Character-Level Feature Extraction with Densely Connected Networks

Chanhee Lee[1], Young-Bum Kim[2], Dongyub Lee[1], HeuiSeok Lim[1]*
[1]Korea University, Republic of Korea
{chanhee0222, judelee93, limhseok}@korea.ac.kr
[2]Amazon Alexa
youngbum@amazon.com

Abstract

Generating character-level features is an important step for achieving good results in various natural language processing tasks. To alleviate the need for human labor in generating hand-crafted features, methods that utilize neural architectures such as Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) to automatically extract such features have been proposed and have shown great results. However, CNN generates position-independent features, and RNN is slow since it needs to process the characters sequentially. In this paper, we propose a novel method of using a densely connected network to automatically extract character-level features. The proposed method does not require any language or task specific assumptions, and shows robustness and effectiveness while being faster than CNN- or RNN-based methods. Evaluating this method on three sequence labeling tasks - slot tagging, Part-of-Speech (POS) tagging, and Named-Entity Recognition (NER) - we obtain state-of-the-art performance with a 96.62 F1-score and 97.73% accuracy on slot tagging and POS tagging, respectively, and comparable performance to the state-of-the-art 91.13 F1-score on NER.

1 Introduction

Effectively extracting character-level features from words is crucial in many Natural Language Processing (NLP) tasks, such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and Slot tagging. Thus, most state-of-the-art methods for these tasks exploit some kind of character-level features (Huang et al., 2015; dos Santos and Zadrozny, 2014). Recently, generating character-level features with neural architectures such as Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) has drawn much attention, mainly because it doesn't require human labor and shows superior performance (Ma and Hovy, 2016; dos Santos and Zadrozny, 2014). However, CNN struggles at distinguishing anagrams, and RNN is inherently slow due to its sequential nature.

In this paper, we propose an effective and efficient way of extracting character-level features using a densely connected network. The key benefits of the proposed method can be summarized as follows. First, it does not require any hand-crafted features or data preprocessing. Each word is processed based on n-gram statistics of the training data, and vectorized using bag-of-characters. Additional features are based on hexadecimal values of the character-set (e.g. UTF-16) and number of characters in the word. Second, it extracts effective character-level features while being efficient. State-of-the-art performance can be achieved using this method, and the feature extraction is done with a simple densely connected network with a single hidden layer. Third, it doesn't depend on features that are language or task specific, such as character type features or gazetteer (i.e. lists of known named entities such as cities or organization names). The only requirement for adopting this method is that the language should be processable as a sequence of words, which is made of sequence of characters. These benefits, combined with minimum requirements for application, make the proposed method an easy replacement for conventional methods such as CNN or RNN.

* corresponding author

Our contributions are three-fold: 1) We propose an effective yet efficient method for character-level feature extraction; 2) We quantitatively show that the proposed method is superior to CNN and RNN via extensive evaluation; 3) We achieve state-of-the-art or comparable to state-of-the-art performance on three of the most popular and well-studied sequence tagging tasks - Slot tagging, Part-of-Speech (POS) tagging, and Named Entity Recognition (NER).

2 Related Work

Prior to the introduction of neural architectures for character-level feature generation, manually engineered features were designed by experts based on language and/or domain knowledge. One example is word shape, in which each word is mapped to a simplified representation that encodes information such as capitalization, numerals, and length (e.g. CoNLL-2003 to AaAAA-0000). Finkel et al. (2005) combined this feature with other information such as n-grams and gazetteers to train a conditional Markov model for identification of gene and protein names in biomedical documents. Huang et al. (2015) introduced more hand-crafted features utilizing punctuation or non-letters and used these as an input to a Bi-LSTM-CRF tagger for POS tagging, CoNLL-2000 chunking, and CoNLL-2003 NER. Even though these kinds of hand-crafted features showed strong empirical results, they are more expensive than our approach in that they require expert knowledge of the target domain and language.

In recent years, methods that utilize neural networks to automatically extract character-level features have been proposed. The most widely adopted and successful method for this is CNN. dos Santos and Zadrozny (2014) combined this approach with a window-based fully-connected neural network tagger to perform English and Portuguese POS tagging. This work achieved state-of-the-art results in Portuguese and near state-of-the-art results in English. In Ma and Hovy (2016), a Bi-LSTM-CRF model incorporated with a character-level CNN is trained in an end-to-end fashion. They evaluated this approach on English POS tagging and NER, achieving state-of-the-art performance on both tasks. However, feature vectors generated by CNN are position-independent due to the max-over-time pooling layer, and are more sensitive to model weight initialization compared to the method proposed in this paper.

Another effective way of generating feature vectors from a variable length sequence of characters is to use RNN. For instance, Lample et al. (2016) extracted character-level features using a bi-directional LSTM and used them with pre-trained word embeddings as word representations for another Bi-LSTM-CRF model. Evaluating this model for NER, they obtained state-of-the-art results for Dutch, German, and Spanish, and close to state-of-the-art results for English. Intuitively, character-level feature generation via RNN should be more effective than CNN, since RNN processes each character sequentially and thus should form a better model of character ordering. However, Reimers and Gurevych (2017) empirically showed that these two methods have no statistically significant difference in terms of performance. Furthermore, RNN has a higher time-complexity caused by its sequential nature, which makes it less favorable.

3 Proposed Method

The proposed method is built on bag-of-characters (BOC) representation. However, BOC is prone to anagrams and thus is susceptible to word collisions, i.e. different words having the same vector representation. The main focus of the proposed method is to minimize word collision while maintaining the key benefits described above. To achieve this goal, we split the word into k pieces, and each piece is vectorized using BOC. Then, two non hand-crafted-features are extracted from the word - character order and word length. These sparse vectors are concatenated and normalized to form the sparse character-level feature vector. For a n-dimensional vector $\mathbf{x} = \langle x_1, x_2, \cdots, x_n \rangle$, normalizing is done as follows:

$$x'_i = \frac{x_i}{\sum_{j=1}^{n} x_j} \tag{1}$$

This sparse vector is then fed into a densely connected network with a single hidden layer to obtain the final dense character feature vector. Note that the sparse vector representation of each word is fixed, so it can be cached for efficiency. Figure 1 illustrates the overall process.

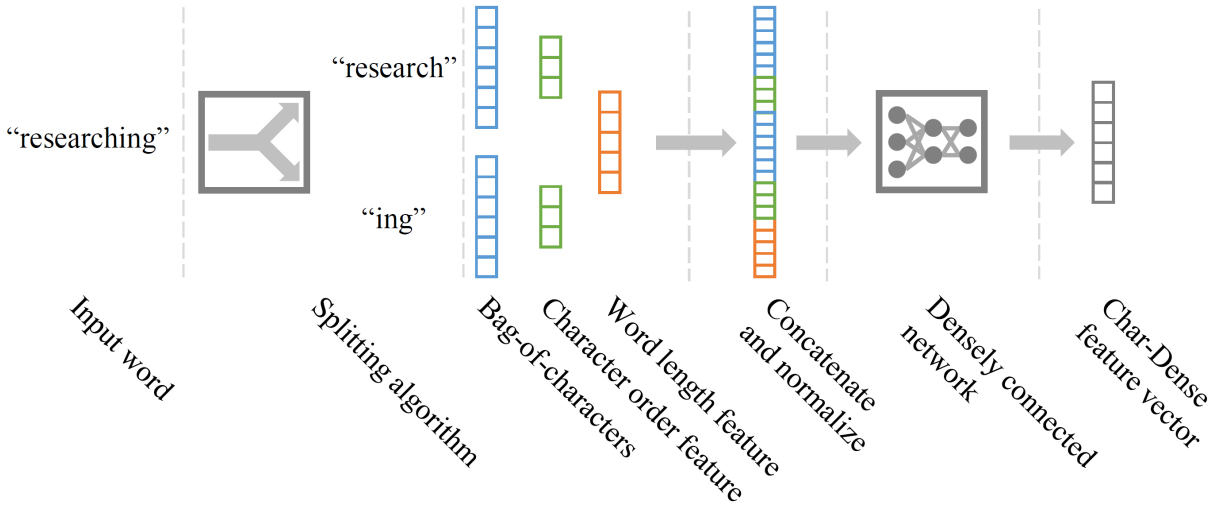


Figure 1: Process of generating the character-level feature vector of a word using the proposed method.

3.1 Splitting Words

Each word is split into k pieces to reduce the number of word collisions. To maintain the ordering of pieces, concatenation is used instead of summation or averaging to merge the vectors. Word splitting is done based on n-gram frequency. First, n-gram statistics C_{ng} is collected from the training corpus where $C_{ng}(s)$ is the number of times the n-gram s appears in the corpus. Then, the n-gram with the highest frequency gets merged into a single piece, and this merging is repeated until only k n-grams are left. The number of pieces k per word is a configurable hyperparameter. Finally, each piece is converted into a fixed length vector using BOC. The detailed algorithm is presented in Algorithm 1. This process is similar to the byte-pair encoding method in Sennrich et al. (2015), except that in the proposed method each word can only be split into k pieces whereas byte-pair encoding produces an arbitrary number of pieces. Producing a fixed number of pieces is important, since concatenation is used to merge the vectors.

Algorithm 1: Splitting word into k pieces

Input : word $w = (c_1, c_2, \cdots, c_n)$, n-gram statistics C_{ng}, number of pieces k
Output: $S = (s_1, \cdots, s_k)$ where $s_1 + s_2 + \cdots + s_k = w$

$S \leftarrow w$
while $|S| > k$ **do**
  $m = \underset{i}{\operatorname{argmax}}\, C_{ng}(c_i + c_{i+1})$
  $S \leftarrow (\cdots, s_{m-1}, s_m + s_{m+1}, s_{m+2}, \cdots)$
end
while $|S| < k$ **do**
  Append empty string to S
end
return S

3.2 Character Order Feature

Every character that has a digital representation can be converted into a numerical value via some character-set (e.g. UTF-16). Then, it is possible to numerically compare two characters. Let $T = \{c_1, c_2, \cdots, c_n\}$ be a character sequence of length n. Then $F_{asc}(T, k)$, $F_{des}(T, k)$, $C_{asc}(T)$, and $C_{des}(T)$ are defined as follows:

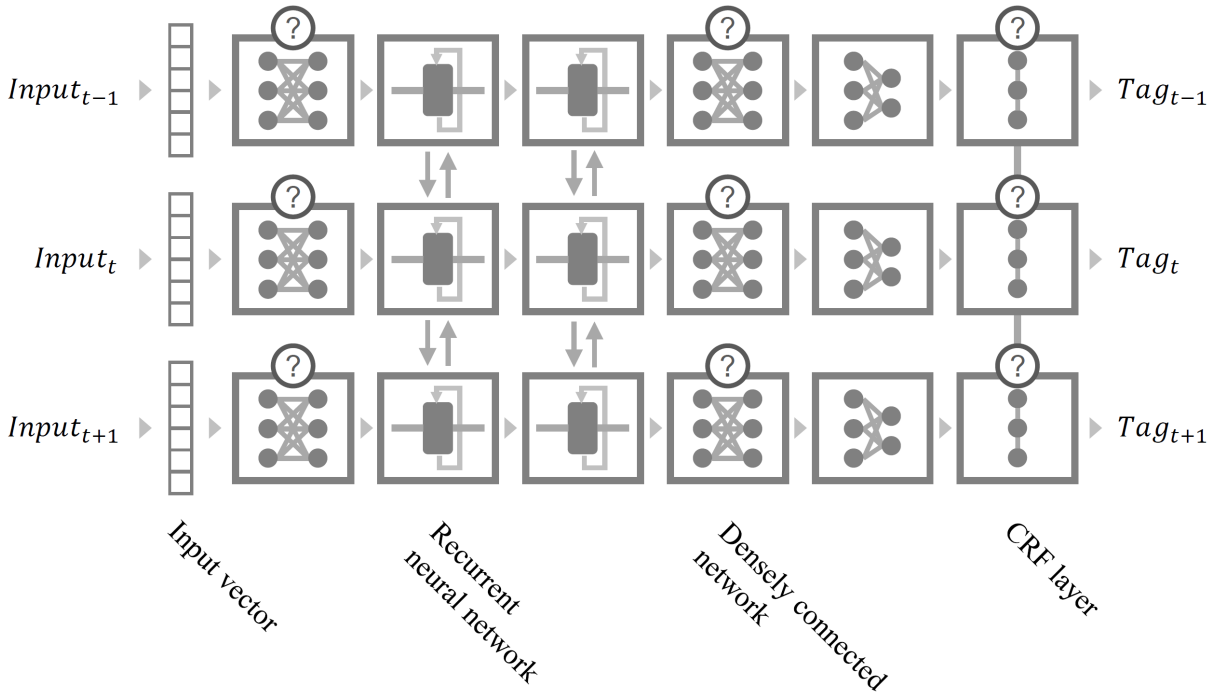


Figure 2: Overview of model architecture for sequence tagging experiments. Question mark indicates that the component is optional.

$$F_{asc}(T,k) = \begin{cases} 1, \text{if } c_k < c_{k+1} \\ 0, \text{otherwise} \end{cases}, \quad F_{des}(T,k) = \begin{cases} 1, \text{if } c_k > c_{k+1} \\ 0, \text{otherwise} \end{cases} \tag{2}$$

$$C_{asc}(T) = \sum_{k=1}^{n-1} F_{asc}(T,k), \quad C_{des}(T) = \sum_{k=1}^{n-1} F_{des}(T,k) \tag{3}$$

Bi-grams with the same character repeating are ignored. A sequence of characters can then be categorized into one of three classes: $C_{asc}(T) > C_{des}(T)$, $C_{asc}(T) = C_{des}(T)$, $C_{asc}(T) < C_{des}(T)$. This category info is calculated for each word piece, which is then converted into a 3-dimensional vector using one-hot encoding and concatenated to the sparse word piece vector.

3.3 Word Length Feature

To further reduce the number of word collisions, information about the word's length is added into the model. One-hot encoding is used to store an integer from 0 to 20, and any word exceeding 20 characters is treated as being 20 characters long.

4 Model

In this section, we describe the sequence tagging model's architecture in detail. Figure 2 illustrates the model architecture.

4.1 Sequence Tagging with Bidirectional RNN

In sequence tagging tasks, such as POS tagging or NER, both future and past input tokens are available to the model. Bidirectional RNNs (Graves and Schmidhuber, 2005) can efficiently make use of future and past features over a certain time frame. We use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for our RNN cell, which is better at capturing long-term dependencies than vanilla RNN. Output of the forward and backward RNN layers are summed to form the feature vector of each time-step. Each word is tagged based on this feature vector, using either a softmax layer or CRF layer.

To capture a more abstract and higher-level representation in different layers, a densely connected layer can be added before and after the Bi-LSTM layers. The input to this network at each time-step is the concatenation of the character-level feature vector and a pre-trained word vector (described in section 5).

4.2 Conditional Random Field

Even though a Bi-LSTM layer can efficiently extract features for each time-step utilizing past and future inputs, the prediction is made on each time-step, independent of past and future tag outputs. The Conditional Random Field (CRF) layer overcomes this limitation by considering state transition probability, thereby decoding the most probable output tag sequence. It has been shown that adding a CRF layer on top of a Bi-LSTM network can lead to statistically significant performance increases (Reimers and Gurevych, 2017). We also test a variant of our model using CRF as the final layer to perform tag sequence prediction.

4.3 Stacking RNNs with Residual Connection

Increasing the depth of the neural network architecture has proven to be an effective way of improving performance. However, naively stacking layers can lead to adversarial effects due to the degradation problem. Residual connection (He et al., 2016) has shown to be an effective way to tackle this issue by creating a shortcut between layers. The same strategy is adopted to our model when there are more than one Bi-LSTM layers, in which case the input is added to the Bi-LSTM layer's output.

4.4 Dropout

Dropout is a popular and effective way of regularizing neural network models, by randomly dropping nodes (Srivastava et al., 2014). In our model, Inverted dropout is applied to all densely connected layers for regularization. For the Bi-LSTM layers, variational recurrent dropout (Gal and Ghahramani, 2016) is used, since naive dropout can deteriorate performance. The word embedding matrix is regularized using the method proposed in Gal and Ghahramani (2016), i.e. dropping words at random.

5 Training Details

Pre-trained Word Embeddings Utilizing word embeddings pre-trained on large unlabeled text has shown to be one of the most effective ways to increase performance on various NLP tasks. Our model uses the GloVe (Pennington et al., 2014) 300-dimensional vectors trained on the Common Crawl corpus with 42B tokens as word level features, as this resulted in the best performance in preliminary experiments. Words that do not appear in the training data are replaced with a special Out-of-Vocabulary (OOV) token. To train the vector of this token, we randomly swap words with OOV tokens while training with a 0.01 probability, as in Lample et al. (2016). The word vector is then concatenated with the character-level feature vector and fed into the subsequent layer.

Freezing Embeddings It is common practice to fine-tune the pre-trained word vectors through the training process. However, preliminary experiments have revealed that fine-tuning the word vectors results in lower performance than freezing the vectors, especially in the early stages of training. We hypothesize that randomly initialized weights in the model act as noise and degrade the pre-trained word vectors. To circumvent this issue, the embeddings are frozen for the first T_{freeze} phase of training so that they are not affected by untrained weights. We use $T_{freeze} = 20\%$ for all experiments.

Dynamic Batch Size Keskar et al. (2016) showed that small batch sizes lead to more global and flat minimizers, while large batch sizes lead to more local and sharp minimizers. Therefore, starting from a small batch size and increasing it during training would result in a more global, but sharp minimizer. While having similar effect to learning rate decay, this strategy also has a benefit of accelerating the training as the batch size grows (Smith et al., 2017). Adopting this method, we start from a fixed initial batch size, and increase the batch size by a factor of two on each quarter of the course of training.

Tagging Scheme It is reported that more complicated tagging schemes such as IOBES does not have statistically significant advantage over BIO scheme (Reimers and Gurevych, 2017), thus we adopt the BIO scheme for all experiments.

Dataset	ATIS		PTB WSJ		CoNLL2003	
	Sentences	Tokens	Sentences	Tokens	Sentences	Tokens
Training	4978	56591	38219	912344	14987	204567
Develop.	-	-	5527	131768	3466	51578
Test	893	9198	5462	129654	3684	46666

Table 1: Corpus statistics of each task.

Parameter Optimization Our network is trained by minimizing the cross entropy loss over the tags for the softmax model, or maximizing the log-likelihood of the tag sequence for the CRF model. The objective function is optimized using the gradient-based optimization algorithm Adam (Kingma and Ba, 2014). For all experiments, we implement the model using the TensorFlow (Abadi et al., 2016) library.
Hyperparameter Tuning Most hyperparameters, with the following exceptions, are tuned on the development sets. Hyperparameters of the character-CNN and character-RNN models are adopted from Ma and Hovy (2016) and Lample et al. (2016), respectively. The chosen hyperparameters for all experiments are summarized in Appendix A.

6 Evaluation

We evaluate the effectiveness of the proposed method using three of the most well-studied and common English sequence tagging tasks - Slot tagging, POS tagging, and NER. Note that to test the generalizability of the proposed method, we do not perform any preprocessing for all experiments. Details on each task and baseline models are described in this section. Table 1 summarizes the statistics of each task.

6.1 Slot Tagging

For slot tagging, we use the Airline Travel Information System (ATIS) dataset. This dataset has 84 types of slot labels and 127 possible tags with BIO tagging scheme. Since this corpus lacks a development set, 20% of the training data is randomly sampled and used as the development set for tuning the hyperparameters. This task's performance is measured in F1-score, which is calculated using the publicly available *conlleval.pl* script.

6.2 Part-of-Speech Tagging

For POS tagging, we use the Wall Street Journal (WSJ) portion of the Penn TreeBank dataset (Marcus et al., 1993) and adopt the standard split for part-of-speech tagging experiments - section 0-18 as training data, section 19-21 as development data, and section 22-24 as test data. This dataset contains 45 different POS tags. Model performance is measured by token-level accuracy.

6.3 Named Entity Recognition

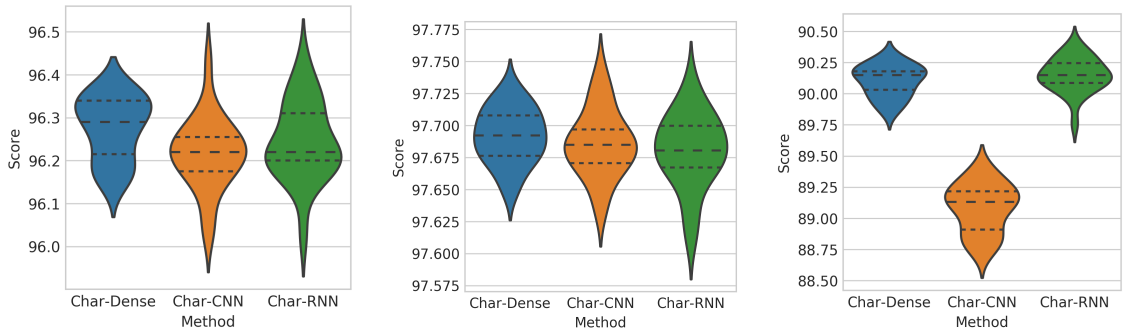
For NER, the English portion of the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003) is used for evaluation. This dataset contains four different types of named entities, which results in nine possible tags with BIO tagging scheme and an 'O' tag. Like slot tagging, the final performance is measured in F1-score using the same *conlleval.pl* script.

6.4 Baseline Models

Character-level CNN and character-level RNN are the most effective and widely adopted methods for character-level feature extraction, and thus are suitable as strong baseline methods. We implement these two methods to use them as baselines for comparison. The CRF layer has the effect of making the model robust to architectural differences (Reimers and Gurevych, 2017). Since the goal of baseline experiments is to evaluate the effect of difference in character-level feature generation methods, we use the softmax layer instead of the CRF layer for these experiments. Every aspect of the sequence tagging model except the character-level feature generation method is identical for all baseline experiments.

Method	Task		
	Slot	POS	NER
Char-CNN	96.22 (SD 0.08)	97.68 (SD 0.03)	89.08 (SD 0.20)
Char-RNN	96.25 (SD 0.09)	97.68 (SD 0.03)	**90.15** (SD 0.14)
Char-Dense (Ours)	**96.28** (SD **0.07**)	**97.69** (SD **0.02**)	90.10 (SD **0.13**)

Table 2: Comparison with baseline models.



(a) Slot (b) POS (c) NER

Figure 3: Score distributions for all experiments. Quartiles marked with dashed lines.

7 Results and Discussion

7.1 Experimental Results

For a more in-depth analysis of the performance of the proposed method and two baselines, we train each model 20 times with different initial parameters, which are randomly initialized (Reimers and Gurevych, 2017). Table 2 summarizes the mean performance with standard deviation in parentheses. Performance distribution is also visualized using a violin plot in Figure 3.

7.1.1 Slot Tagging

On the task of tagging semantic slots using the ATIS dataset, the proposed method shows the best results in terms of both performance and variability. Our method has the highest mean F1-score of 96.28. Furthermore, it has the lowest standard deviation across all runs, which means it is robust to parameter initialization. On the contrary, both CNN and RNN models have lower performance and higher variability compared to the proposed method.

Analyzing the violin plot reveals that there are also differences in score distribution. While CNN models tend to have a low F1-score on average with occasional high peaks, RNN models have higher F1-score in general but suffer from a large performance drop with poor parameter initialization. This could be one of the reasons why models using CNN seem to have superior performance when only the best performance is reported. On the other hand, our model does not result in peaks or serious drops in performance with different seed values, which makes it more suitable for real-world applications.

7.1.2 Part-of-Speech Tagging

The proposed method also achieves the best results on the POS tagging task. Similar to the slot tagging task, our method shows the highest mean accuracy of 97.69 with the lowest standard deviation of 0.02. For the baseline models, CNN and RNN performed on par.

CNN-based models have higher variability with high peak performance on this task also, as shown in the violin plot. Similar to the slot tagging task, our method shows the lowest variability, which supports the robustness of this method.

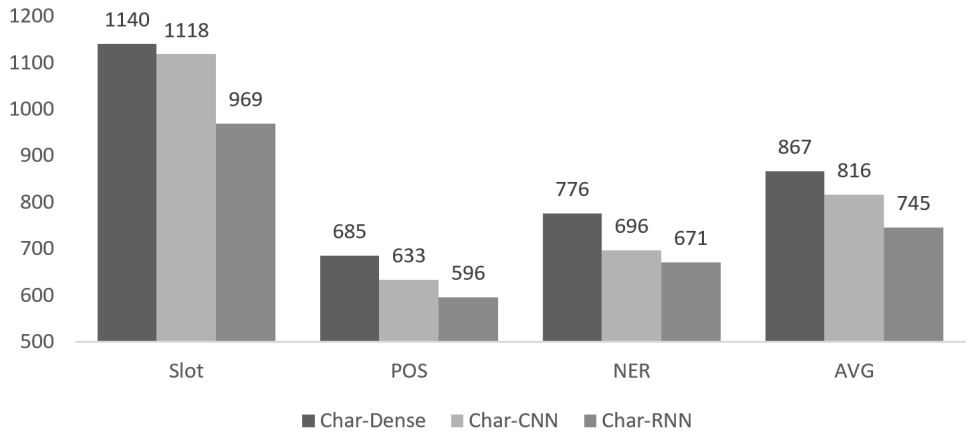


Figure 4: Sentence processing speed in terms of number of sentences per second.

Slot		**POS**		**NER**	
Approach	F1	Approach	Acc.	Approach	F1
Mesnil et al. (2015)	94.73	Toutanova et al. (2003)	97.24	Ando and Zhang (2005)	89.31
Yao et al. (2014)	94.85	Manning (2011)	97.32	Collobert et al. (2011)	89.59
Liu and Lane (2015)	94.89	Shen et al. (2007)	97.33	Huang et al. (2015)	90.10
Yao et al. (2014)	95.08	Sun (2014)	97.36	Chiu and Nichols (2015)	90.77
Peng and Yao (2015)	95.25	Moore (2015)	97.36	Ratinov and Roth (2009)	90.80
Vu et al. (2016)	95.56	Hajič et al. (2009)	97.44	Lin and Wu (2009)	90.90
Vu (2016)	95.61	Søgaard (2011)	97.50	Passos et al. (2014)	90.90
Kurata et al. (2016)	95.66	Tsuboi (2014)	97.51	Lample et al. (2016)	90.94
Zhu and Yu (2017)	95.79	Huang et al. (2015)	97.55	Luo et al. (2015)	91.20
Zhai et al. (2017)	95.86	Choi (2016)	97.64	Ma and Hovy (2016)	**91.21**
Char-Dense w/o CRF (Ours)	96.36	Char-Dense w/o CRF (Ours)	**97.73**	Char-Dense w/o CRF (Ours)	90.28
Char-Dense w/ CRF (Ours)	**96.62**	Char-Dense w/ CRF (Ours)	97.65	Char-Dense w/ CRF (Ours)	91.13

Table 3: Comparison with state-of-the-art approaches in the literature.

7.1.3 Named Entity Recognition

On the NER task, the RNN-based model has a slightly better F1-score (90.15) than the proposed method (90.10). However, our method consistently shows the lowest standard deviation, like as the other tasks, at 0.13. By analyzing the violin plot, we can see that the RNN again shows occasional performance drops for certain cases of poor weight initialization. Unlike the other two tasks, the model utilizing CNN has a relatively poor F1-score and does not show any peaks in performance.

7.2 Training Speed

To compare the efficiency of three models, average training speed (i.e. number of sentences processed per second) is presented in Figure 4. All trainings are performed utilizing a single GeForce GTX 1080 Ti GPU, and the RNN model is implemented using the highly efficient cuDNN LSTM API. It is clear that the proposed method has the highest training speed, followed by CNN and RNN. On average, our method was able to process around 867 sentences per second, which is 6.29% and 16.32% higher than CNN and RNN, respectively.

7.3 Comparison with Published Results

For comparison with published results, we summarize the performance of our best models along with state-of-the-art approaches in Table 3. The proposed method was able to surpass the previous state-of-the-art result on the ATIS dataset with a large margin, even without the CRF layer. With the help of CRF, our method obtains a new state-of-the-art result with a 96.62 F1-score.
For the POS tagging task with PTB WSJ dataset, we obtain a new state-of-the-art result with a 97.73% accuracy with the model without a CRF layer. Interestingly, utilizing a CRF layer on this model degraded the performance on this task whereas it helped with the other two tasks. We hypothesize that this is due to the fact that unlike the other two tasks where there are many hard constraints between labels (e.g. an O tag cannot be followed by I- tags), the label dependencies are more "soft" on POS tagging task. In the latter case, it is possible that naively taking label transition probability into account could have a negative impact on performance.
On the task of recognizing named entities, we obtain a result that is comparable to state-of-the-art with a 91.13 F1-score when a CRF layer is used. Like in slot tagging task, utilizing CRF lead to a significant increase in performance. It is notable that all results from our method are achieved without depending on any hand-crafted or language/task-specific features (e.g. capitalization, character type, gazetteer), whereas most previous approaches utilizes one or more type of such features. This fact supports the generalizability of the proposed method.

8 Conclusion and Future Work

In this paper, we proposed a fast and effective method of using a densely connected network to automatically generate character-level features. With extensive evaluation, it is shown that this method is robust to parameter initialization and has high processing speed compared to conventional methods such as CNN or RNN. This method has also high generalizability and this is supported by the fact that we were able to obtain superior performance without any task or language specific features.
We plan to explore the followings as future work: 1) In this work, we focused on clean text where there are minimal semantic or syntactic errors. We would like to test the robustness of this method against such errors to evaluate whether this method is suitable for real-world applications. 2) Adopting the proposed method and analyzing the effectiveness on other NLP tasks such as neural machine translation or automatic text summarization could also be worth investigating.

Acknowledgements

This research was supported by the MSIT (Ministry of Science and ICT), South Korea, under the ITRC (Information Technology Research Center) support program ("Research and Development of Human-Inspired Multiple Intelligence") supervised by the IITP (Institute for Information & Communications Technology Promotion). Additionally, this work was supported by the National Research Foundation of Korea (NRF) grant funded by the South Korean government (MSIP) (No. NRF-2016R1A2B2015912).

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

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

Jinho D Choi. 2016. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of NAACL-HLT*, pages 271–281.

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

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Jenny Finkel, Shipra Dingare, Christopher D Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: gene and protein identification in biomedical text. *BMC bioinformatics*, 6(1):S5.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Jan Hajič, Jan Raab, Miroslav Spousta, et al. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

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

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

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

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 171–189. Springer.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Robert Moore. 2015. An improved tag dictionary for faster part-of-speech tagging. In *Proc. of EMNLP*. Citeseer.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL*, volume 7, pages 760–767. Citeseer.

Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 48–52. Association for Computational Linguistics.

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):1929–1958.

Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 2402–2410.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *EMNLP*, pages 938–950.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6060–6064. IEEE.

Ngoc Thang Vu. 2016. Sequential convolutional neural networks for slot filling in spoken language understanding. *arXiv preprint arXiv:1606.07783*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194. IEEE.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking.

Su Zhu and Kai Yu. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5675–5679. IEEE.

Appendix A Hyperparameters

Group	Hyperparameter	Slot	POS	NER
Char-CNN	Window size	3	3	3
	Number of filters	30	30	30
	Character dimension	30	30	30
Char-RNN	Layer size	50	50	50
	Character dimension	50	50	50
Char-Dense	Layer size	50	50	50
	Number of pieces per word	2	2	2
Word-level	Pre-trained word embeddings	GloVe 300d	GloVe 300d	GloVe 300d
	RNN layer size	350	350	350
	RNN layer depth	2	3	3
	Pre-RNN layer size	350	350	None
	Post-RNN layer size	350	350	None
Dropout keep probability	Char-Dense	0.7	0.7	0.7
	Word feature	0.9	0.9	0.9
	Word-level RNN layer	0.5	0.5	0.5
	Pre/post-RNN layers	0.5	0.5	0.5
Training	Initial batch size	8	16	16
	Number of epochs	100	100	100

Table 4: Chosen hyperparameters for all experiments.