Learning Better Internal Structure of Words for Sequence Labeling

Yingwei Xin, Ethan Hart, Vibhuti Mahajan, Jean-David Ruvini
eBay Research
2025 Hamilton Ave, San Jose, CA 95125, USA
{yixin,ejhart,vibmahajan,jean-david.ruvini}@ebay.com

Abstract

Character-based neural models have recently proven very useful for many NLP tasks. However, there is a gap of sophistication between methods for learning representations of sentences and words. While, most character models for learning representations of sentences are deep and complex, models for learning representations of words are shallow and simple. Also, in spite of considerable research on learning character embeddings, it is still not clear which kind of architecture is the best for capturing character-to-word representations. To address these questions, we first investigate the gaps between methods for learning word and sentence representations. We conduct detailed experiments and comparisons on different state-of-the-art convolutional models, and also investigate the advantages and disadvantages of their constituents. Furthermore, we propose IntNet, a funnel-shaped wide convolutional neural architecture with no down-sampling for learning representations of the internal structure of words by composing their characters from limited, supervised training corpora. We evaluate our proposed model on six sequence labeling datasets, including named entity recognition, part-of-speech tagging, and syntactic chunking. Our in-depth analysis shows that IntNet significantly outperforms other character embedding models and obtains new state-of-the-art performance without relying on any external knowledge or resources.

1 Introduction

Sequence labeling is the task of assigning a label or class to each element of a sequence of data, and is one of the first stages in many natural language processing (NLP) tasks. For example, named entity recognition (NER) aims to classify words in a sentence into several predefined categories of interest such as person, organization, location, etc. Part-of-speech (POS) tagging assigns a part of speech to each word in an input sentence. Syntactic chunking divides text into syntactically related, non-overlapping groups of words. Sequence labeling is a challenging problem because human annotation is very expensive and typically only a small amount of tagging data is available.

Most traditional sequence labeling systems have been dominated by linear statistical models which heavily rely on feature engineering. As a result, carefully constructed hand-crafted features and domain-specific knowledge are widely used for solving these tasks. Unfortunately, it is costly to develop domain specific knowledge and hand-crafted features. Recently, neural networks using character-level information have been used successfully for minimizing the need of feature engineering. There are basically two threads of character-based modeling, one focuses on learning representations of sentences for semantics and syntax (Zhang et al., 2015; Conneau et al., 2017); the other focuses on learning representations of words for the purpose of eliminating hand-crafted features for word shape information (Lample et al., 2016; Ma and Hovy, 2016).

Two main state-of-the-art approaches of learning character representations for sequence labeling emerged from the latter thread. One is based on RNNs and uses bidirectional LSTMs or GRUs to learn forward and backward character information (Ling et al., 2015; Lample et al., 2016; Yang et al., 2017). The other approach is based on CNNs with a fixed-size window around each word to create character-level representations (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016). However, there is a gap in the sophistication between character-based methods for learning representations of sentences compared to that of words. We found that most of the state-of-the-art character-based CNN models for words

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2584–2593
Brussels, Belgium, October 31 - November 4, 2018.

use a convolution followed by max pooling as a shallow feature extractor, which is very different from the CNN models with deep and complex architecture for sentences. In spite of considerable research on learning character embeddings, it is still not clear which kind of architecture is the best for capturing character-to-word representations.

Therefore, a number of questions remain open:

- Why is there a gap between methods for learning representations of sentences and words? How can this gap be bridged?
- How do state-of-the-art character embedding models differ in term of performance?
- What kind of neural network architecture is better for learning the internal structure of a word? Deep or shallow? Narrow or wide?

To answer these questions, we first investigate the gap between learning word representations and sentence representations for convolutional architectures. The most straightforward idea is to add more convolutional layers which follows the approaches from learning representations of sentences. Interestingly, we observe the accuracy does not increase much and found that accuracy drops when we increased the depth of the network. This observation shows that learning character representations for the internal structure of words is very different than sentences, and also might explain one of the reasons there has been a gap in character-based CNN models for representing words and sentences.

In this paper, we present detailed experiments and comparisons across different state-of-the-art convolutional models from natural language processing and computer vision. We also investigate the advantages and disadvantages of some of their constituents on different convolutional architectures. Furthermore, we propose IntNet, a funnel-shaped wide convolutional neural network for learning the internal structure of words by composing their characters. Unlike previous CNN-based approaches, our funnel-shaped IntNet explores deeper and wider architecture with no down-sampling for learning character-to-word representations from limited supervised training corpora. Lastly, we combine our IntNet model with LSTM-CRF, which captures both word shape and context information, and jointly decode tags for sequence labeling.

The main contributions of this paper are the following:

- We conduct detailed studies on investigating the gap between learning word representations and sentence representations.
- We provide in-depth experiments and empirical comparisons of different convolutional models and explore the advantages and disadvantages of their components for learning character-to-word representations.
- We propose a funnel-shaped wide convolutional neural architecture with no down-sampling that focuses on learning a better internal structure of words.
- Our proposed compositional character-to-word model combined with LSTM-CRF achieves state-of-the-art performance for various sequence labeling tasks.

This paper is organized as follows: Section 2 describes multiple threads of related work. Section 3 presents the whole architecture of the neural network. Section 4 provides details about experimental settings and compared methods. Section 5 reports model results on different benchmarks with detailed analyses and discussion.

2 Related Work

There exist three threads of related work regarding the topic of this paper: (i) different convolutional architectures from different domains; (ii) character embedding models for words; (iii) sequence labeling with deep neural network.

CNN models across domains. Convolutional neural networks (CNNs) are very useful in extracting information from raw signals. In the area of NLP, Kim (2014) was the first to propose shallow CNN with word embeddings for sentence classification. Zhang et al. (2015) proposed CNN with 6 convolutional layers by directly extracting character level information for learning representations of semantic structure on sentences. Recently, Conneau et al. (2017) proposed a VDCNN architecture with 29 convolutional layers using residual connections for text classification. However, one study on randomly dropping layers for training deep residual networks, (Huang et al., 2016), has shown that not all layers may be needed and highlighted there is some amount of redundancy in

ResNet (He et al., 2016). Also, some research has shown promising results with wide architectures, for example, wide ResNet (Zagoruyko and Komodakis, 2016), Inception-ResNet (Szegedy et al., 2017) and DenseNet (Huang et al., 2017). These models use character-level information to learn representations are for sentences, not words.

Character embedding models. Santos and Zadrozny (2014) proposed a CNN model to learn character representations of words to replace hand-crafted features for part-of-speech tagging. Ling et al. (2015) proposed a bidirectional LSTM over characters to use as input for learning character-to-word representations. Chiu and Nichols (2016) proposed a bidirectional LSTM-CNN with lexicons for named entity recognition by applying the CNN-based character embedding model from Santos and Zadrozny (2014). Plank et al. (2016) proposed a bi-LSTM model with auxiliary loss for multilingual part-of-speech tagging by following the LSTM-based character embedding model from Ling et al. (2015). Cotterell and Heigold (2017) proposed a character-level transfer learning model for neural morphological tagging.

Sequence labeling. Collobert et al. (2011) first proposed a method based on CNN-CRF that learns important features from words and requires few hand-crafted features. Huang et al. (2015) proposed a bidirectional LSTM-CRF model by using word embeddings and hand-crafted features for sequence tagging. Lample et al. (2016) applied the LSTM-based character embedding model from Ling et al. (2015) with bidirectional LSTM-CRF and obtained best results on NER for Spanish, Dutch, and German. Ma and Hovy (2016) applied the CNN-based character embedding model from Chiu and Nichols (2016), but without using any data preprocessing or external knowledge and achieved the best result on NER for English and part-of-speech tagging. Also, there have been some joint models which use additional knowledge, like transfer learning (Yang et al., 2017), pre-trained language models (Peters et al., 2017), language model joint training (Rei, 2017), and multi-task learning (Liu et al., 2018). Without any additional supervision or extra resources, LSTM-CRF (Lample et al., 2016) and LSTM-CNN-CRF (Ma and Hovy, 2016) are current state-of-the-art methods. To test the effectiveness of our proposed model, we use these two models as our baselines in the latter sections.

3 Neural Network Architecture

3.1 IntNet

Character embeddings. The first step is to initialize the character embeddings for each word w in the input sequence. We define the finite set of characters V^{char}. This vocabulary contains all the variations of the raw text, including uppercase and lowercase letters, numbers, punctuation marks, and symbols. Unlike some character-based approaches, we do not use any character-level prepossessing which enables our model to learn and capture regularities from prefixes to suffixes to construct character-to-word representations. The input word w is decomposed into a sequence of characters $\{c_1, ..., c_n\}$, where n is the length of w. Character embeddings are encoded by column vectors in the embedding matrix $W^{char} \in \mathbb{R}^{d^{char} \times |V^{char}|}$, where d^{char} is the number of parameters for each character in V^{char}. Given a character c_i, its embedding r_i^{char} is obtained by the matrix-vector product:

$$r_i^{char} = W^{char} v_i^{char}, \tag{1}$$

where v_i^{char} is defined as a one-hot vector for c_i. We randomly initialize a look-up table with values drawn from a uniform distribution with range $[-\sqrt{\frac{3}{d^{char}}}, +\sqrt{\frac{3}{d^{char}}}]$, where d^{char} is empirically chosen by users. The character set includes all unique characters and the special tokens `PADDING` and `UNKNOWN`. We do not perform any character-level preprocessing, including case normalization, digit replacement (e.g. replacing all sequences of digits 0-9 with a single "0"), nor do we use any capitalization features (e.g. `allCaps`, `upperInitial`, `lowercase`, `mixedCaps`, `noinfo`).

Convolutional blocks. The input for the IntNet is the sequence of character embeddings $\{r_1^{char}, ..., r_n^{char}\}$. First is the initial convolutional layer, which is a temporal convolutional module that computes 1-D convolutions. Let $\mathbf{x}_i \in \mathbb{R}^{d^{char} \times r^{char}}$ be the concatenation of the character embeddings for each w. The initial convolutional layer applies a matrix-vector operation to each successive window of size k^{char}. An input k-grams $\mathbf{x}_{i:i+k-1}$ is transformed through a convolution filter $\mathbf{w}_c$:

$$\mathbf{c}_i = f(\mathbf{w}_c \cdot \mathbf{x}_{i:i+k-1} + b_c), \tag{2}$$

where $\mathbf{c}_i$ is the feature map of 1-D convolution, f

is the non-linear ReLU function, and b_c is a bias term. Equation 2 produces m filters with different kernel sizes. The filters are computed with different kernels by the initial convolutional layers are concatenated:

$$\mathbf{g}_0 = [c_1^{k_1} \dots c_m^{k_1}; c_1^{k_2} \dots c_m^{k_2}; c_1^{k_h} \dots {}^{k_h}_m], \quad (3)$$

where h is the number of kernels, $\mathbf{g}_0$ is the output for the initial convolutional layer which feeds into the next convolutional block.

We define $\mathcal{F}(\cdot)$ as a function of several consecutive operations within a convolutional block. Firstly, a N×1 convolution transforms the input. The output size is $4 \times m \times h$ feature maps, like a bottleneck layer. The next step consists of multiple 1-D convolutions with kernels of different sizes. Lastly, we concatenate all the feature maps from kernels of different size. In each convolution, we use a batch normalization, followed by a ReLU activation and N×k temporal convolution.

Funnel-shaped wide architecture. The network comprises of L convolutional layers, which implies $(\frac{L-1}{2})$ convolutional blocks. We use direct connections from every other layer to all subsequent layers, inspired by dense connection. Therefore, the l^{th} layer has access to the feature maps of all the alternate layers:

$$\mathbf{g}_l = \mathcal{F}_l([\mathbf{g}_0, \mathbf{g}_2, \dots, \mathbf{g}_{l-2}]). \quad (4)$$

Equation 4 ensures maximum information flow between blocks in the network. Compared to residual connection $\mathcal{F}_l(\mathbf{g}_{l-1}) + \mathbf{g}_{l-1}$, it can be viewed as an extreme case of residual connection and makes feature reuse possible. Unlike DenseNet and ResNet, we concatenate feature maps by different kernels in every other convolutional layers, which captures different levels of features and makes our wide architecture possible, inspired by Inception. Different levels of concatenation can help IntNet to learn different patterns of word shape information. We compare our architecture to residual connection and dense connection for learning character-to-word representations in Section 5.

Without down-sampling. Compared to other CNN models like ResNet and DenseNet, our model does not contain any halve down-sampling layer or average pooling to reduce resolution. We did not find these operations to be helpful and, in some cases, found them to be detrimental to performance. These operations are useful for sentences and images, but might break the internal structure of words, like the sequential patterns for prefixes and suffixes.

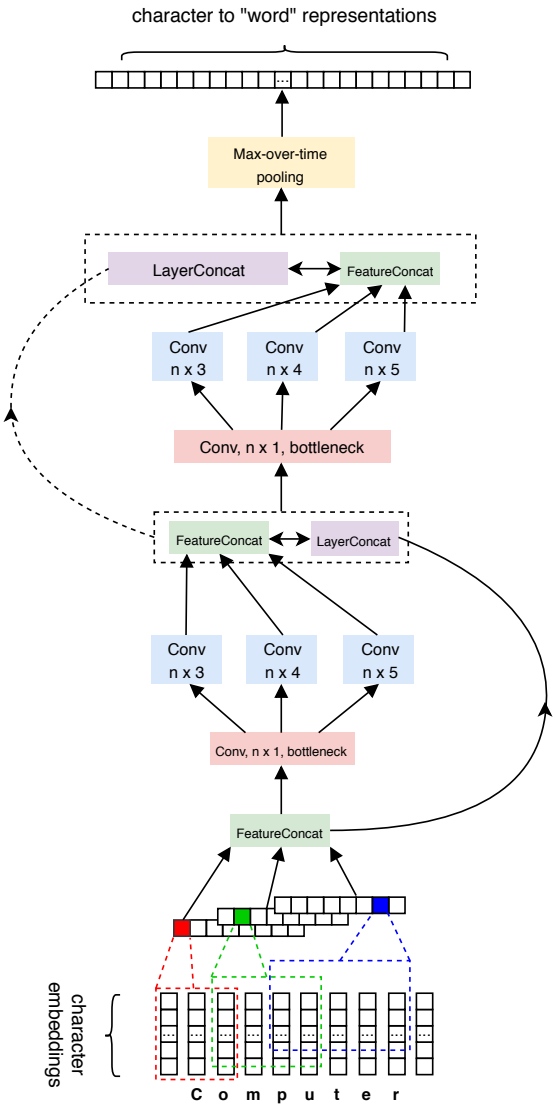

Figure 1: The main architecture of IntNet.

Character-to-word representations. In the last layer, we use a max-over-time pooling operation:

$$\hat{\mathbf{c}}_i = max(\mathbf{c}_i), \quad (5)$$

which takes the maximum value corresponding to a particular filter. The idea is to capture the most important feature with the highest value for each feature map. Finally, we concatenate all of salient features together as a representation for this word:

$$\mathbf{z} = [\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \dots \hat{\mathbf{c}}_u], \quad (6)$$

where u is the number of salient features which is equal to the total number of output feature maps in the last layer. If each function $\mathcal{F}_l$ produces p feature maps, we obtain $(p_0 + p \times \frac{L-1}{2})$ representations, where p_0 is the number of output feature maps in the initial convolution layer.

3.2 Bi-directional RNN

Given the character-to-word representations are computed by IntNet in Equation 6, we denote the input vector $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$ for a sentence. LSTM (Hochreiter and Schmidhuber, 1997) returns the sequence $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$ that represents the sequential information at every step. We use the following implementation:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{zi}\mathbf{z}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{zf}\mathbf{z}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\widetilde{\mathbf{c}}_t &= tanh(\mathbf{W}_{zc}\mathbf{z}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t),
\end{aligned}
$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. $\mathbf{z}_t$ is the input vector at time t and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ are the input gate, forget gate, output gate, and cell vectors, all of which are the same size as the hidden vector $\mathbf{h}_t$. $\mathbf{W}_{zi}$, $\mathbf{W}_{zf}$, $\mathbf{W}_{zo}$, $\mathbf{W}_{zc}$ denote the weight matrices of different gates for input $\mathbf{z}_t$; $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho}$, $\mathbf{W}_{hc}$ are the weight matrices for hidden state $\mathbf{h}_t$, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, $\mathbf{b}_c$ denote the bias vectors. Forward LSTM and backward LSTM compute the representations of $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ for left and right context of the sentence, respectively. We concatenate two hidden states to form the output of bi-directional LSTM $[\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}]$ for capturing context information from both sides.

3.3 Scoring Function

Instead of predicting each label independently, we consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence by leveraging a conditional random field (Lafferty et al., 2001). Formally, the sequence of labels is defined as:

$$\mathbf{y} = (y_1, y_2, ..., y_T). \tag{7}$$

To define the scoring function $f(\mathbf{h}, \mathbf{y})$ for each position t, we multiply the hidden state $\mathbf{h}_t^w$ with a parameter vector $\mathbf{w}_{y_t}$ that is indexed by the tag y_t to obtain the matrix of scores output by the bidirectional LSTM network. Therefore, the function f can be written as:

$$f(\mathbf{h}, \mathbf{y}) = \sum_{t=1}^{T} \mathbf{w}_{y_t}\mathbf{h}_t^w + \sum_{t=1}^{T} \mathbf{A}_{y_{t-1}, y_t}. \tag{8}$$

In Equation 8, $\mathbf{A}$ is a matrix of transition scores, $\mathbf{A}_{i,j}$ represents the score of a transition from the tag i to tag j, y_1 is the start tag of a sentence. Let $\mathcal{Y}(\mathbf{h})$ denote the set of possible label sequences for $\mathbf{h}$. A probabilistic model for a sequence defines a family of conditional probabilities $p(\mathbf{y}|\mathbf{h})$ over all possible label sequences $\mathbf{y}$ given $\mathbf{h}$ with the following form:

$$p(\mathbf{y}|\mathbf{h}) = \frac{e^{f(\mathbf{h},\mathbf{y})}}{\sum_{y' \in \mathcal{Y}(\mathbf{h})} e^{f(\mathbf{h},\mathbf{y}')}}. \tag{9}$$

3.4 Objective Function and Inference

For end-to-end network training, we use maximum conditional likelihood estimation to maximize the log probability of the correct tag sequence:

$$log(p(\mathbf{y}|\mathbf{h})) = f(\mathbf{h}, \mathbf{y}) - log\left(\sum_{y' \in \mathcal{Y}(\mathbf{h})} e^{f(\mathbf{h},\mathbf{y}')}\right).$$

While decoding, we predict the label sequence that obtains the highest score given by:

$$\mathbf{y}^* = \arg\max_{y' \in \mathcal{Y}(\mathbf{h})} f(\mathbf{h}, \mathbf{y}'). \tag{10}$$

The objective function and its gradients can be efficiently computed by dynamic programming; for inference, we use the Viterbi algorithm to find the best tag path which maximizes the score.

4 Experiments

4.1 Datasets

We performed experiments on six standard datasets for sequence labeling tasks, i.e. named entity recognition, part-of-speech tagging, and syntactic chunking. To test the effectiveness of our proposed model, we do not use language-specific resources (such as gazetteers), external knowledge

Model	Spanish NER	Dutch NER	English NER	German NER	Chunking	PTB POS
Baseline	70.73±0.42	63.49±0.42	77.51±0.39	54.07±0.42	91.97±0.21	95.76±0.13
+ char-LSTM	79.93±0.43	77.16±0.47	83.98±0.46	64.29±0.47	93.31±0.23	97.14±0.11
+ char-CNN	79.78±0.41	76.43±0.48	83.85±0.38	63.53±0.41	92.67±0.24	97.02±0.12
+ char-CNN-5	79.63±0.38	**76.92**±0.42	83.60±0.39	**64.26**±0.42	**93.11**±0.26	**97.15**±0.12
+ char-CNN-9	79.25±0.56	74.82±0.46	83.31±0.47	63.97±0.46	**92.92**±0.27	**97.13**±0.13
+ char-ResNet-9	74.34±0.45	**76.54**±0.39	**83.91**±0.42	**66.15**±0.44	**93.85**±0.24	96.99±0.15
+ char-DenseNet-9	78.25±0.52	**76.71**±0.53	**84.16**±0.41	**67.54**±0.46	**93.82**±0.25	**97.13**±0.11
+ char-IntNet-9	78.53±0.44	**76.93**±0.47	83.83±0.44	**70.11**±0.41	**93.94**±0.26	**97.19**±0.12
+ char-IntNet-5	**80.44**±0.43	**78.06**±0.45	**85.34**±0.39	**69.48**±0.42	**94.27**±0.23	**97.23**±0.11

Table 1: F1 score of different character-to-word models.

(such as transfer learning, joint training), handcrafted features, or any character preprocessing, we do not replace any rare words into `UNKNOWN`.

Named entity recognition. CoNLL-2002 and CoNLL2003 datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) contain named entity labels for Spanish, Dutch, English and German as separate datasets. These four datasets contain different types of named entities: locations, persons, organizations, and miscellaneous entities. Unlike some approaches, we do not combine the validation set with the training set. Although POS tags were made available for these datasets, we do not leverage those as additional information which sets our approach apart from that of transfer learning.

Part-of-speech tagging. The Wall Street Journal (WSJ) portion of Penn Treebank (PTB) (Marcus et al., 1993) contains 25 sections and categorizes each word into one out of 45 POS tags. We adopt the standard split and use sections 0-18 as training data, sections 19-21 as development data, and sections 22-24 as test data.

Syntactic chunking. The CoNLL 2000 chunking task (Tjong Kim Sang and Buchholz, 2000) uses sections 15-18 from the Wall Street Journal corpus for training and section 20 for testing. It defines 11 syntactic chunk types (e.g., NP, VP, ADJP), we adopt the standard split and sample 1000 sentences from the training set as the development set.

4.2 Training Settings

Initialization. The size of the dimensions of character embeddings is 32 which are randomly initialized using a uniform distribution. We adopt the same initialization method for randomly initialized word embeddings that are updated during training. For IntNet, the filter size of the initial convolution is 32 and that of other convolutions is 16. We have used filters of size $[3, 4, 5]$ for all the kernels. The number of convolutional layers are 5 and 9 for IntNet-5 and IntNet-9, respectively, and we have adopted the same weight initialization as that of ResNet. We use pre-trained word embeddings for initialization, GloVe (Pennington et al., 2014) 100-dimension word embeddings for English, and fastText (Bojanowski et al., 2017) 300-dimension word embeddings for Spanish, Dutch, and German. The state size of the bi-directional LSTMs is set to 256. We adopt standard BIOES tagging scheme for NER and Chunking.

Optimization. We employ mini-batch stochastic gradient descent with momentum. The batch size, momentum and learning rate are set to 10, 0.9 and $\eta_t = \frac{\eta_0}{1+\rho t}$, where η_0 is the initial learning rate 0.01 and $\rho = 0.05$ is the decay ratio, the value of gradient clipping is 5. Dropout is applied on the input of IntNet, LSTMs, and CRF, and its ratio 0.5 is fixed, but with no dropout inside of IntNet.

4.3 Compared Methods

To address those open questions in Section 1, we conduct detailed experiments and empirical comparisons on different state-of-the-art character embedding models across different domains. Firstly, we use LSTM-CRF with randomly initialized word embeddings as our initial baseline. We adopt two state-of-the-art methods in sequence labeling, denoted as char-LSTM (Lample et al., 2016) and char-CNN (Ma and Hovy, 2016). We add more layers to the char-CNN model and refer to that as char-CNN-5 and char-CNN-9, respectively for 5 and 9 convolutional layers. Furthermore, we add residual connections to the char-CNN-9 and refer it as char-ResNet. Also, we apply 3 dense blocks based on char-ResNet which we refer to as char-DenseNet, to compare the difference between residual connection and dense connection. Lastly, we refer to our proposed

Model	Spanish	Dutch	English	German	Chunking	POS
Conv-CRF+Lexicon (Collobert et al., 2011)	-	-	89.59	-	94.32	97.29
LSTM-CRF+Lexicon (Huang et al., 2015)	-	-	90.10	-	94.46	97.43
LSTM-CRF+Lexicon+char-CNN (Chiu and Nichols, 2016)	-	-	90.77	-	-	-
LSTM-Softmax+char-LSTM (Ling et al., 2015)	-	-	-	-	-	97.55
LSTM-CRF+char-LSTM (Lample et al., 2016)	85.75	81.74	90.94	78.76	-	-
LSTM-CRF+char-CNN (Ma and Hovy, 2016)	-	-	91.21	-	-	97.55
GRM-CRF+char-GRU (Yang et al., 2017)	84.69	85.00	91.20	-	94.66	97.55
LSTM-CRF	80.33±0.37	79.87±0.28	88.41±0.22	73.42±0.39	94.29±0.11	96.63±0.08
LSTM-CRF+char-LSTM	86.12±0.34	87.13±0.25	91.13±0.15	78.31±0.35	94.97±0.09	97.49±0.04
LSTM-CRF+char-CNN	85.91±0.38	86.69±0.22	91.11±0.14	78.15±0.31	94.91±0.08	97.45±0.03
LSTM-CRF+char-IntNet-9	85.71±0.39	**87.38**±0.27	**91.39**±0.16	**79.43**±0.33	**95.08**±0.07	97.51±0.04
LSTM-CRF+char-IntNet-5	**86.68**±0.35	**87.81**±0.24	**91.64**±0.17	78.58±0.32	**95.29**±0.08	**97.58**±0.02

Table 2: F1 score of our proposed models in comparison with state-of-the-art results.

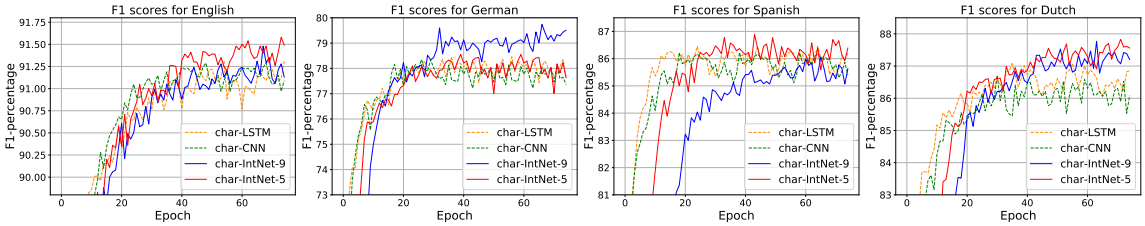

Figure 2: Training details of different models for English, German, Spanish, and Dutch.

model, which uses different convolution layers, as char-IntNet-5 and char-IntNet-9.

5 Results and Analysis

5.1 Character-to-word Models

Table 1 presents the performance of different character-to-word models on six benchmark datasets. For sequence labeling, char-LSTM and char-CNN are current state-of-the-art character embedding models for learning character-to-word representations. We observe that char-LSTM performs better than char-CNN in most cases, however, char-CNN uses a convolution layer followed by max pooling as a shallow feature extractor, that does not explore the full potential of CNNs.

Therefore, we implement two variations based on char-CNN, referred to as char-CNN-5 and char-CNN-9. The result shows that for most of the datasets, the F1 score does not improve much when we directly add more layers. We also observe some accuracy drop when we continuously increase the depth. This confirms why most CNN-based approaches for learning representations on words are shallow, which is very different from learning representations for sentences. Furthermore, we add residual connections to char-CNN-9 as char-ResNet-9, which confirms that residual connections can help train deep layers. We further improve char-ResNet-9 by changing residual connections into dense connection blocks as char-DenseNet-9, which shows that the dense connections are better than residual connections for learning word shape information.

Our proposed character-to-word model, char-IntNet-5 and char-IntNet-9 generally improves the results across all datasets. Our IntNet significantly outperforms other character embedding models, for example, the improvement is more than 2% in terms of F1 score for German and Dutch. Also, we observe that char-IntNet-5 is more effective for learning character-to-word representations than char-IntNet-9 in most of the cases. The only exception is German which seems to require a deeper and wider model for learning better representations.

5.2 State-of-the-art Results

Table 2 presents our proposed model in comparison with state-of-the-art results. LSTM-CRF is our baseline which uses fine-tuned pre-trained word embeddings. Its comparison with LSTM-CRF using random initializations for word embeddings, as shown in Table 1, confirms that pre-trained word embeddings are useful for sequence labeling. Since the training corpus for sequence labeling is relatively small, pre-trained embeddings learned from a huge unlabeled corpus can help to enhance word semantics. Furthermore, we adopt and re-implement two state-of-the-art character models, char-LSTM and char-CNN, by combining with LSTM-CRF, which we

	Model	English				German				Spanish				Dutch			
		IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
Dev	char-LSTM	97.15	89.87	89.41	87.07	86.97	85.80	68.35	64.76	89.63	89.06	78.14	74.13	94.50	87.98	80.00	72.37
	char-CNN	97.10	90.04	95.45	88.02	87.45	86.13	57.14	63.28	88.93	88.85	72.90	71.96	94.54	87.27	74.55	68.77
	char-IntNet-9	96.86	**90.52**	**91.95**	**90.16**	**87.92**	85.29	**76.07**	**67.98**	88.43	**88.58**	74.53	**72.09**	93.68	87.49	**89.09**	**75.58**
	char-IntNet-5	96.65	**90.14**	88.10	**88.31**	87.21	85.00	67.10	64.17	88.56	88.47	**78.90**	70.23	**94.63**	**88.56**	**89.09**	**74.40**
Test	char-LSTM	93.68	92.48	100.00	82.64	86.97	83.95	69.67	62.74	87.19	87.79	95.29	76.01	95.13	83.00	78.26	72.34
	char-CNN	93.85	92.65	100.00	84.09	64.72	83.67	69.67	58.19	87.81	88.46	87.96	73.68	94.25	82.50	73.27	73.37
	char-IntNet-9	93.79	**94.94**	100.00	82.31	**87.56**	**83.85**	**74.33**	**65.75**	87.08	87.98	**95.29**	**77.16**	94.42	**83.85**	**85.02**	**75.46**
	char-IntNet-5	**93.94**	**92.72**	100.00	**83.91**	**87.11**	83.60	67.22	60.92	87.19	**88.42**	**97.38**	**78.02**	94.71	**84.84**	**82.13**	**76.99**

Table 3: F1 score of different models for IV, OOTV, OOEV and OOBV.

Model	Frequent Words			Rare Words			OOV Words		
	newspapers	*slipped*	*world*	*Commerce*	*youthful*	*sessions*	11-*month*	*Thursdays*	*undetermined*
char-LSTM	enclosures	stirred	wolrd	Committee	luthier	cessions	19-month	Thousands	undereducated
	nelsonville	clipped	worde	Computer	loughmoe	sensible	10-month	Tunbridge	underpinned
	entrances	snipped	lowed	Comments	wrathful	stepanos	12-month	Standings	undermined
	newpapers	striped	wowed	Corrects	slothful	stefanos	14-month	Torrance	underlined
	necklaces	stifled	crowd	Clippers	ephorus	constans	11-inch	Phillies	underprepared
char-CNN	newspaper	slipper	worli	Committee	mouthful	suppressions	31-month	Thursday	determined
	newspapermen	slippy	worle	Community	eeyou	oppressions	51-month	Wednesday	overdetermined
	newpapers	stripped	worse	Commodities	mouthfeel	digressions	1-month	Tuesday	determinist
	nitrification	shipped	werle	Communist	motul	confessions	21-month	Ecuador	determiners
	megaphones	stopped	wereld	Comments	yourself	fissions	41-month	Windass	determiner
char-IntNet	newpapers	blipped	eworld	Commissioner	mouthful	recessions	55-month	Thursday	undermined
	wallpapers	unclipped	offworld	Commodities	mirthful	accessions	51-month	Saturday	determined
	escapers	tripped	homeworld	Clarence	mouthfuls	missions	22-month	thursdays	overdetermined
	carcases	dripped	linuxworld	Commission	youths	conversions	25-month	Tuesday	unexamined
	spacers	slopped	westworld	Commons	slothful	possessions	12-month	tuesdays	predetermined

Table 4: Nearest neighbours of different models for frequent words, rare words and OOV words.

refer to as LSTM-CRF-char-LSTM and LSTM-CRF-char-CNN. Lastly, we combine our proposed model with LSTM-CRF which we refer to as LSTM-CRF-char-IntNet-9 and LSTM-CRF-char-IntNet-5.

These experiments show that our char-IntNet generally improves results across different models and datasets. The improvement is more pronounced for non-English datasets, for example, IntNet improves the F-1 score over the state-of-the-art results by more than 2% for Dutch and Spanish. It also shows that the results of LSTM-CRF are significantly improved after adding character-to-word models, which confirms that word shape information is very important for sequence labeling. Figure 2 presents the details of training epochs in comparison with other state-of-the-art character models for different languages. It shows that char-CNN and char-LSTM converge early whereas char-IntNet takes more epochs to converge and generally performs better. It alludes to the fact that IntNet is suitable for reducing overfitting, since we have used early stopping while training.

5.3 Rare and OOV Words Analysis

Another advantage of learning internal structure of words is that it can capture representations for out-of-vocabulary (OOV) words. To better understand the behavior of IntNet, Table 3 presents error analysis on in-vocabulary words (IV), out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV), and out-of-both-vocabulary words (OOBV) compared to different character models. The result shows that our proposed model significantly outperforms other character models on OOV words including OOTV, OOEV, and OOBV. For example, in OOBV category, our IntNet outperforms other models by more than 3% in terms of F1 score for Dutch and German datasets.

Furthermore, we present comparisons of nearest neighbors with different models for frequent words, rare words, and OOV words. Table 4 shows the results of nearest neighbors for learning word shape information, which gives insights on what kind of character-to-word representations can be learned by different models. For example, in OOV words, our IntNet model learns a better $xx\text{-}month$ shape pattern when matching $11\text{-}month$ compared to other models.

5.4 Discussion

In many situations, learning character-to-word representations of subword sequences that exceed the typical length of word shape pattern or morpheme sequences might result in noise. RNNs can capture longer sequences in theory, however,

longer sequences do not guarantee better results when learning prefixes and suffixes. The funnel-shaped wide architecture of IntNet, uses different kernels with different levels of concatenation to capture patterns of different subword lengths and that is flexible than char-LSTM and char-CNN. For example, Table 4 shows $Thursday$ in OOV words, our model learns a better word-shape structure for character-to-word representations compared to other methods.

When considering training time, IntNet is only 20% slower than char-CNN for the whole training process. Also, learning word representations use fewer parameters than learning sentence representations. Therefore, the impact of training speed for sequence labeling is limited. The inference time of IntNet is almost the same as char-CNN.

6 Conclusion

We presented empirical comparisons of different character embedding models for learning character-to-word representations and investigated the gaps between methods for learning representations of words and sentences. We conducted detailed experiments of different state-of-the-art convolutional models, and explored the advantages and disadvantages of their components for learning word shape information. Furthermore, we presented IntNet, a funnel-shaped wide convolutional neural architecture with no down-sampling that focuses on learning better internal structure of words by composing their characters from limited supervised training corpora. Our in-depth analysis showed that a shallow wide architecture is better than a narrow deep architecture for learning character-to-word representations. Omitting down-sampling operations is useful for capturing the sequential patterns of prefixes and suffixes. Our proposed compositional character-to-word model does not leverage any external resources, hand-crafted features, additional knowledge, joint training, or character-level preprocessing, and achieves new state-of-the-art performance for various sequence labeling tasks, including named entity recognition, part-of-speech tagging and syntactic chunking. In the future, we would like to explore using the IntNet model for other NLP tasks.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

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.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1107–1116, Valencia, Spain.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759. 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.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In *Proceedings of European Conference on Computer Vision*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT 2016*, pages 260–270.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank F. Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Xuezhe Ma and Eduard Hovy. 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*, pages 1064–1074, Berlin, Germany.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, volume 14, pages 1532–1543. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1765.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 412–418.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2121–2130.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1818–1826.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex A. Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

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 CoNLL-2003*, pages 142–147. Edmonton, Canada.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of the International Conference on Learning Representations*.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pages 87.1–87.12.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657.