[1] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[5] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575, 2016.

[6] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. N-ary relation extraction using graph state lstm. *arXiv preprint arXiv:1808.09101*, 2018.

[7] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, 2015.

[8] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212, 2016.