Study of Information Extraction in Resume

Van Vinh Nguyen[€], Van Long Pham[*], Ngoc Sang Vu[¶]

VNU University of Engineering and Technology,
E3 Building, 144 Xuan Thuy Street, Cau Giay, Hanoi, Vietnam
[€]vinhnv@vnu.edu.vn, [*]phamvanlong021096@gmail.com, [¶]ngocsangnd259@gmail.com

Abstract

This paper deals with the parsing application developed for the resumes (or CV) received in multiple formats like doc, docx, pdf, txt. These resumes can be automatically retrieved and processed by a resume information extraction system. Extracted information such as name, phone / mobile number, e-mail id, qualification, experience, skill sets etc., can be stored as a structured information in a database and then can be used in many different areas. Our system consists of 4 phases: Text Segmentation, Name Entity Recognition using Rule-based, Find Name Entities using Deep Neural Network and Text Normalization. Our work is conducted on a medium-sized collections of CV files in Vietnamese. We archived promising results with over 81% F1 for NER and also compared our model with other systems.

Keywords: *Resume Parser, Convolutional Neural Network, Long Short-Term Memory, Conditional Random Fields.*

1. Introduction

Big enterprises and head-hunters receive hundreds of resume from job applicants every day. Automatically extracting structured information from resume of different styles and formats is needed to support the automatic construction of database, searching and resume routing. The definition of resume information fields varies in different applications. Normally, resume information is described as a hierarchical structure with two layers. The first layer is composed of consecutive general information blocks such as Personal Information, Education etc. Then within each general information block, detailed information pieces can be found, e.g., in Personal Information block, detailed information such as Name, Address, Email, etc. can be further extracted.

Extracting information from resume with high precision and recall is not an easy task. In spite of constituting a restricted domain, resume can be written in multitude of formats (e.g. structured tables or plain texts), in different languages (e.g. Vietnamese and

English) and in different file types (e.g. Text, PDF, Word etc.). Moreover, writing styles could be very diversified.

Extracting the Named Entity (or Named Entity Recognition – NER) in the resume is very necessary and beneficial. It allows employers to compare two candidates or automatically fill the candidate information into the database. Consequently, a large number of approaches have been designed for performing NER, see [16] for a survey. One simple approach to extract Named Entity in resume is to use a list of gazette, for example, a gazette for the Named Entity "Degree" would consist of all known educational degree names, such as "Bachelor of Arts", "B.A.", "PhD", etc. The gazette-based approach results in fast and high precision NER, since one simply looks for occurrences of any entries in the gazette. However, this method requires a large predefined gazette, and does not completely solve the semantic ambiguity problem. For example, "Tôn Đức Thắng" is the name, but in sentence "Đại học Tôn Đức Thắng", it is the university, and in sentence "Số 12 đường Tôn Đức Thắng", the meaning of the word is the address.

This paper deals with the Named Entity Recognition in resume data, using Rule-based and Deep Learning approach. Our Deep Learning model is combination of Convolutional Neural Networks, Bi-directional Long Short-Term Memory and Conditional Random Field, to label the sequence in resume and then extract Name Entities from this labeled sequence.

2. Related work

Recent advances in information technology such as Information Extraction (IE) provide dramatic improvements in conversion of the overflow of raw textual information into structured data which constitute the input for discovering more complex patterns in textual data collections.

Resume information extraction, also called resume parsing, enables extraction of relevant information from resumes which have relatively structured form. Although, there are many commercial products on resume information extraction, some of the commercial products include Sovren Resume/CV Parser [22], Akken Staffing, ALEX Resume parsing [1], ResumeGrabber Suite and Daxtra CVX [7]. There are four types of methods used in resume information extraction: Named-entity-based, rule-based, statistical and learning-based methods. Usually a combination of these methods is used in many applications.

- Named-entity-based information extraction methods try to identify certain words, phrases and patterns usually using regular expressions or dictionaries. This is usually used as a second step after lexical analysis of a given document [4, 21]. Rule-based information extraction is based on grammars.
- Rule-based information extraction methods include a large number of grammatical rules to extract information from a given document [21].
- Statistical information extraction methods apply numerical models to identify structures in given documents [21].
- The learning-based methods employ classification algorithms to extract information from a document.

Many Resume Named Entity extraction systems employ a hybrid approach by using a combination of different methods. However, until now, there are no systems which use Deep Learning approach and apply for Vietnamese.

3. The method

The design of Information Extraction System consists of 4 phases: Text Segmentation, Name Entity Recognition using Rule-based, Find Name Entities using Deep Neural Network and Text Normalization. In the first phase, a resume is segmented into blocks according to their information types. In the second phase, information are found by using special rules for each information type. In the third phase, we used a combination of Bi-directional Long Short-term Memory – Convolutional Neural Networks – Conditional Random Field to extract Named Entities information. In the fourth phase, normalization methods are applied to the text.

3.1. Text Segmentation

Text Segmentation phase do work on the fact that each heading in a resume contains a block of related information following it. So in that case our resume will separate out into segments named as contact information, education information, professional details and personal information segment as shown in Table 1.

Table 1. Segment containing extracted Information Types.

Segment Type	Related Info under the Segment
Contact	Name
	Phone
	Email
	Web
Education	Degree
	Program
	Institution
Experience	Position
	Company
	Date Range

A data-dictionary is used to store common headings in a resume which are definitely occurring in the resume. These headings are searched in a given resume to find segments of related information. All of the text information between the heading and the start of the next heading is accepted as a segment. One exception will possible or may occur is the first segment which contains the name of the person and generally the contact information. It is found by extracting the text between the top of the document and the first heading. For each segment there is a group of named entity recognizers, called chunkers, that works only for that segment. This improves the performance and the simplicity of the system since a certain group of chunkers only works for a given segment. Segmentation is a crucial phase. If there is an error in the segmentation phase, chunkers will run on a wrong context. This will produce unexpected results.

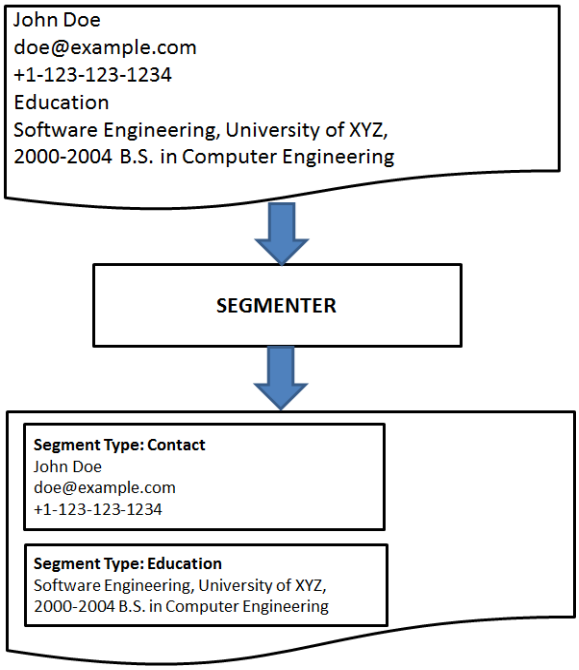

Figure 1. Segmentation in the Resume by Segmenter.

3.2. Named Entity Recognition using Rule-based

The tokenized text documents are fed to a Named Entity Recognizer using Rule-based. The term Named Entity refers to

noun phrases (type of token) within a text document that have some predefined categories like Person, Location, Organization, expressions of times, quantities, monetary values, percentages, etc. Numeric values like phone numbers and dates are also Named Entities.

Resumes consist of mostly named entities and some full sentences. Because of this nature of the resumes, the most important task is to recognize the named entities. For each type of information, there is specially designed Rule-based (also called chunker). Information types are shown in Table 1. Each Rule-based is run independently as shown in Figure 2.

Rule-based use four types of information to find named entities:

- Clue words: like prepositions (e.g. in the work experience information segment the word after "at" most probably a company name)
- Well Known or Famous names: Through data-dictionaries of well-known institutions, well known places, companies or organization, academic degrees, etc.
- From prefixes and suffixes of word: For institutions (e.g. University of, College etc.) and companies (e.g. Corp., Associates, etc.)
- Style of Writing Name of person: Generally the name of the person is written as First Letter capitalize then we will guess that this word possibly name of person.

Examples are "is-headquarter-of" between an organization and a location, "is-CEO-of" between a person and an organization, "has-phone number" between a person and phone number and "is-price-of" between a product name and a currency amount.

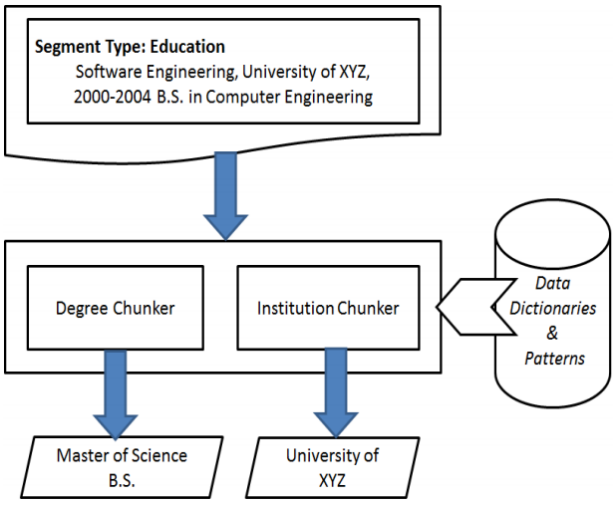

Figure 2. Chunkers extracting from Segment Education.

The chunkers produce an output that contains information about named entities as shown in Table 2.

Table 2. Found Named Entities along with their Start & End position in the resume.

Named Entity	Start	End	Type
University of XYZ	22	39	Institution
B.S.	51	54	Degree

3.3. Find Name Entity using Deep Neural Network

Although we use Rule-based, a lot of Name Entities still can't be found. To solve this, we using a Deep Neural Network (DNN), in this case DNN is the combination of Bi-directional Long Short-term Memory (LSTM) – Convolutional Neural Networks (CNNs) – Conditional Random Field (CRF), to label the sequence in resume and then extract Name Entities from this labeled sequence.

In this section, we describe the components (layers) of our neural network architecture. We introduce the neural layers

in our neural network one-by-one from bottom to top.

3.3.1. CNN for Character-level Representation

It has been shown that distributed representations of words (words embeddings) help improve the accuracy of a various natural language models. In this work, we investigate a method to create word embeddings using a CNN model (which is described in chapter 2).

Previous studies (Santos and Zadrozny, 2014 [5]; Chiu and Nichols, 2015 [9]) have shown that CNN is an effective approach to extract morphological information (like prefix or suffix of a word) from characters of words and encode it into neural representations. For this reason, we incorporate the CNN to the word-level model to get richer information from character-level word vectors. These vectors are learnt during training together with the parameters of the word models. The CNN we use in this thesis is described in figure 3.

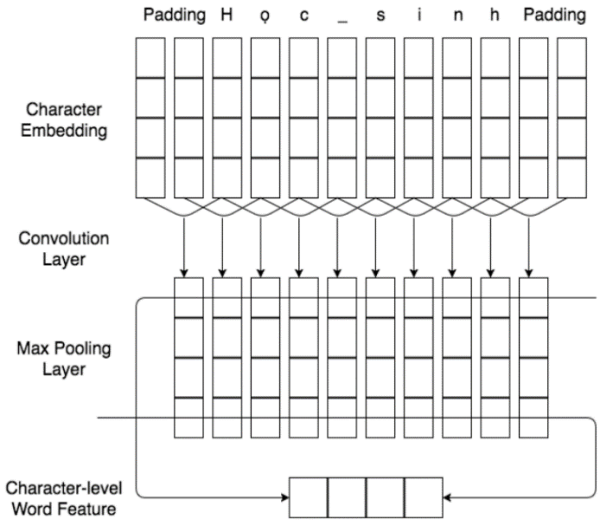

Figure 3. The CNN for extracting character-level representations of words.

3.3.2. Bi-directional LSTM

Recurrent neural networks (RNNs) are a powerful family of connectionist models that capture time dynamics via cycles in the graph. Though, in theory, RNNs are capable to capturing long-distance dependencies, in practice, they fail due to the gradient vanishing/exploding problems (Bengio et al., 1994; Pascanu et al., 2012).

LSTMs (Hochreiter and Schmidhuber, 1997) are variants of RNNs designed to cope with these gradient vanishing problems. Basically, a LSTM unit is composed of three multiplicative gates which control the proportions of information to forget and to pass on to the next time step. Figure 4 gives the basic structure of an LSTM unit.

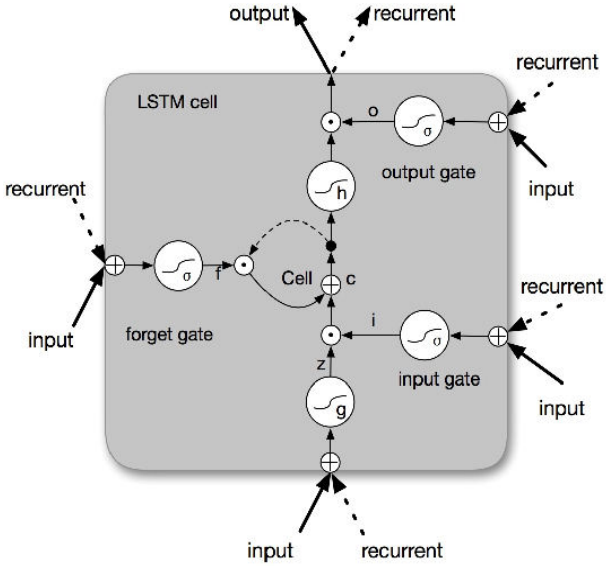

Figure 4. Schematic of LSTM unit.

Formally, the formulas to update an LSTM unit at time t are:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i\mathbf{h}_{t-1} + \boldsymbol{U}_i\mathbf{x}_t + \boldsymbol{b}_i) \\
\mathbf{f}_t &= \sigma(\boldsymbol{W}_f\mathbf{h}_{t-1} + \boldsymbol{U}_f\mathbf{x}_t + \boldsymbol{b}_f) \\
\tilde{\mathbf{c}}_t &= \tanh(\boldsymbol{W}_c\mathbf{h}_{t-1} + \boldsymbol{U}_c\mathbf{x}_t + \boldsymbol{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o\mathbf{h}_{t-1} + \boldsymbol{U}_o\mathbf{x}_t + \boldsymbol{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
$$

where σ is the element-wise sigmoid function and Θ is the element-wise product.

x_t is the input vector (e.g. word embedding) at time t, and h_t is the hidden state (also called output) vector storing all the useful information at (and before) time t. U_i , U_f , U_c, U_o denote the weight matrices of different gates for input x_t , and W_i ,W_f ,W_c,W_o are the weight matrices for hidden state h_t. b_i , b_f , b_c, b_o denote the bias vectors. It should be noted that we do not include peephole connections (Gers et al., 2003) in the our LSTM formulation.

For many sequence labeling task it is beneficial to have access to both past (left) and future (right) contexts. However, the LSTM's hidden state h_t takes information only from past, knowing nothing about the future. An elegant solution whose effectiveness has been proven by previous work (Dyer et al., 2015 [6]) is Bi-directional LSTM (Bi-LSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output.

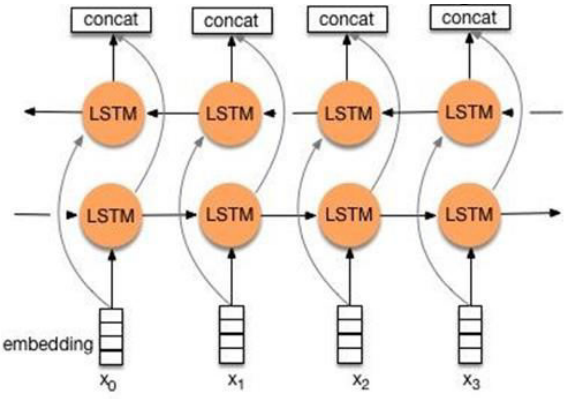

Figure 5. Bi-directional LSTM.

3.3.3. CRF for predict the output

Conditional Random Field (CRF) (Lafferty et al., 2001 [10]) is a type of graphical model designed for labeling sequence of data. Although the LSTM is likely to handle the sequence of the input data by learning the dependencies between the inputs at each time step but it predicts the outputs independently. The CRF, therefore, is beneficial to explore the correlations between outputs and jointly decode the best sequence of labels. In NER task, we implement the CRF on the top of Bi-LSTM instead of the softmax layer and take outputs of Bi-LSTM as the inputs of this model. The parameter of the CRF is the transition matrix $\boldsymbol{A}$ where $\boldsymbol{A}_{i,j}$ represents the transition score from tag i to tag j. The score of the input sentence $\boldsymbol{x}$ along with the sequence of tags $\boldsymbol{y}$ is computed as follow:

$$S(\boldsymbol{x}, \boldsymbol{y}, \theta \cup \boldsymbol{A}_{i,j}) = \sum_{t=1}^{T} (\boldsymbol{A}_{y_{t-1}, y_t} + f_{\theta_{(y_t,t)}})$$

where θ is the parameters of Bi-LSTM, f_θ is the score outputed by Bi-LSTM, and T is the number of time steps. Then the tag-sequence likelihood is computed by the softmax equation:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{A}) = \frac{\exp(S(\boldsymbol{x}, \boldsymbol{y}, \theta \cup \boldsymbol{A}_{i,j}))}{\sum_{y' \in Y} \exp(S(\boldsymbol{x}, \boldsymbol{y}', \theta \cup \boldsymbol{A}_{i,j}))}$$

where Y is the set of all possible output sequences. In the training stage, we maximize the log-likelihood function:

$$L = \sum_{i=1}^{N} \log p(y^i|x^i; \boldsymbol{A})$$

where N is the number of training samples. In the inference stage, the Viterbi algorithm is used to find the output sequence $\boldsymbol{y}^*$ that maximize the conditional probability:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in Y}{argmax}\, p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{A})$$

3.3.4. CNNs - Bi-LSTM - CRF

Figure 6 illustrates the architecture of neural network in detail. For each word, the character-level representation is computed by the CNN in figure 3 with character embeddings as inputs. Then the character-

level representation vector is concatenated with the word embedding vector to feed into the Bi-LSTM network. Finally, the output vectors of BLSTM are fed to the CRF layer to jointly decode the best label sequence. As shown in figure 6, dropout layers are applied on both the input and output vectors of Bi-LSTM. Experimental results show that using dropout significantly improve the performance of this model.

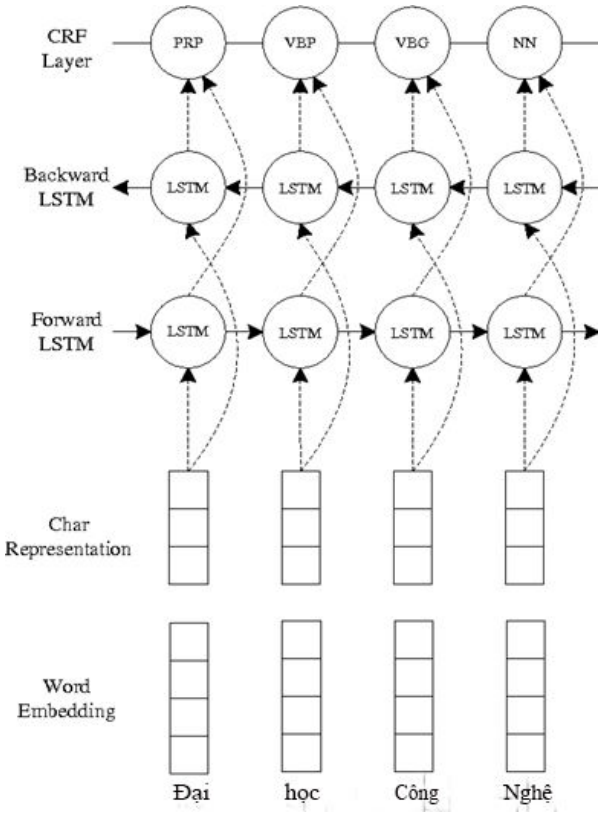

Figure 6. The main architecture of deep neural network.

3.3.5. Network training

Word Embeddings

For Vietnamese, we use Facebook's publicly available pre-trained word vectors, trained on Wikipedia using fastText. It can be downloaded from:

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Character Embeddings

Character embeddings are initialized with uniform samples from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$, where we set dim = 30.

Weight Matrices and Bias Vectors

Matrix parameters are randomly initialized with uniform sample from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$, where r and c are the number of rows and columns in the structure (Glorot and Bengio, 2010 [24]). Bias vectors are initialized to zero, except the bias b_f for the forget gate in Bi-LSTM, which is initialized to 1.0 (Jozefowicz et al., 2015 [19]).

Optimization Algorithm

Parameter optimization is performed with mini-batch stochastic gradient decent (SGD) with batch size 10 and momentum 0.9. We choose an initial learning rate of η_0 = 0.015, and the learning rate is updated on each epoch of training as $\eta_t = \eta_0/(1 + \rho t)$, with decay rate $\rho = 0.05$ and t is the number of epoch completed. To reduce the effects of "gradient exploding", we use a gradient clipping of 5.0 (Pascanu et al., 2012 [20]). We explored other more sophisticated optimization algorithms such as AdaDelta (Zeiler, 2012 [13]), Adam (Kingma and Ba, 2014 [8]) or RMSProp (Dauphin et al., 2015 [25]), but none of them meaningfully improve upon SGD with momentum and gradient clipping in our preliminary experiments.

Early Stopping

We use early stopping (Giles, 2001 [17]; Graves et al., 2013 [3]) based on performance on validation sets. The "best" parameters appear at around 50 epochs, according to my experiments.

Fine Tuning

For each of the embeddings, we fine-tune initial embeddings, modifying them during gradient updates of the neural network model by back-propagating gradients. The effectiveness of this method has been previously explored in sequential and structured prediction problems (Collobert et al., 2011 [18]; Peng and Dredze, 2015 [14])

Dropout Training

To mitigate over-fitting, we apply the dropout method (Srivastava et al., 2014 [15]) to regularize my model. As shown in figure 6, we apply dropout on both the input and output vector of Bi-LSTM. We fix dropout rate at 0.5 for all dropout layers through all the experiments. We obtain significant improvements on model performance after using dropout.

Tuning Hyper-Parameters

Table 3 summarizes the chosen hyper-parameters for all experiments. We tune the hyper-parameters on the development sets by random search. Due to time constrains it is infeasible to do a random search across the full hyper-parameter space. We set the state size of LSTM to 200. Tuning this parameter did not significantly impact the performance of my model. For CNN, we use 30 filters with window length 3.

Table 3. Hyper-parameters for all experiments.

Layer	Hyper-parameter	POS	NER
CNN	window size	3	3
	number of filters	30	30
LSTM	state size	200	200
	initial state	0.0	0.0
	peepholes	no	no
Dropout	dropout rate	0.5	0.5
	batch size	10	10
	initial learning rate	0.01	0.015
	decay rate	0.05	0.05
	gradient clipping	5.0	5.0

3.4. Text Normalization

In text normalization, some of the named entities are transformed to make it consistent. Table 4 shows some of the transformations performed on several text phrases.

In normalization phase, we expand some of abbreviations using dictionaries similar to the dictionary given in Table 5. For example, the abbreviation "B.S." is expanded as "Bachelor of Science". We also convert some of the text into a new form. For example, the first letters in a person's name is capitalized as shown in Table 4.

Table 4. Applying Text Normalization using Data-Dictionary.

Input	Output	Type
B.S.	Bachelor of Science	Degree
JOHN DOE	John Doe	Name
University of ABC	ABC University	Institution

Table 5. Sample Data-Dictionary of the degree chunker.

Term (Full Form/word)	Abbreviation
Bachelor of Science	B.S.,BS ,BSc
Master of Science	M.S.,MS ,MSc
Bachelor of Arts	B.A., BA
Doctor of Philosophy	Ph.D., PhD
Doctor of Medicine	Medicine Doctor, M.D.
Bachelor of Computer Application	BCA,B.C.A

4. Experiments

4.1. Statistics of dataset

We collected about 1000 CV files and then labeled to create dataset. The table 6 below shows the statistics of training data labels, which includes the number of each label in the dataset.

Table 6. Statistics of dataset.

Label	Amount	Percentage (%)
Degree	488	3.20
Experience	645	4.29
Name	986	6.56
Address	3279	21.83
University	1055	7.02
Skill	8566	57.10

Here, for *Address*, each name of street, city, district… corresponds to one entity. For example, the sentence "105 Doãn Kế Thiện, Cầu Giấy, Hà Nội" has 3 named entities, they are "105 Doãn Kế Thiện", "Cầu Giấy" and "Hà Nội". As you can see, the amount of *Skill* label is biggest, with 8566 in total, that because each candidate can have multi-skill (skill in this work is programming language, as we focus on IT domain). And that of *Degree* label is smallest with only 986 entities in dataset.

4.2. Main results

The table 7 reveals the results on the test set, train on above dataset. We evaluate 2 labels Degree and Experience on Rule-based method and the remain labels on Deep Learning method.

Table 7. Evaluation of our model.

Label	Precision (%)	Recall (%)	F1 (%)
Degree	66.47	67.53	66.99
Experience	70.12	69.77	69.94
Name	82.06	70.51	75.86
Address	76.11	69.77	72.80
University	86.80	78.66	82.53
Skill	90.05	92.32	91.17

As you can see from the table, our model can handle flexible-structure label quite well, with 75.86% F1-score for *Name* label. The performance on other labels is also great with the accuracy over 80% for *University* and *Skill* label. The rare data of *Address* label, along with its flexible structures, which consists of both numbers and words, requires the model to generate a unique vector for each number that leads to the low accuracy.

4.3. Comparison with other Deep Learning model

For Deep Learning method, we run experiments to dissect the effectiveness of each component (layer) of our neural network architecture by ablation studies. We compare the performance with three baseline systems: the Bi-directional RNN (BRNN), the Bi-directional LSTM (BLSTM) and CNN-Bi LSTM (the combination of BLSTM with CNN to model character-level information). All there models are run using the same parameters as shown in section 3.3.5. Table 8 is the results on both validation set and test set.

Table 8. Performance of our model on both the validation and test sets, together with three baseline systems.

Model	Validation			Test		
	Prec.	**Recall**	**F1**	**Prec.**	**Recall**	**F1**
Bi-RNN	88.06	86.23	87.14	78.56	76.13	77.33
Bi-LSTM	89.21	90.67	89.93	79.88	76.75	78.28
CNN-Bi-LSTM	90.55	91.02	91.29	82.34	77.20	79.69
CNN-Bi-LSTM-CRF	91.46	92.06	92.77	83.80	78.85	81.25

According to the results shown in table 8, Bi-LSTM obtain better performance than Bi-RNN on all evaluation metrics of both validation set and test set. CNN-Bi-LSTM models significantly outperform the Bi-LSTM model, showing that character-level representations are important for linguistic sequence labeling task. Finally, by adding CRF layer for joint decoding we achieve significant improvements over CNN-Bi-LSTM models on all metrics. This demonstrates that jointly decoding label sequences can significantly benefit the final performance of neural network models.

5. Conclusions

In this research project, we applied Rule-based method and Deep Learning method to build a model in order to extract useful information from resumes files. Our model have achieved significant results with over 81% for Named Entity Recognition. We also compared my model with other systems.

However, we still recognize several drawbacks of our work that could be improved. The amount of data we use to train our model is not big enough, so that the bigger our training data is, the more accurate our model is. In terms of LSTM model, we can make use of the power of neural networks with larger dataset. Also, the model requires powerful computers for calculation. we can enhance the hardware systems to have better performance.

There are several potential directions for future work. First, we will focus on analyzing and handling data to improve the performance in this Information Technology domain. We will collect more CV files, label it and make a bigger set of data. This work requires a huge amount of time and even needs a lot of people if we want a really big set of data, a big enough data set.

Furthermore, we will improve our model to detect and extract more information. We not only extract the Named Entities as defined in this work but also detect every other necessary information such as birthday, email, company, etc.

Another interesting direction is to apply our model to data from other CV domains. In this work, we choose Information Technology domain but there are still many more domains that are promising like Accounting, Banking and Finance, etc. Solving problem in these domains can help us to solve the big problem.

Finally, we tend to extend our work in other languages. It sounds interesting and expects good results.

Acknowledgments

We thanks to PhD. Xuan Hieu Phan - Department of Information Systems, Faculty of Information Technology, University of Engineering and Technology, for his guidance, motivation and continuous

support throughout the whole project and reviewing.

References

[1] ALEX Resume Parsing, http://www.hireability.com/ALEX/ (Accessed on Feb 2, 2012).

[2] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013, "Speech recognition with deep recurrent neural networks", *In Proceedings of ICASSP-2013*, pages 6645–6649. IEEE.

[3] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks 5.2 (1994),* pages 157-166.

[4] C. Siefkes and P. Siniakov, "An Overview and Classification of Adaptive Approaches to Information Extraction", *Journal on Data Semantics*, vol.4, Springer, 2005, pp. 172–212.

[5] Cicero D Santos and Bianca Zadrozny. 2014, "Learning character-level representations for part-of-speech tagging", *In Proceedings of ICML-2014*, pages 1818–1826.

[6] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015, "Transition-based dependency parsing with stack long short-term memory", *In Proceedings of ACL-2015* (Volume 1: Long Papers), pages 334–343, Beijing, China, July

[7] Daxtra CVX, http://www.daxtra.com/ (Accessed on Feb 2, 2012).

[8] Diederik Kingma and Jimmy Ba. 2014, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980.*

[9] Jason PC Chiu and Eric Nichols. 2015, "Named entity recognition with bidirectional lstm-cnns", *arXiv preprint arXiv:1511.08308.*

[10] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", *In Proceedings of ICML- 2001*, volume 951, pages 282–289.

[11] Krizhevsky, A., Sutskever, I., and Hinton, G.E, "Imagenet classification with deep convolutional neural networks", *In NIPS*, 2012.

[12] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D, "Backpropagation applied to handwritten zip code recognition. Neural Comput", 1(4):541–551, 1989.

[13] Matthew D Zeiler. 2012, "Adadelta: an adaptive learning rate method*", arXiv preprint arXiv:1212.5701.*

[14] Nanyun Peng and Mark Dredze. 2015, "Named entity recognition for chinese social media with jointly trained embeddings", *In Proceedings of EMNLP-2015*, pages 548–554, Lisbon, Portugal, September.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, 15(1):1929– 958.

[16] PALSHIKAR, G.K., 2011, "Techniques for named entity recognition: a survey", *TRDDC Technical Report.*

[17] Rich Caruana Steve Lawrence Lee Giles. 2001, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping", *In Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press.

[18] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011, "Natural language processing (almost) from scratch", *The Journal of Machine Learning Research*, 12:2493–2537.

[19] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015, "An empirical exploration of recurrent network architectures", *In Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

[20] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012, "On the difficulty of training recurrent neural networks", *arXiv preprint arXiv:1211.5063.*

[21] S.Sarawagi, "Information Extraction", *Foundations and Trends in Databases*, vol. 1, 2008, pp 261-377.

[22] Sovren Resume/CV Parser, http://www. sovren.com/ (Accessed on Feb 2, 2012).

[23] Sepp Hochreiter and Jurgen Schmidhuber. 1997, "Long short-term memory. Neural computation", 9(8):1735–1780.

[24] Xavier Glorot and Yoshua Bengio. 2010, "Understanding the difficulty of training deep feedforward neural networks", *In International conference on artificial intelligence and statistics*, pages 249–256.

[25] Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization", *arXiv preprint arXiv:1502.04390.*