DeepVar: An End-to-End Deep Learning Approach for Genomic Variant Recognition in Biomedical Literature

Chaoran Cheng,[1] **Fei Tan,**[2] **Zhi Wei**[1]*
[1]New Jersey Institute of Technology, Newark, NJ, USA
[2]Yahoo Research, New York, NY, USA
{cc424, zhi.wei}@njit.edu, fei.tan@verizonmedia.com

Abstract

We consider the problem of Named Entity Recognition (NER) on biomedical scientific literature, and more specifically the genomic variants recognition in this work. Significant success has been achieved for NER on canonical tasks in recent years where large data sets are generally available. However, it remains a challenging problem on many domain-specific areas, especially the domains where only small gold annotations can be obtained. In addition, genomic variant entities exhibit diverse linguistic heterogeneity, differing much from those that have been characterized in existing canonical NER tasks. The state-of-the-art machine learning approaches heavily rely on arduous feature engineering to characterize those unique patterns. In this work, we present the first successful end-to-end deep learning approach to bridge the gap between generic NER algorithms and low-resource applications through genomic variants recognition. Our proposed model can result in promising performance without any hand-crafted features or post-processing rules. Our extensive experiments and results may shed light on other similar low-resource NER applications.

1 Introduction

Due to the up-surging volume of new biomedical literature in the past decades, it becomes infeasible for researchers to access all those up-to-date publications manually. The automated information extraction tools play a critical role in assisting researchers to keep up with the explosive knowledge effectively. In general, the first step is to identify name entities from text, which is termed as Named Entity Recognition (NER), a common task in the nature language processing field. In the biomedical context, entities are typically short phrases as the representations of a specific object, e.g., names of genes or proteins, genetic variants, diseases, drugs, etc. Moreover, a noticeable amount of those entities contain letters, digits, and punctuation, resulting in more complex semantic alternations and differing much from entities characterized in news or conventional articles.

To identify named entities present in the text, statistical approaches such as Maximum Entropy (ME) (Berger, Pietra, and Pietra 1996) and Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira 2001) are used in most of the previous works with either learning patterns associated with a particular type of entities or hand-built rules. The performance of such algorithms heavily depends on the design of hand-crafted features. Recently, the Deep Neural Network (DNN) models have increasingly been used in generic NER tasks and achieved significant success, pushing most of the benchmarks to a new level. More importantly, those models minimized the feature engineering efforts by learning the hidden patterns from a large volume of labeled samples.

Our goal in this work is to develop an end-to-end DNN NER model that can automatically identify variants in biomedical literature and classify them into a set of predefined types. However, due to the prohibitive cost of expert curation, the size of curated training data with gold label annotations is often restricted in biomedical domains. As shown in Table 1, the sample size of the benchmark dataset of variants, tmVar (Wei et al. 2013), is much smaller than others. Furthermore, it exhibits more exotic linguistic heterogeneity. The complex morphological heterogeneity exacerbates the challenge for solving this problem, let alone the small data size. Despite numerous attempts on other biomedical benchmarks in the past, it is the first attempt to leverage a deep learning approach for the genomic variants recognition. The main challenges in this work include:

- To minimize feature engineering effort, automatically generalizing hidden diverse linguistic patterns is harder from limited training resources.
- To differentiate the ambiguous entities or synonym, learning some effective feature representation is harder with shallow networks from restricted resources.
- To limit the false positive error, both the entity identification and the entity boundaries need to be accurately inferred, which is critical for downstream applications such as entity normalization and relation extraction.

In this work, we took full advantage of the generic state-of-the-art deep learning algorithms and proposed a Deep Variant (DeepVar) Named Entities Recognition model. We aimed to find a principled way to transfer domain knowledge and build an end-to-end DeepVar model. Our results show

*Corresponding Author

Table 1: Comparison of Other Data Sets with Ours to Demonstrate The Extreme Low-Resource Situation in Our Work

Data Set	Size	Entity types and counts	Named Entity Example
BC4CHEMD	47,402 sentences	Chemical (84,310)	(25)MgPMC16; SAHA
BC5CDR	30,677 sentences	Chemical(15,935) Disease(12,852)	cyclosporin A; L-dopa cardiovascular arrhythmias; swelling
BC2GM	20,000 sentences	Gene/Protein (24,583)	S-100; Cdc42; RecA; ROCK-I
JNLPBA	13,484 sentences	Cell Line(4,330) DNA (10,589) Gene/Protein (20,448) Cell Type (8,649) RNA(1,069)	Jurkat T-cells; Hsp60-specific T cells cytokine gene; human interleukin-2 gene; NF-kappaB site; Hsp60; retinoic acid receptors 16HBE human bronchial epithelial cells GR mRNA; glucocorticoid receptor mRNA
NCBI-Disease	8,336 sentences	Disease(6,881)	MCF-7 tumours; breast and ovarian cancer
tmVar - Ours	4,783 sentences	Protein Mutation (653) DNA Mutation (751) SNP (136)	p.Pro246HisfsX13; S276T; Arg987Ter c.399_402del AGAG; Ex2+860G>C; -866 promoter(G/A); rs2234671; rs1639679

that DeepVar could achieve better performance than state-of-the-art algorithms using significantly less domain knowledge and without any feature engineering.

2 Related Works

In this section, we summarized the recent works in generic NER and the recent efforts of applying generic NER algorithms to the biomedical domain. We also outlined the recent works in genomic variants recognition.

Generic NER. In earlier works, statistical machine learning systems have proven its success for NER (Nadeau and Sekine 2007; Nothman, Murphy, and Curran 2009) with various feature engineering efforts like building internal linguistic features. More recently, due to the development of deep learning techniques, it becomes a fashion in NER applications to minimize the efforts of feature engineering and build an end-to-end system. The first attempt to use deep learning in NER task should be the SENNA system (Collobert et al. 2011), which still utilized lots of hand-crafted features. The current state-of-the-art approaches regulate both the word level and character level representations intertwined by both bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) Neural Network and Convolutional Neural Network (CNN) (Zeiler et al. 2010). Some works focused on building the shallow word-level representations with character-based feature through CNN (Collobert et al. 2011; Zhang, Zhao, and LeCun 2015; Kim et al. 2016; Strubell et al. 2017), or bidirectional LSTM (BiLSTM) (Ma and Hovy 2016; Strubell et al. 2017). The majority of works combined both word-level and character-level features to achieve the best performance. Nevertheless, some still applied slight pre-processing steps like normalizing digit characters, while some works employed marginal hand-crafted features to some extent (Chiu and Nichols 2016; Strubell et al. 2017). Lample et al. (2016) and Ma and Hovy (2016) achieved the end-to-end manner without any hand-crafted effort.

BioNER. The Biomedical Named Entity Recognition (BioNER) tasks focus on extracting biomedical domain entities such as cell lines, diseases, genes, and proteins. Various similar machine learning-based approaches have also been applied in earlier works and achieved good performance. The widely used hand-crafted features include different types of linguistic features such as orthographic features, word-shape features, n-gram features, dictionary features, and context features, as well as domain-specific features from biomedical terminologies. Various works have been done on several BioNER tasks to prove the effectiveness of aforementioned models. Habibi et al. (2017) investigated the effectiveness of approach proposed in (Lample et al. 2016) for chemicals, diseases, cell lines, species, and genes name recognition, while Dernoncourt et al. (2017) verified the same approach on patient notes. Yoon et al. (2019) investigated the approaches of (Ma and Hovy 2016) on chemicals and disease entities. Wang et al. (2018) utilized the similar multitask architecture in (Liu et al. 2018) and verified on chemicals, cell lines, disease, genes, and other name recognition. Xu, Wang, and He (2018) proposed a modified framework based on (Lample et al. 2016) by adding extra sentence level representation as global attention information and verified on clinical NER task. Nevertheless, some recent works still need to elaborate on marginal external information.

Genomic Variants Recognition. With respect to the genomic variants recognition, all the previous works including MutationFinder (Caporaso et al. 2007), EBNF (Laros et al. 2011), OpenMutationMiner (Naderi and Witte 2012), tmVar (Wei et al. 2013), SETH (Thomas et al. 2016), and NALA (Cejuela et al. 2017) employed dozens of regular expressions to build orthographic and morphological features, like word shape, prefixes, and suffixes, for their variants entities identification systems. Since the regular expressions used for generating customized hand-crafted features are fixed and can only describe limited patterns, all the previous works mainly focused on techniques improving the regular expressions to capture more patterns (Naderi and Witte 2012; Wei et al. 2013; Thomas et al. 2016). Nevertheless, they still need to add a bunch of post-processing steps to achieve better results (Wei et al. 2013; Cejuela et al. 2017). Moreover, despite the efforts in recent BioNER tasks, the leverage of deep learning approach in variant identification tasks remains open in literature, and to build an end-to-end approach can be challenging.

Word Embedding. It's worth noting that most of those

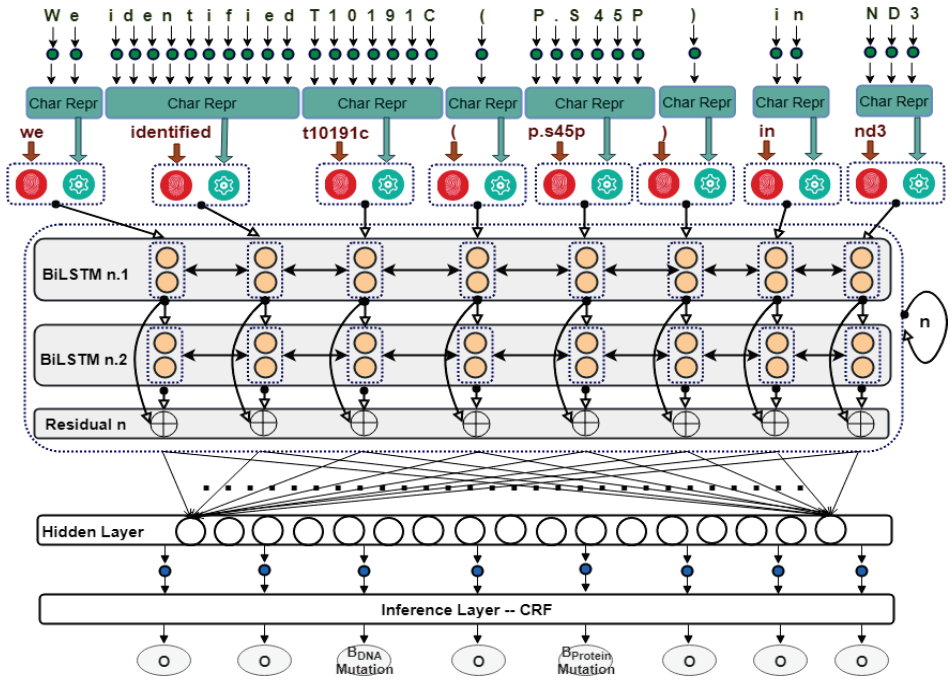

Figure 1: The Architecture of Proposed DeepVar NER Model. The green color module on top represent character input and sequence representation learning; the red circle icon represents word embedding; the gray boxes represent word sequence representation learning module including multiply BiLSTM and Residual layers. The sentence "We identified T10191C(P.S45P) in ND3." used in the figure is for illustration purpose.

works employed the word-level distributional representations, well-known as word embedding. However, the pre-trained embeddings on generic corpora cannot fully capture the semantics of biomedical entities. One of the common challenges is the OOV words, which can be rare terms like mutants or unseen forms of known words like chemical names. Those entities are not typo and have high occurrence but cannot be found in the canonical pre-trained embeddings. Recently, the word representations pretrained on a large collection of domain-specific texts (PubMed, PMC, etc.) are proved its superiority than generic word embeddings (Habibi et al. 2017; Mohan et al. 2018). Moreover, the most recent proposed contextual embeddings such as ELMO (Peters et al. 2018), Flair (Akbik et al. 2019), and BERT (Devlin et al. 2019) achieved state-of-the-art performance on all the generic NLP applications. Their domain-specific embeddings (BioELMO, BioFlair, and BioBert), which are trained on the large biomedical corpus, are simultaneously made available to the public. Various works (Jin et al. 2019; Peng, Yan, and Lu 2019) showed that they outperform word2vec (Mikolov et al. 2013) on BioNER tasks. In our experiments, we also investigated different pre-trained BioEmbeddings.

3 Deep Variants Identification Model

In this section, we presented our DeepVar NER model for identifying variants in a low-resource data set. We focused on neural sequence representation learning to capture contextual information and hidden linguistic pattern without hand-crafted features or regular expressions. The architecture is shown in Figure 1. As illustrated, our DeepVar model contains three parts:

Input Embeddings. Each word in the sentence has two types of input: word-level (words in red color) and character-level (characters in green color). For character-level input, we applied one-hot encoding (green circle on top); with respect to word-level input, we used the word embedding (red circle icon; Sec. 2). It's noted that the word embeddings are pre-trained on a separate large collection of biomedical corpus, while character embeddings are built from our variant BioNER task.

Feature Representation Learning. The character representation (green circle icon) is learned from module with LSTM or CNN ("Char Repr"; Sec. 3.1). Then it would be concatenated with word embeddings as the input of word sequence representation learning module (gray boxes in the middle; Sec. 3.2). This module contains the stacked BiLSTM networks with residual layer integrated, and it's designed to capture long-term information and effective contextual representations.

Inference Module. The final word feature representation for each word will be the hidden status from the hidden layer (blue circle). The CRFs inference layer will take it and assign labels to each word (Sec. 3.3).

3.1 Character feature Representation

Character information has been proven to be critical for entity identification tasks (Chiu and Nichols 2016; Lample et

al. 2016; Ma and Hovy 2016). First of all, character embedding could handle the Out-of-Vocabulary (OOV) words to some extent since it could enclose the morphological similarities to some established words. Moreover, it also could be able to insert the orthographic and linguistic patterns for variants such as prefix, suffix, and punctuation. For example, mutation names often contain alphabets, digits, hyphens, and other characters like "HIV-1", "IL2", "rs2297882", and "C>T". It's crucial to learn all those hidden morphological and orthographic patterns automatically for inference.

Table 2: The Look-up Table for Character One-hot Encoding

letters	abcdefghijklmnopqrstuvwxyz
digits	0123456789
others	,;.!?:'‘“”/|_@#$%ˆ&*˜+−=<>()[]{}

We represented the character-level input by one-hot encoding through a lookup table. The lookup table in our work contains 70 characters, including 26 English letters, ten digits, 33 other characters, and one placeholder for the unknown character. The full list is shown in Table 2. Subsequently, each word instance is then represented by a sequence of $m = 70$ sized vectors with character sequence length l, where l is a hyperparameter in our work. Then LSTM or CNN is used to learn character-level representation to capture the hidden morphological and orthographic patterns:

Character CNN. Chiu and Nichols (2016) and Ma and Hovy (2016) have investigated the effectiveness of using the CNN structure to encode character sequences. In our work, we employed the same architecture as in (Ma and Hovy 2016). More specifically, one CNN layer was used following with max-pooling to capture character-level representation.

Character BiLSTM. Lample et al. (2016) utilized the BiLSTM to model the global character sequence information. In our work, we employed the same architecture as (Lample et al. 2016) in which final states from the *left-to-right* forward LSTM and *right-to-left* backward LSTM are concatenated as character sequence representations.

3.2 Word Representation Learning

We employed the BiLSTM in our work to model word-level representations as it's more widely used (Lample et al. 2016; Ma and Hovy 2016; Chiu and Nichols 2016; Liu et al. 2018) and more powerful to capture the contextual distributional sensitivity. As shown in the gray boxes of Fig. 1, our word representation learning module includes n units of modules in which n is a hyper-parameter. Each unit includes 2 BiLSTM layers stacked together, followed by a residual layer. The residual layer would take the hidden states from both BiLSTM layers and apply the transformation.

Basically, the input to an LSTM network is a sequence of vectors $X = \{x_1, x_2, \ldots, x_T\}$, where x_t is a representation of a word in the input sentence x at certain layer of the network. The output is a sequence of vectors $H = \{h_1, h_2, \ldots, h_T\}$, where h_t is a hidden state vector storing all the useful information at time t. At step t of the recurrent calculation, the network takes x_t, c_{t-1}, h_{t-1} as inputs and produces c_t, h_t through the input (i_t), forget (f_t) and output (o_t) gates via the following intermediate calculations:

$$\mathbf{i}_t = \sigma(\mathbf{W}^i \mathbf{x}_t + \mathbf{U}^i \mathbf{h}_{t-1} + \mathbf{b}^i) \quad (1)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{x}_t + \mathbf{U}^f \mathbf{h}_{t-1} + \mathbf{b}^f) \quad (2)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o \mathbf{x}_t + \mathbf{U}^o \mathbf{h}_{t-1} + \mathbf{b}^o) \quad (3)$$

$$\hat{\mathbf{c}}_t = \sigma(\mathbf{W}^c \mathbf{x}_t + \mathbf{U}^g \mathbf{h}_{t_1} + \mathbf{b}^g) \quad (4)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + i_t \odot \hat{c}_t \quad (5)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \quad (6)$$

where $\sigma(\cdot)$ and $tanh(\cdot)$ is the element-wise sigmoid and hyperbolic tangent functions, and $\odot$ denotes element-wise product. $\mathbf{W}^i, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^c$ denote the weight matrices of different gates for input $\mathbf{x}_t$, and $\mathbf{U}^i, \mathbf{U}^f, \mathbf{U}^o, \mathbf{U}^c$ are the weight matrices for recurrent hidden state $\mathbf{h}_t$. $\mathbf{b}^i, \mathbf{b}^f, \mathbf{b}^o, \mathbf{b}^c$ denote the bias vectors. As shown in above formulation, we used the LSTM design (Hochreiter and Schmidhuber 1997) without peephole connections. A BiLSTM includes forward LSTM and backward LSTM. The hidden states of the forward and backward LSTM are concatenated for each word and are the input of the next layer, in our case, the BiLSTM transformation.

The semantic representations learned from a shallow network in (Lample et al. 2016; Chiu and Nichols 2016; Ma and Hovy 2016) isn't able to differentiate the variants apart from genes/proteins having similar orthographic patterns. However, simply increasing the depth of a network won't necessarily improve the performance, and on the contrary, it often leads to a decline in performance beyond a certain point (Srivastava, Greff, and Schmidhuber 2015). The introduce of residual could bridge some shallow global information to deeper levels, and facilitate to address the vanishing gradient problem when training a deeper network (He et al. 2016). In our work, we used the identity residual formulated as:

$$y(x) = F(x) + x \quad (7)$$

where $F(\cdot)$ is a nonlinear parametric function.

3.3 Inference Procedure

The CRFs is commonly used for labeling and segmenting sequences tasks, and also has been extensively applied to NER. It's especially helpful for tasks with strong dependencies between token tags. Reimers and Gurevych (2017) and Yang, Liang, and Zhang (2018) demonstrated that CRFs can deliver a larger performance increase than the softmax classifier across all NER tasks. Reimers and Gurevych (2017) also suggested that a dense layer followed by a linear-chain CRF as variant CRF classifier would be able to maximize the tag probability of the complete sentence. In our work, we employed the same variant CRF classifier design for inference.

First of all, the output for the sequence from the last residual layer is mapped with a dense layer and a linear chain CRF layer to the number of tags. The linear-chain CRF maximizes the tag probability of the complete sentence. More formally, given an input sentence x of length N $x = [w_1, w_2, \ldots, w_N]$ in which w_t is the a word in sentence, we

predict corresponding variant types $Y = [y_1, y_2, \ldots, y_N]$. The score of a sequence of tags z is defined as:

$$S(x, y, z) = \sum_{t=1}^{N-1} \mathcal{T}_{z_{t-1}, z_t} + \sum_{t=1}^{N} \mathcal{U}_{x_t, z_t} \quad (8)$$

where $\mathcal{T}$ is a transition matrix in which $\mathcal{T}_{p,q}$ represents the score of transitioning from tag p to tag q and $\mathcal{U}_{x_t, z_t}$ represents the score of assigning tag z to word w given representation x at time step t. Given the ground truth sequence of tags z, we minimize the negative log-likelihood loss function during the training phase:

$$\begin{aligned} \mathcal{L} &= -log\mathcal{P}(z|x) \\ &= log \sum_{\hat{z} \in \mathcal{Z}} e^{S(x,y,\hat{z})} - S(x, y, z) \end{aligned} \quad (9)$$

where $\mathcal{Z}$ is the set of all possible tagging paths. For efficient training and decoding, viterbi algorithm is used.

4 Experiment Setup

4.1 Data

We trained and tested our model on the expert-annotated corpus from tmVar (Wei et al. 2013), while 20% of the training is held out for validation. The details are shown in Table 3.

Table 3: The Data in Our Work

Data Set	Size
Training	2936 sentences
Validation	734 sentences
Testing	1113 sentences

Tokenization. The only preprocessing we performed on the data is customized tokenization. The traditional tokenization in generic NER tasks would split a sentence by the white space and remove all the digits and special characters. However, those digits and special characters like punctuation are part of the genomic variants. Moreover, due to the great variations of those entities, appropriate tokenization can significantly affect the performance. For example, whether the sequence "(lL-2)" is tokenized to {"(", "IL-2" and ")"} or {"(IL", "-", and "2)"} would result in considerable difference in representation learning and accuracy. In our work, we first tokenize a sentence using white space and characters in {" " # & $ _ * ; / \ ∼ ! ? = { } }, then for each token t, if there's any character from {"", . ':} at the end of t, then strip this character. Finally, strip the brackets if t is bracketed.

Annotation Scheme. The choice of annotation scheme varied from applications. There's no consensus on which one is better. Chiu and Nichols (2016) demonstrated that BIOES (for Begin, Inside, Outside, End, Single) could achieve considerable performance improvements over BIO (for Begin, Inside, Outside). Lample et al. (2016) showed Using BIOES and BIO yields similar performance. Reimers and Gurevych (2017) demonstrated that BIO scheme is preferred over BIOES through extensive experiments on varied NER tasks. Therefore, in our work, we adopted the BIO scheme without comparing it with BIOES or other schemes.

4.2 Evaluation

One challenge for NER research is establishing an appropriate evaluation metric (Nadeau and Sekine 2007). In particular, entities may be correctly delimited but misclassified, or entity boundaries may be mismatched. The partial matching (text offsets overlap, e.g., left match or right mach) or oversized boundaries may be considered as accurate tagging in some generic NER tasks. However, same as work in tmVar (Wei et al. 2013), we only considered exact matching (two entities match if their boundaries are identical and tags are correctly classified), and any other prediction was considered as misclassification.

In our work, we use the Precision (P), Recall (R), and macro-averages F1 score (F1) to evaluate different models. Moreover, there are three types of variants: DNA mutation, protein mutation, and SNP. Therefore, the tags used for training and prediction are B-DNAMutation, I-DNAMutation, B-ProteinMutation, I-ProteinMuataion, B-SNP (no I-SNP), and O. In the reported results for all DNN models, we removed the tag header B- and I-, and only used the tag body with their entity boundaries to calculate precision, recall and F1 score.

Table 4: Hyperparameters and Training Settings

Parameters	Values
max char length	15, 30, 50
char emb size	25, 50, 100
char emb dropout	0, 0.25, 0.5
char CNN filter size	30, 50, 70
char CNN window	3, 5, 7
char LSTM states	25, 50, 100
char LSTM dropout	0, 0.25, 0.5
max word length	115
word emb	BioW2V, BioELMO BioFlair, BioBert
word repr. learning unit n	1, 2
word LSTM states	50, 100, 200
word LSTM dropout	0, 0.25, 0.5
hidden states	50, 100, 200
hidden layer dropout	0, 0.25, 0.5
batch size	32, 64, 128
optimizer	SGD, RMSP, ADAM
learning rate	1e-4
learning rate decay	1e-5
clipnorm	1.0

4.3 Settings

We implemented our model using Keras with TensorFlow backend. The computations for a single model are run on Tesla P100 GPU. Table 4 summarizes the chosen hyperparameter settings for all DNN models. Moreover, the embed-

ding size for BioW2V is also a hyperparameter, which includes 50 and 100. With respect to the SGD optimizer, besides the common settings, we also set the momentum to 0.9 and used Nesterov. All the models are trained in 100 epochs with early stopping.

5 Results and Discussion

In this section, we reported our experimental results and investigated some key components in our model design. We also discussed the relation between genomic variants recognition with other similar NER tasks.

5.1 Results

We compared DeepVar with several state-of-the-art NER systems: (1) generic DNN; (2) vanilla DNN (Lample et al. 2016; Ma and Hovy 2016); and (3) rule-based machine learning variants identification systems (tmVar and nala). We also investigated both BiLSTM and CNN in learning the character-level representation and compared their role in different models. We performed extensive parameter tuning for the DNN models using settings shown in Table 4. For vanilla models, we used the same setting as in (Lample et al. 2016; Ma and Hovy 2016) on character feature learning and tuned other settings including the word embedding, word representation, and optimizer. For tmVar (Wei et al. 2013) and nala (Cejuela et al. 2017), we quoted their experimental results directly. It's noted that we used tmVar[b] (without post-processing) as baseline while tmVar[c] (with post-processing) as state-of-the-art benchmark. The results are reported in Table 5.

First of all, we observed that the DeepVar model achieves significantly higher F1 scores than state-of-the-art vanilla DNN models, nala, and tmVar ([a,b] without post-processing). DeepVar also achieves appreciably higher F1 score than generic DNN models. The DeepVar and generic DNN models differ at the introduction of residual layer, which is designed to learn better semantic representations by training deeper networks. For the results reported in Table 5, the generic DNN models achieved best performance using shallow network with single BiLSTM layer. Meanwhile, the result of DeepVar is very close to the best record of tmVar ([c] with extensive hand-crafted features and post-processing). However, it's worth noting that DeepVar is a truly end-to-end system requiring no preprocessing, feature engineering, nor post-processing. DeepVar should be able to achieve a higher score by adding the post-processing regex in tmVar and create a more practically useful tool in the real application.

5.2 Word Embeddings

In our experiments, we compared four different pre-trained domain-specific word embeddings: BioW2V, BioElmo, BioFlair, and BioBert (Beltagy, Cohan, and Lo 2019). More specifically, we used CBOW word2vec (Mikolov et al. 2013) model to train BioW2V on the large up-to-date collection of PubMed corpus. BioELMO[1] and BioFlair[2] are pre-trained on biomedical literature as well. We used the concatenated representations from the last three layers for BioELMO, and BioFlair took the stacked representations from pubmed-forward and pubmed-backward. BioBert[3] was pre-trained on scientific literature, we used the concatenated representations from the last 4 layers.

Table 5: The Results of Test Set Performance

Models	Char Repr	P (%)	R (%)	F1 (%)
DeepVar	BiLSTM	91.72	89.86	**90.78**
	CNN	90.67	90.48	90.58
DNN	BiLSTM	91.84	89.05	90.42
	CNN	90.91	89.25	90.07
Vanilla[†]	BiLSTM (Lample et al. 2016)	88.76	89.66	89.20
	CNN (Ma and Hovy 2016)	90.32	87.02	88.64
tmVar		85.81	80.82	83.24[a]
		92.01	83.72	**87.67**[b]
		91.38	91.40	**91.39**[c]
nala		87.00	92.00	89.00[d]

[†]same character learning settings, tuning other settings
[a]used BIO annotation scheme, no post-processing
[b]used 11 annotation scheme, no post-processing
[c]used 11 annotation scheme, with post-processing
[d]used partial match, result of exact match should be lower.

The best performance of DeepVar is reported on BioW2V, however, as shown in Table 6, the overall performances of BioELMO, BioBert, and BioFlair significantly outperform BioW2V in generic DNN models of which were usually built with shallow networks. This interesting observation demonstrated that word2vec can achieve compelling performance in deeper neural networks. Moreover, while the performances of BioBert and BioELMO are very close and slightly better than BioFlair, it's surprising that BioBert didn't outperform BioELMO. One suspicious factor is that the BioBert we used was pretrained on scientific articles that contain broader topics than biomedical domains, thus affected by the domain shift problem (Komiya and Shinnou 2018)[4].

Table 6: The Comparisons on Pre-trained Word Embeddings

Model	Embedding	P (%)	R (%)	F1 (%)
DeepVar	BioW2V	91.72	89.86	90.78
	BioELMO	90.67	90.48	90.58
	BioBert	91.49	89.45	90.46
	BioFlair	91.27	89.05	90.14
DNN	BioW2V	87.52	89.47	88.49
	BioELMO	91.84	89.05	90.42
	BioBert	90.97	89.86	90.41
	BioFlair	90.22	89.86	90.04

[1]https://allennlp.org/elmo
[2]https://github.com/zalandoresearch/flair
[3]https://github.com/allenai/scibert
[4]Recently, Lee et al. (2019) published an NCBI BioBert which is pre-trained on PubMed corpus. This BioBert should alleviate the domain shift problem.

5.3 Optimizer

For DeepVar training, we observed that RMSP slightly outperforms Adam while both of them significantly outperform SGD. For generic DNN models, we had the same observation over RMSP and Adam, while SGD has much worse performance. This observation is significantly divergent from knowledge learned from generic NER tasks (Lample et al. 2016; Ma and Hovy 2016; Reimers and Gurevych 2017; Yang, Liang, and Zhang 2018) in which SGD and Adam are preferred over RMSP.

Table 7: The Comparisons on Optimizers

Model	Optimizer	P (%)	R (%)	F1 (%)
DeepVar	SGD	87.45	85.86	86.65
	RMSP	91.72	89.86	90.78
	ADAM	91.84	89.05	90.42
DNN	SGD	82.52	82.35	82.44
	RMSP	91.84	89.05	90.42
	ADAM	88.36	90.87	89.60

5.4 Relation to other NER tasks

Deep Learning on small datasets is on the horizon of research field. State-of-the-art NER systems and some recent BioNER tasks with large datasets have been discussed in Section 2. However, it is not uncommon that we may encounter domain-specific applications for which only small datasets are produced with high-quality gold label annotations, due to the cost of expert curation. Besides the genomic variants recognition task in this study, another similar example in the industrial NER domain is personal identifier entity recognition from various products' logs, like user names, passwords, taggers, etc. The machine logging language is different from canonical natural language, and it's hard to read and interpret without domain knowledge. Out of hundreds of thousands of logging records, product experts may only be able to obtain hundreds of samples with gold labels for different types of entities. Those personal identifiers also exhibit quite diverse linguistic heterogeneity as we see in genomic variants, mixed with massive digits and punctuation. Moreover, it is critical to recognize the exact boundaries of personal identifiers and minimize the false negative for preventing privacy leakage. Hundreds of rule-based regular expressions are generally developed to capture identifiers, which are painstaking to maintain. It is also hard to generalize them across various products. Our study would motivate such similar NER tasks.

6 Conclusion

In this paper, We propose the first end-to-end neural network approach "DeepVar" for genomic variant entities identification. The proposed approach significantly outperformed the benchmark baseline and vanilla DNN models. While requiring no feature engineering nor post-processing, it achieved comparable performance to the state-of-the-art rule-based machine learning system. We also demonstrated through detailed analysis that the performance gain is achieved by the introduced residual, which facilitates to train a deeper network, and confirmed the domain-specific contextual word embeddings make significant contributions to the performance gain. The significant reduction of reliance on domain-specific knowledge would play a crucial role in certain expert-costly fields. Our investigation on key components may also shed light upon other deep low-resource NER applications.

7 Acknowledgments

This research was partially supported by Adobe Data Science Research Award.

References

Akbik, A.; Bergmann, T.; Blythe, D.; Rasul, K.; Schweter, S.; and Vollgraf, R. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 54–59.

Beltagy, I.; Cohan, A.; and Lo, K. 2019. Scibert: Pretrained contextualized embeddings for scientific text.

Berger, A. L.; Pietra, V. J. D.; and Pietra, S. A. D. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22(1):39–71.

Caporaso, J. G.; Baumgartner Jr, W. A.; Randolph, D. A.; Cohen, K. B.; and Hunter, L. 2007. Mutationfinder: a high-performance system for extracting point mutation mentions from text. *Bioinformatics* 23(14):1862–1865.

Cejuela, J. M.; Bojchevski, A.; Uhlig, C.; Bekmukhametov, R.; Kumar Karn, S.; Mahmuti, S.; Baghudana, A.; Dubey, A.; Satagopam, V. P.; and Rost, B. 2017. nala: text mining natural language mutation mentions. *Bioinformatics* 33(12):1852–1858.

Chiu, J. P., and Nichols, E. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12(Aug):2493–2537.

Dernoncourt, F.; Lee, J. Y.; Uzuner, O.; and Szolovits, P. 2017. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association* 24(3):596–606.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Habibi, M.; Weber, L.; Neves, M.; Wiegandt, D. L.; and Leser, U. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics* 33(14):i37–i48.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the*

IEEE conference on computer vision and pattern recognition, 770–778.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jin, Q.; Dhingra, B.; Cohen, W. W.; and Lu, X. 2019. Probing biomedical embeddings from language models. *NAACL HLT 2019* 82.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Komiya, K., and Shinnou, H. 2018. Investigating effective parameters for fine-tuning of word embeddings using only a small corpus. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, 60–67.

Lafferty, J.; McCallum, A.; and Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, 260–270.

Laros, J. F.; Blavier, A.; den Dunnen, J. T.; and Taschner, P. E. 2011. A formalized description of the standard human variant nomenclature in extended backus-naur form. *BMC bioinformatics* 12(4):S5.

Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C. H.; and Kang, J. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Liu, L.; Shang, J.; Ren, X.; Xu, F. F.; Gui, H.; Peng, J.; and Han, J. 2018. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1064–1074.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Mohan, S.; Fiorini, N.; Kim, S.; and Lu, Z. 2018. A fast deep learning model for textual relevance in biomedical information retrieval. In *Proceedings of the 2018 World Wide Web Conference*, 77–86. International World Wide Web Conferences Steering Committee.

Nadeau, D., and Sekine, S. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Naderi, N., and Witte, R. 2012. Automated extraction and semantic analysis of mutation impacts from the biomedical literature. In *BMC genomics*, volume 13, S10. BioMed Central.

Nothman, J.; Murphy, T.; and Curran, J. R. 2009. Analysing wikipedia and gold-standard corpora for ner training. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, 612–620.

Peng, Y.; Yan, S.; and Lu, Z. 2019. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. *arXiv preprint arXiv:1906.05474*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Reimers, N., and Gurevych, I. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Training very deep networks. In *Advances in neural information processing systems*, 2377–2385.

Strubell, E.; Verga, P.; Belanger, D.; and McCallum, A. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2670–2680.

Thomas, P.; Rocktäschel, T.; Hakenberg, J.; Lichtblau, Y.; and Leser, U. 2016. Seth detects and normalizes genetic variants in text. *Bioinformatics*.

Wang, X.; Zhang, Y.; Ren, X.; Zhang, Y.; Zitnik, M.; Shang, J.; Langlotz, C.; and Han, J. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics* 35(10):1745–1752.

Wei, C.-H.; Harris, B. R.; Kao, H.-Y.; and Lu, Z. 2013. tmvar: a text mining approach for extracting sequence variants in biomedical literature. *Bioinformatics* 29(11):1433–1439.

Xu, G.; Wang, C.; and He, X. 2018. Improving clinical named entity recognition with global neural attention. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, 264–279. Springer.

Yang, J.; Liang, S.; and Zhang, Y. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, 3879–3889.

Yoon, W.; So, C. H.; Lee, J.; and Kang, J. 2019. Collabonet: collaboration of deep neural networks for biomedical named entity recognition. *BMC bioinformatics* 20(10):249.

Zeiler, M. D.; Krishnan, D.; Taylor, G. W.; and Fergus, R. 2010. Deconvolutional networks. In *Cvpr*, volume 10, 7.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.