Using ELMo to Solve the Named Entity Recognition Problem of Groningen Meaning Bank

58119213 Danying Xu
58119321 Yuchen Li

Abstract

Named Entity Recognition (NER), also known as proper name recognition, is a basic task in natural language processing and has a wide range of applications. Named entities generally refer to entities with specific meaning or strong referentiality in the text, which usually include names of persons, names of places, names of organizations, date and time, proper nouns, etc. At present, CNN/RNN-CRF, which combines neural networks and CRF models, is the current mainstream model of NER. But it cannot understand the context of forward tags; at the same time using LSTM could add additional complexity. Therefore, in our work, we aim to study the use of existing word embedding models to try to solve the problem of named entity recognition. We use the ELMo model trained on the 1 Billion Word Benchmark and found the Groningen Meaning Bank (GMB) dataset. We find that it has achieved very good results on this data set in the financial field.

1 Introduction

Named Entity Recognition (NER) is an information extraction technique, simply to find related entities from a piece of natural language text and mark their location and type. Generally, it is classified as a kind of sequence labeling problem. Compared with the classification problem, the current predicted label in the sequence labeling problem is not only related to the current input feature, but also related to the previous predicted label, that is, there is a strong interdependence between the predicted label sequences. For example, when using the BIO tag for NER, the tag O will not be followed by the tag I in the correct tag sequence.

The entities can be pre-defined and generic like location names, organizations, time and etc, or they can be very specific like the example with the resume. NER has a wide variety of use cases in the business. For example Gmail is applying NER when you are writing an email and you mention a time in your email or attaching a file, Gmail offers to set a calendar notification or remind you to attach the file in case you are sending the email without an attachment. Other applications of NER include: extracting important named entities from legal, financial, and medical documents, classifying content for news providers, improving the search algorithms, and etc.

At beginning, we want to solve the name entity recognition problems specifically in e-mail scenario. However we find it difficult to find well-labeled datasets and it is also impossible to just use private mails and label them manually. Therefore, we changed our goal into the finance scenario which is described in detail in section 4.1.

2 Related Work

In this part, we focus on mainly two parts, word embedding and NER. Section 2.1 focused on the development in the task NER and discussed several state-of-art models. Section 2.2 focused on simply embedding, and discussed several widely used embedding models.

2.1 Word Embedding

Word Embedding is to convert words into the form of vectors. Since we want the computer to process human languages, we have to transform words into vectors. The simplest way is to use One-Hot, however it failed to give out semantic understanding. The word2vec proposed by google in 2013 opened up a new world of word vectors(Mikolov et al., 2013). The disadvantage is that there is a unique embedding

representation for each word, and for polysemous words, this approach is obviously not intuitive, and the meaning of the word is context-dependent.

Therefore we consider using ELMo as embedding model(Peters et al., 2018). It is a new type of deep contextualized word representations, which simulates the complex features of word usage (for example, syntax and semantics), and how these usages change in the language context (ie , Modeling the ambiguity). The ELMo approach is that we only pre-train the language model, and output the input sentence in real time through word embedding, so that the meaning of the word is context-sensitive, which greatly reduces the occurrence of ambiguity. In addition, ELMo outputs multi-layer embedding. In the experiment, it is found that the information output by each layer of ELMo has different effects on different tasks. Therefore, using a different embedding layer for each token will improve the effect.

ELMo has three important representations:

1. Context: The expression of each word depends on the entire context in which it is used.

2. Depth: Word expression combines all layers of a deep pre-trained neural network.

3. Character-based: ELMo representation is purely character-based, allowing the network to use morphological cues to form a robust representation of vocabulary that cannot be seen in training.

2.2 Name Entity Recognition

The development of NER is originated from methods from dictionaries and rules, which is then solved through traditional machine learning and deep learning. And in recent years, attention models, transfer learning and semi-supervised learning are widely used. There are mainly two ways in traditional machine learning methods: treat the problem as a multi-class classification or Conditional Random Field (CRF) model.

When treating the problem as a multi-class classification where named entities are our labels, we can apply different classification algorithms. The problem here is that identifying and labeling named entities require thorough understanding of the context of a sentence and sequence of the word labels in it, which this method ignores that.

When it comes to CRF(a probabilistic graphical model that can be used to model sequential data such as labels of words in a sentence), the CRF model is able to capture the features of the current and previous labels in a sequence but it cannot understand the context of the forward labels. This shortcoming plus the extra feature engineering involved with training a CRF model, makes it less appealing to be adapted by the industry. Other previous studies used models like bidirectional LSTM-CRF (Huang et al., 2015), bidirectional LSTM-CNNs (Chiu and Nichols, 2016) and bidirectional LSTM-CNNs-CRF(Ma and Hovy, 2016) LSTM to improve performances of sequence labeling models, yet if the task that needs to be identified does not need to rely too much on long-term information, at this time models using LSTM could add additional complexity.

3 Methods

This part is divided into 3 parts:Bidirectional language models(section 3.1), ELMo to complete word embeddings(section3.2) and do the name entity recognition prediction(section 3.3).

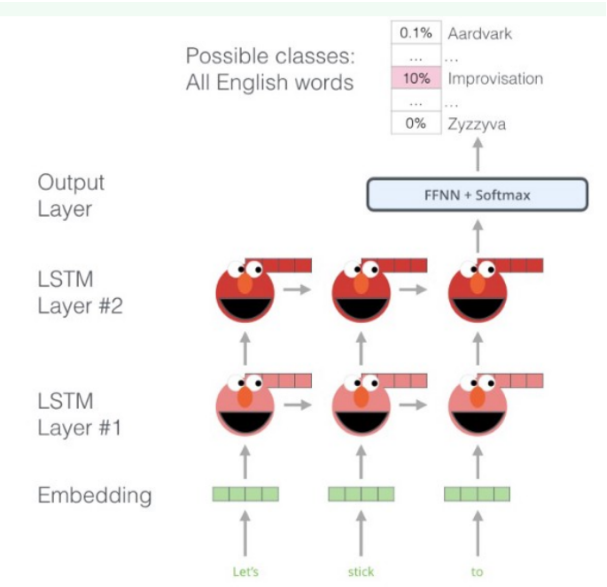


Figure 1: ELMo Framework

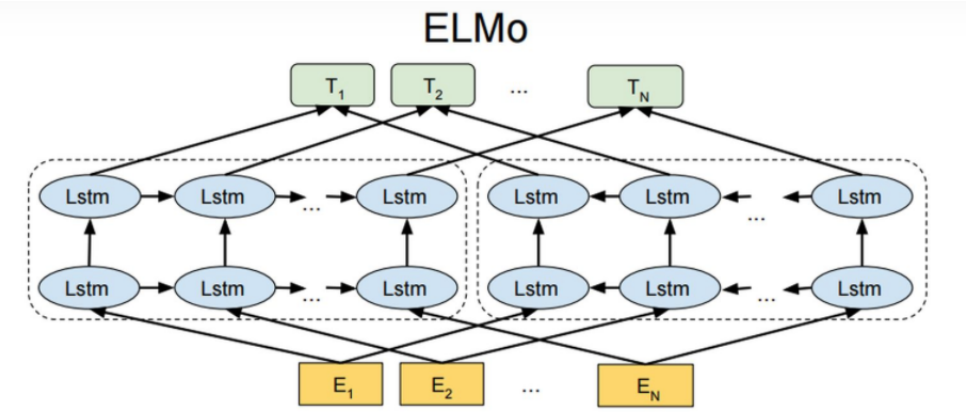


Figure 2: The details of ELMo

3.1 Bidirectional language models

Given a sequence of N tokens, (t_1, t_2, ..., t_N), a forward language model computes the probability of the sequence by modeling the probability of token t_k given the history (t_1, ..., t_k−1):

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k)|(t_1, t_2, ..., t_{k-1}) \tag{1}$$

Recent state-of-the-art neural language models (Jozefowicz et al., 2016)(Melis et al., 2017) (Merity et al., 2017) compute a context-independent token representation $\mathbf{x}_k^{LM}$ (via token embeddings or a CNN over characters) then pass it through L layers of forward LSTMs. At each position k, each LSTM layer outputs a context-dependent representation $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$ where j = 1, ...,L. The top layer LSTM output, $\overrightarrow{\mathbf{h}}_{k,L}^{LM}$, is used to predict the next token tk+1 with a Softmax layer.

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k)|(t_{k+1}, t_{k+2}, ..., t_N) \tag{2}$$

It can be implemented in an analogous way to a forward LM, with each backward LSTM layer j in a L layer deep model producing representations $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ of t_k given (t_k+1, . . . , t_N).

A biLM combines both a forward and backward LM. Our formulation jointly maximizes the log likelihood of the forward and backward directions:

$$\begin{aligned}\prod_{k=1}^{N} (&logp(t_k|t_1, ..., t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) \\ &+logp(t_k|t_{k+1}, ..., t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s))\end{aligned} \tag{3}$$

We tie the parameters for both the token representation (Θ_x) and Softmax layer (Θ_s) in the forward and backward direction while maintaining separate parameters for the LSTMs in each direction. Overall, this formulation is similar to the approach of (Peters et al., 2017), with the exception that we share some weights between directions instead of using completely independent parameters. In the next section, we depart from previous work by introducing a new approach for learning word representations that are a linear combination of the biLM layers.

Figure 3 shows the model realized in our work.

3.2 ELMo

The ELMo network is computed on top of two-layer biLMs with character convolutions, as a linear function of the internal network states. This setup allows us to do semi-supervised learning, where the biLM is pre-trained at a large scale and easily incorporated into a wide range of existing neural NLP architectures. Figure 1 and 2 show the architecture of the model.

For each token t_k, a L-layer biLM computes a set of 2L + 1 representations

$$\begin{aligned}R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, ..., L\} \\ &= \{\mathbf{h}_{k,j}^{LM} | j = 0, ..., L\}\end{aligned} \tag{4}$$

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 50)	0	
lambda_1 (Lambda)	(None, 50, 1024)	0	input_1[0][0]
bidirectional_1 (Bidirectional)	(None, 50, 1024)	6295552	lambda_1[0][0]
bidirectional_2 (Bidirectional)	(None, 50, 1024)	6295552	bidirectional_1[0][0]
add_1 (Add)	(None, 50, 1024)	0	bidirectional_1[0][0] bidirectional_2[0][0]
time_distributed_1 (TimeDistrib	(None, 50, 17)	17425	add_1[0][0]

Total params: 12,608,529
Trainable params: 12,608,529
Non-trainable params: 0

Figure 3: The Model of Network (using "model.summary()")

where $\mathbf{h}_{k,j}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

For inclusion in a downstream model, ELMo collapses all layers in R into a single vector, ELMo_k = E(R_k; Θ_e). In the simplest case, ELMo just selects the top layer, E(R_k) $=\mathbf{h}_{k,j}^{LM}$, as in TagLM (Peters et al., 2017) and CoVe (McCann et al., 2017). More generally, we compute a task specific weighting of all biLM layers:

$$\begin{aligned} ELMo_k^{task} &= E(R_k; \Theta^{task}) \\ &= \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM} \end{aligned} \tag{5}$$

In (5), s^{task} are softmax-normalized weights and the scalar parameter γ task allows the task model to scale the entire ELMo vector. γ is of practical importance to aid the optimization process. Considering that the activations of each biLM layer have a different distribution, in some cases it also helped to apply layer normalization(Ba et al., 2016) to each biLM layer before weighting.

3.3 Name Entity Recognition

The name entity recognition follows the most common steps by simply using the fine-tuned embedding to accomplished the predictions. To be specific, during training, we should first collect a set of representative training documents, then label each token for its entity class or other (O), and design feature extractors appropriate to the text and classes. Finally train a sequence classifier to predict the labels from the data. During testing or classifying, first to receive a set of testing documents,then run sequence model inference to label each token, and finally appropriately output the recognized entities.

In our experiment, we will map labels to different numbers and train our bi-LSTM model by optimizing the loss function. In other words, we can achieve the final annotation of the NER task through the neural network.

4 Experiments

The experiments are completed by Jupyter notebook and Pycharm2018 for implementation of state-of-the-art Named Entity Recognition with bidirectional LSTMs and ELMo mainly uses TensorFlow and Keras.

4.1 Environment

The experiment was done on the professional software of PyCharm 2018.1.4 x64. Due to the need to use TensorFlow's deep learning network architecture, the experiment was completed on the laboratory server of Associate Professor Wang Beilun. TensorFlow uses version 1.15.0, and strictly matches python3.6.13, CUDA10.0, CUDNN7.4, Keras2.3.1 and tensorflow_hub0.12.0 (the process of configuring the environment is too difficult). The project code is submitted as an attachment to the report, and the specific implementation can be directly compiled through the server.

4.2 Dataset

We use the existing Kaggle dataset as the input feeding to the embedding model. It is the extract from GMB corpus which is tagged, annotated and built specifically to train the classifier to predict named entities such as name, location, etc. After calculating, we find it has total 47959 sentences, 35179 different words and 17 labels. And by using histogram, we find the longgest sentence has 104 words but almost every sentence has less than 60 words. Therefore, since the length for ELMo input needs to be precisely equal, we set the length of all sentence to 50 words. For those sentences less than 50 words, we use 'PADword' to fulfill the blanks.

longgest sentence has 104 words

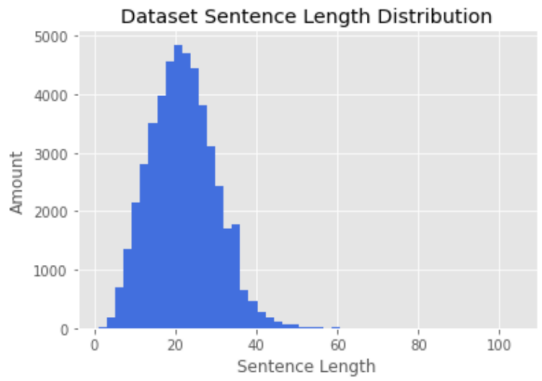


Figure 4: The Distribution of Sentence Length in the Dataset

To make it more convenient for the model to do embedding, we map all the labels to numbers. And we split the dataset into training set(including validation set), and testing set(test_size= 0.1).

4.3 Model

The overall idea of the implementation process is: First, we convert each sentence with a named entity (marker) into a list of tuples [(word, named entity),...]. Because ELMo does not require any feature engineering. We only need to put sentences of the same size and their labeled words into the ELMo meter. We can set the length of all sentences to 50, and fill the blank space with the same word. Below, we map the labels to numbers, perform data segmentation (training and test sets), and then convert the sentences into ELMo embeddings and optimize them through neural networks. Finally, we output some indicators for model evaluation and some prediction results for actual comparison.

4.4 Name Entity Recognition

ELMo is a model embedded from a language model trained on the 1 Billion Word Benchmark. The module outputs fixed embeddings at each LSTM layer, a learnable aggregation of the 3 layers, and a fixed mean-pooled vector representation of the input.

We build the neural network referring to the ELMo paper. We first get the existing ElMo model and then construct the two bidirectional LSTM networks, with each of them has 512 units and dropout=0.2. These two bidirectional LSTM networks are connected together and finally output with same dimensions as label numbers.

The model is optimized with adam and uses cross-entropy loss function. Figure 5 shows the process of our training.

```
Train on 38816 samples, validate on 4320 samples
Epoch 1/3
2021-12-25 15:30:12.693266: I tensorflow/stream_executor/platform/default/dso_
so.10.0
2021-12-25 15:30:13.041085: I tensorflow/stream_executor/platform/default/dso_
o.7
38816/38816 [==============================] - 499s 13ms/step - loss: 0.0640 -
Epoch 2/3
38816/38816 [==============================] - 502s 13ms/step - loss: 0.0415 -
Epoch 3/3
38816/38816 [==============================] - 531s 14ms/step - loss: 0.0346 -
4768/4768 [==============================] - 47s 10ms/step
```

Figure 5: The Training Process

Using ELMo as the embedding model to do the NER task could reach an precision 0.81 under 3 epochs and batch_size=32. Figure 6 shows several results and evaluations after using ELMo as embedding model.

F1-score: 81.5%

	precision	recall	f1-score	support
art	0.42	0.10	0.16	49
eve	0.35	0.33	0.34	33
geo	0.84	0.89	0.86	3720
gpe	0.95	0.94	0.94	1591
nat	0.22	0.45	0.30	22
org	0.67	0.67	0.67	2061
per	0.71	0.80	0.75	1677
tim	0.87	0.83	0.85	2148
micro avg	0.81	0.82	0.81	11301
macro avg	0.63	0.63	0.61	11301
weighted avg	0.81	0.82	0.81	11301

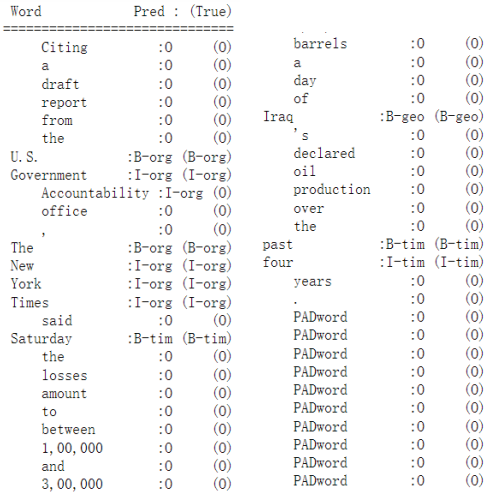
Figure 6: Few evaluations of the model

Then, after the embedding is proved to be available, we use it to make name entity recognition. Figure 7 shows the comparison between prediction and true labels over few words.

4.5 Problems We Met

During this experiment, the main problems we encountered are as follows:

- Regarding environment configuration: TensorFlow's environment configuration



```
Word            Pred : (True)
==============================
     Citing          :0      (0)
     a               :0      (0)
     draft           :0      (0)
     report          :0      (0)
     from            :0      (0)
     the             :0      (0)
U.S.             :B-org (B-org)
Government       :I-org (I-org)
     Accountability :I-org (0)
     office          :0      (0)
     ,               :0      (0)
The              :B-org (B-org)
New              :I-org (I-org)
York             :I-org (I-org)
Times            :I-org (I-org)
     said            :0      (0)
Saturday         :B-tim (B-tim)
     the             :0      (0)
     losses          :0      (0)
     amount          :0      (0)
     to              :0      (0)
     between         :0      (0)
     1,00,000        :0      (0)
     and             :0      (0)
     3,00,000        :0      (0)
     barrels         :0      (0)
     a               :0      (0)
     day             :0      (0)
     of              :0      (0)
Iraq             :B-geo (B-geo)
     's              :0      (0)
     declared        :0      (0)
     oil             :0      (0)
     production      :0      (0)
     over            :0      (0)
     the             :0      (0)
past             :B-tim (B-tim)
four             :I-tim (I-tim)
     years           :0      (0)
     .               :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
     PADword         :0      (0)
```

Figure 7: Some Predicting Results

is too cumbersome, and there are strict corresponding requirements for the version of each library, otherwise there will be a compilation mismatch, which consumes a lot of time.

- Downloading the pre-trained model from the web site on the server may not be accessible. The two solutions are: download the pre-trained model locally and upload it directly to the server for local transfer; let the machine allow the local area network to join, and The server is configured to transmit data through the local network. We finally adopted the first method.

- In the process of implementing the network architecture, I was not proficient in the use of TensorFlow and conducted many discussions and learning.

5 Conclusion

We first introduced the definition and situation of named entity recognition, and explained our experimental tasks. Then, we introduced the overall development of existing work and the latest research results from two aspects of word embedding and named entity recognition, and explained why we used the ELMo model for experiments. ELMo has a good understanding of language because it is trained on a huge data set, and ELMo embeddings are trained on a benchmark of 1 billion words. This kind of training is called a bidirectional language model (biLM), which can pass from the past and predict the next word according to a sequence of words (such as a sentence). In the method part, we give the definition of the ELMo model implementation and the logic formula for the implementation, and explain the flow of our NER task. In the experimental part, we respectively elaborated and explained from the data set and code results, and showed some of our experimental results. It finally showed good results.

Through this experiment, we have a deeper understanding of the course of natural language processing. We fused the knowledge of the two parts of word vector and named entity recognition, and learned the implementation process of the ELMo model. In the experiment, although the process of configuring the environment and finding relevant materials and understanding is relatively complicated, we have learned a lot of experience in deep learning implementation, which makes us feel that we have gained something.

Acknowledgement

Thanks to teacher Wang Meng for the interesting courses this semester, which allowed us to learn a lot of new knowledge. Thanks to Associate Professor Wang Beilun for allowing us to complete the experiment on the laboratory server. Thanks to the teaching assistant for her homework correction and help.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics, 4:357–370.

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

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. arXiv preprint arXiv:1708.00107.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. arXiv preprint arXiv:1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. arXiv preprint arXiv:1708.02182.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. arXiv preprint arXiv:1705.00108.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.