Research on Extraction of Address from Criminal Case Records via Lattice LSTM

To cite this article: Xiangwu Ding and Ziqi Yang 2019 J. Phys.: Conf. Ser. 1325 012082

View the article online for updates and enhancements.

You may also like

- Legal risk and criminal imputation of weak artificial intelligence
  Lei Wei

- Environmental legislation of criminal responsibility of international comparison and reference ——Based on the Perspective of Law and Economic
  L N Fa

- Diversion approach as an alternative case resolution son of the conflict with the law on child criminal justice system (case study of Merauke)
  M J Alputila, M A Tajuddin and Kahar
Research on Extraction of Address from Criminal Case Records via Lattice LSTM

Xiangwu Ding\textsuperscript{1} and Ziqi Yang\textsuperscript{1}\textsuperscript{*}

\textsuperscript{1} School of Computer Science and Technology, Donghua University, Shanghai, 201620, China

*ziqiyang522@gmail.com

Abstract. As an important carrier for recording the details of case in the field of criminal investigation, case records contain address of the crime scene which is crucial to the case analysis. This paper uses Lattice LSTM to extract address from case records which combines the advantages of the character-based and word-based models. It can exploit word and word sequence information without performing Chinese word segmentation. For the purpose of optimizing the model, this paper proposes a word vector template which is generated by address repository of Shanghai to pretrain more accurate and more controllable word embeddings. The experiment shows that the model achieves the performance with the precision rate of 96.49\%, recall rate of 94.83\% and F1 score of 95.65\%.

1. Introduction

Along with the implementation of the informationization construction in the public security department in China, a large number of case records have been collected electronically. Case record, as an unstructured text, is a brief statement of the case including the personal description of the victim and descriptive information collected by the police. Among all the details described in the case records, address is one of the most important features. This paper focuses on address extraction.

2. Related Work

Extracting features from text can be regarded as named entity recognition (NER) in the field of natural language processing (NLP). Rule-based approach and statistics-based approach are all traditional methods\cite{1}. Rule-based approach needs man-made rules and domain dictionary which is expensive to design and build. The coverage rate of the rules and the completeness of dictionary have huge impact on the model. HMM and CRF are two typical statistics-based approaches\cite{2-3}. Compared to the rule-based approaches, statistics-based approaches can learn sufficient features from labeled training dataset. However, a series of conditionally independent probabilistic assumptions cannot always be statified. Nowadays, NER with deep learning is state-of-the-art. This paper regards NER as the sequence labeling problem and solve it with Recurrent Neural Network (RNN). It is apparent that the words sequence is crucial. As the result, bi-directional RNN model performs better. The structure of LSTM cell is more complex than normal RNN one which can remember long distance information\cite{4-6}. Adding CRF layer can learn more rationality of the label sequence\cite{7}. Bi-LSTM-CRF is state-of-the-art for English NER. Because of its superior adaptability, it can also be applied to Chinese NER\cite{8-11}. However, word-based Bi-LSTM-CRF may suffer wrong word segmentation because space-separated word tokens do not exist in Chinese. It has been shown that character-based methods perform better than word-based methods for Chinese NER. One disadvantage of character-based methods is that important information of words...
sequence has been lost[10]. Zhang et al. proposed a Lattice LSTM which has a well-designed architecture can integrate words sequence information into character-based Bi-LSTM-CRF[12].

3. Methods

3.1. Labeling Sequence
Formally, \( \text{case}^w = \langle w_1, w_2, ..., w_n \rangle \) denotes a case record which consists of \( n \) words, \( w_i \) denotes the \( i \)-th word which appears in \( \text{case}^w \). One case record can also be denoted by \( m \) characters, which can be written as \( \text{case}^c = \langle c_1, c_2, ..., c_m \rangle \), \( c_i \) denotes the \( i \)-th character which appears in \( \text{case}^c \). This paper adopts BMEOS(Begin, Middle, End, Other, Single) tagging schema to label the character sequence.

![Figure 1. Lattice LSTM structure(only shown in one direction)](image)

3.2. Lattice LSTM
Lattice LSTM is an extension of Bi-LSTM-CRF. Special cells are added into the network which span the main framework of the network. We call them word cell because they send word message to the character-based cell. By the help of word cells, Lattice LSTM can integrate crucial information of words sequence into character-based Bi-LSTM-CRF. The structure of Lattice LSTM is shown in Figure 1.

Essentially, Lattice LSTM is a character-based Bi-LSTM-CRF model so it is unnecessary to do Chinese word segmentation. Every character \( c_j \) from the case record \( \text{case}^c \) is fed into the network. And in the first layer, \( c_j \) would be transformed to a word vector \( x_j^w = e^w(c_j) \), \( e^c \) denotes a pre-trained character embedding lookup table. Bi-directional structure network gives us two kinds of output. Feeding characters sequence \( c_1, c_2, ..., c_m \), then we obtain two output sequences \( \vec{h}^c_1, \vec{h}^c_2, ..., \vec{h}^c_m \) and \( \vec{h}^c_{-1}, \vec{h}^c_{-2}, ..., \vec{h}^c_{-m} \). The former is left to right sequence and the latter is right to left sequence. Finally, we concatenate them as the end output \( \vec{h}^w_f = [\vec{h}^c_f; \vec{h}^c_{-f}] \).

Lattice LSTM adds a kind of special word cells which take word as their input. Before a character is fed into the network, Lattice LSTM would check if the current character and the former characters can be concatenated together to obtain a word. Formally, we denote a word dictionary \( \mathbb{D} \). \( w_{b,e} \in \mathbb{D} \) denotes that characters sequence \( c_b, c_{b+1}, ..., c_e \) can make up a word \( w_{b,e} \) belongs to \( \mathbb{D} \).

\( w_{b,e} \) is transformed into word vector \( x_{b,e}^w \), then \( x_{b,e}^w = e^w(w_{b,e}) \) would be fed into the word cell \( \text{cell}_{b,e}^w \). \( e^w \) denotes a pre-trained word embedding lookup table. Just as the characters sequence, we put words sequence \( w_1, w_2, ..., w_n \) and then obtain bi-directional outputs \( \vec{h}^w_1, \vec{h}^w_2, ..., \vec{h}^w_n \) and \( \vec{h}^w_{-1}, \vec{h}^w_{-2}, ..., \vec{h}^w_{-n} \) respectively, which are concatenated as the final output \( \vec{h}^w_f = [\vec{h}^c_f; \vec{h}^c_{-f}] \).
Figure 2. The structure of word cell

The structure of word cell is shown in Figure 2. Word cell $cell_{b,e}^w$ span between character-based cell $cell_{b}^c$ and $cell_{e}^c$. Every word cell $cell_{b,e}^w$ takes word vector $x_{b,e}^w$ as its input. It also receives the information $c_{b}^c$ and $h_{b}^c$ outputed by $cell_{b}^c$. It is worth mentioning that word cells do not have output gate because they are not engaged in predicting the labels sequence. The output $c_{b,e}^w$ of word cell $cell_{b,e}^w$ will be transmitted to $cell_{e}^c$. By this way, the words sequence information is integrated into character-based LSTM cells. The functions of word cell are:

$$i_{b,e}^w = \sigma (w_{i}^w [h_{b}^c, x_{b,e}^w] + b_{i}^w)$$
$$f_{b,e}^w = \sigma (w_{f}^w [h_{b}^c, x_{b,e}^w] + b_{f}^w)$$
$$c_{b,e}^w = \tanh (w_{c}^w [h_{b}^c, x_{b,e}^w] + b_{c}^w)$$
$$c_{b,e}^w = f_{b,e}^w \times c_{b}^c + i_{b,e}^w \times c_{e}^w$$

$i_{b,e}^w$ and $f_{b,e}^w$ denote as input gate and forget gate, respectively.

Figure 3. character-based Lattice LSTM cell structure

The structure of character-based Lattice LSTM cell is different from the normal LSTM cell in order to receive implicit words sequence information, which is shown in Figure 3. $cell_{b,l}^f$ denotes current input character. Word cells $cell_{b,l}^f (b \in \{b' | w_{b,l}^d \in \mathbb{D}\})$ send word information $c_{b,l}^w$ that ends with $c_{f}^c$ to the current character-based cell $cell_{l}^c$. At the same time, $cell_{l}^c$ receives character vector $x_{l}^c$ as input. An additional gate $i_{b,l}^c$ is added into the cell to control the contribution of every implicit word information to the $cell_{l}^c$. It is worth mentioning that forget gate is removed from the cell because the additional gate can play the same effect as the forget gate. The functions of charcater-based cell are:

$$i_{b,l}^c = \sigma (w_{i,d}^c [c_{b,l}^d, x_{l}^c] + b_{i,d}^c)$$
$$f_{l}^c = \sigma (w_{f}^c [h_{l}^c, x_{l}^c] + b_{f}^c)$$
$$o_{l}^c = \sigma (w_{o}^c [h_{l}^c, x_{l}^c] + b_{o}^c)$$
$$c_{l}^c = \sum_{b \in \{b' | w_{b,l}^d \in \mathbb{D}\}} \alpha_{b,l}^c \times c_{b,l}^w + \alpha_{l}^c \times c_{l}^c$$
$$h_{l}^c = o_{l}^c \times \tanh (c_{l}^c)$$

$i_{l}^c$ and $o_{l}^c$ denote input gate and output gate, respectively. $w_{i,d}^c, w_{f}^c, w_{o}^c, b_{i,d}^c, b_{f}^c, b_{o}^c$ and $b_{l}^c$
are parameters. The values of additional gate \( i_{b,j} \) and input gate \( i_{f} \) are normalized to \( \alpha_{b,j} \) and \( \alpha_{f} \). The normalized functions are:

\[
\alpha_{b,j} = \frac{\exp(i_{b,j})}{\exp(i_{f}) + \sum_{b' \in \{b''|w_{b''}^{i_{b'}j} \in D\}} \exp(i_{b'j})}
\]

\[
\alpha_{f} = \frac{\exp(i_{f})}{\exp(i_{f}) + \sum_{b' \in \{b''|w_{b''}^{i_{b'}j} \in D\}} \exp(i_{b'j})}
\]

### 3.3. CRF Layer And Loss Function

CRF layer learns more knowledge about constrains and validations of the predicted labels. The probability formula of labels sequence \( label^c = \{l_1, l_2, ..., l_m\} \) corresponding to one specific input characters sequence \( case^c = \{c_1, c_2, ..., c_m\} \) is:

\[
P(label^c|case^c) = \frac{\exp(\sum_{i=1}^{m}(E(l_i) + L(l_{i-1}, l_i)))}{\sum_{label^c} \exp(\sum_{i=1}^{m}(W_{CRF}^{l_i}h_i + b_{CRF}^{l_{i-1}, l_i}))}
\]

\( E() \) is the emission score denoting that the probability of \( c_i \) labeled as \( l_i \). \( L() \) is the transition score denoting that how likely the label \( l_i \) turns into \( l_{i-1} \). \( label^c \) denotes all the possible labels sequence. \( W_{CRF} \) and \( b_{CRF}^{l_{i-1}, l_i} \) are parameters of the CRF layer. The former is specific to the label \( l_i \) and the latter is a bias of \( l_{i-1} \) and \( l_i \).

As a result, the right labels sequence ought to get the best score among all the possible ones. \( Score_{RealPath} \) denotes the score of the right labels sequence. \( \sum Score_{AllPath} \) denotes the sum score of all the possible labels sequences. Our aim is to maximize the value \( \frac{Score_{RealPath}}{\sum Score_{AllPath}} \). The training dataset is denoted as \( \{\{case^c, label^c\}| i = 1, 2, ..., N\} \). According to the probability model, the loss function is:

\[
L = \sum_{i=1}^{N} log(P(label^i|case^i)) + \frac{\lambda}{2} ||\Theta||^2
\]

\( L_2 \) regularization is used to avoid over-fitting problem. \( \lambda \) is the \( L_2 \) regularization parameter and \( \Theta \) is parameters of the model. After training the model, Viterbi algorithm is used to obtain the highest score labels sequence as the final prediction.

### 4. Experiment

**Algorithm 1. Address Extraction by Using Lattice LSTM Along With Word Vector Template**

**Require:** Dictionary D: address dictionary of Shanghai  
Lattice LSTM parameters \( \Theta \): parameter vector of the model

1: template = \( \emptyset \)
2: for each entity,tag in D:
3:     line = <tag>entity</tag> + \`
`4:     template += line
5: end for
6: use Skip-Gram algorithm to train template and obtain word vector looking up table \( W^{50} \)
7: Initialize \( \Theta \) randomly
8: while \( \Theta \) not converged do:
9:     for each character c in training case record text:
10:         find character vector \( e^{50} \) in character vector looking up table \( C^{50} \)
11:         if \( e^{50} == null \) then:
12:             initialize character vector \( e^{50} \) randomly
13:     end if
14: if \( c \) is the last character of a word \( w \) contained in \( W^{50} \) then:
15:     find word vector \( w^{50} \) in word vector looking up table \( W^{50} \)
16: end if
17: end for
18: use stochastic gradient descent algorithm to optimize \( \Theta \)
19: end while
20: return \( \Theta \)

4.1. Dataset and Evaluation Metrics
We obtain 6k case records provided by Shanghai Criminal Investigation Headquarters. We also collect 12.2k delivery addresses of Shanghai. In order to pre-train word vectors, we adopt Skip-gram algorithm. Word embeddings are 50-dimentional. Character vectors are pre-trained on the CCL Corpus of Chinese Texts including 700 million Chinese characters(http://ccl.pku.edu.cn:8080/ccl_corpus). We shuffle case records which is arranged in chronological order originally. For labeling task using the character-based method, case records are converted into the format that every line only has one character. We obtain 550k lines and split them with the rate of 60%, 20% and 20% in order to get 330k train data, 110k cross-validation data and 110k test data sets.We adopt BMEOS tag schema to label the address. We take the character-based Bi-LSTM-CRF as our baseline and evaluate model with precision rate, recall rate and F1 score.

4.2. Experiment Steps and Details
It is unnecessary to do Chinese word segmentation when using Lattice LSTM for Chinese NER. However, the accuracy and quality of the pre-trained word vectors can affect the final result of the model. This papers follow word2vec proposed by Mikolov et al.(2013) as the representation of word vector[13]. Shanghai address dictionary is used to generate the pre-trained word vectors. Therefore, we proposed a word vector training template, as shown in the following.

\[
\text{<tag> entity </tag>}
\]

We use Skip-Gram algorithm to train word embeddings and obtain the lookup table. Algorithm flowchart is shown as Algorithm 1. When training word vectors using Skip-gram, we set the window size to 1 and the word embedding size to 50. When we use Lattice LSTM to extract address, we set LSTM layer to 1, hidden size of LSTM models to 128 and the dropout rate to 0.2. We use Stochastic gradient descent(SGD) for optimization and set the initial learning rate to 0.02 with a decay rate of 0.05. The number of iterations is 100. It is worth mentioning that instead of normalizing all the Arabic numerals to 0, we retain all the original number in the text.

4.3. Results and Discussion
Table 1 shows the results of various models performing on our dataset. Lattice LSTM with the help of word vector templates achieves the best performance at the precision rate of 96.49%, recall rate of 94.83% and F1 score of 95.65%. Our model is 5.21%, 4.21% and 4.70% higher than the baseline character-based Bi-LSTM-CRF. By comparing the last two rows of the Table 1, we can savely draw a conclusion that word vector templates can generate more accuracy and more controllable word vectors so that the Lattice LSTM can learn more about the words information.

| Model                     | P     | R     | F1   |
|---------------------------|-------|-------|------|
| character-based Bi-LSTM-CRF | 91.28 | 90.62 | 90.95|
| Lattice LSTM              | 94.74 | 93.10 | 93.91|
| Lattice LSTM + word vector templates | 96.49 | 94.83 | 95.65|
5. Conclusions
This paper focuses on the address extractions of case records. The task can be regarded as sequence labeling problem. Lattice LSTM is used to extract address from the text which is based on character so it can avoid the Chinese word segmentation. Besides, Lattice LSTM can integrate important words sequence information into the character-based LSTM cell by adding a special kind of cells called word cell. This paper also proposes a word vector training template which can generate more accurate and more controllable word vectors. The experiment shows that the model achieves the best performance at the precision rate of 96.49%, recall rate of 94.83% and F1 score of 95.65%.

Acknowledgments
We would like to express our gratitude to the anonymous reviewers for their constructive comments to this paper.

References
[1] N. Tashpolat, K. Wang, H. Askar and T. Palidan (2017) Combination of Statistical and Rule-based Approaches for Uyghur Person Name Recognition. Zidonghua Xuebao/Acta Automatica Sinica, 43: 653-664.
[2] Y. Feng, L. Sun and J. Zhang (2005) Early Results for Chinese Named Entity Recognition Using Conditional Random Fields Model, HMM and Maximum Entropy. In: Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering. Wuhan. pp. 549-552.
[3] S. Song, N. Zhang and H. Huang (2017) Named Entity Recognition Based on Conditional Random Fields. In: Cluster Computing. Honolulu. pp. 1-12.
[4] K. Cho, B. v. Merrienboer, D. Bahdanau, Y. Bengio (2014) On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, arXiv: 1409.1259.
[5] J. Chung, C. Gulcehre, K. Cho and Y. Bengio (2014) Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS 2014 Deep Learning and Representation Learning Workshop, arXiv: 1412.3555.
[6] S. Hochreiter and S. Jürgen (1997) Long short-term memory. Neural Computation, 9(8): 1735-1780.
[7] L. Guillaume, B. Miguel, S. Sandeep, K. Kazuya and D. Chris (2016) Neural Architectures for Named Entity Recognition. In: 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference. Berlin. pp. 260-270.
[8] J. Chiu and E. Nichols (2015) Named Entity Recognition with Bidirectional LSTM-CNNs. Transactions of the Association for Computational Linguistics, arxiv: 1511.08308.
[9] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. CoRR (2016) Neural Architectures for Named Entity Recognition. CoRR, abs: 1603.01360.
[10] C. Dong, J. Zhang, C. Zong, M. Hattori and H. Di (2016) Character-Based LSTM-CRF with Radical-Level Features for Chinese Named Entity Recognition. In: Computer Science. Springer. pp. 239-250.
[11] Y. Qin and Y. Zeng (2018) Research of Clinical Named Entity Recognition Based on Bi-LSTM-CRF. Journal of Shanghai Jiaotong University, 23:392-397.
[12] Y. Zhang and J. Yang (2018) Chinese NER Using Lattice LSTM. In: 56th Annual Meeting of the Association for Computational Linguistics. Melbourne. pp. 1554-1564.
[13] T. Mikolov, K. Chen, G. Corrado and J. Dean (2013) Efficient estimation of word representations in vector space. Computer Science, arXiv: 1301.3781.