Attending to Characters in Neural Sequence Labeling Models

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Gamal K.O. Crichton **Sampo Pyysalo**
Language Technology Lab
Dept. of Theoretical & Applied Linguistics
University of Cambridge
United Kingdom
{gkoc2,smp66}@cam.ac.uk

Abstract

Sequence labeling architectures use word embeddings for capturing similarity, but suffer when handling previously unseen or rare words. We investigate character-level extensions to such models and propose a novel architecture for combining alternative word representations. By using an attention mechanism, the model is able to dynamically decide how much information to use from a word- or character-level component. We evaluated different architectures on a range of sequence labeling datasets, and character-level extensions were found to improve performance on every benchmark. In addition, the proposed attention-based architecture delivered the best results even with a smaller number of trainable parameters.

1 Introduction

Many NLP tasks, including named entity recognition (NER), part-of-speech (POS) tagging and shallow parsing can be framed as types of sequence labeling. The development of accurate and efficient sequence labeling models is thereby useful for a wide range of downstream applications. Work in this area has traditionally involved task-specific feature engineering – for example, integrating gazetteers for named entity recognition, or using features from a morphological analyser in POS-tagging. Recent developments in neural architectures and representation learning have opened the door to models that can discover useful features automatically from the data. Such sequence labeling systems are applicable to many tasks, using only the surface text as input, yet are able to achieve competitive results (Collobert et al., 2011; Irsoy and Cardie, 2014).

Current neural models generally make use of word embeddings, which allow them to learn similar representations for semantically or functionally similar words. While this is an important improvement over count-based models, they still have weaknesses that should be addressed. The most obvious problem arises when dealing with out-of-vocabulary (OOV) words – if a token has never been seen before, then it does not have an embedding and the model needs to back-off to a generic OOV representation. Words that have been seen very infrequently have embeddings, but they will likely have low quality due to lack of training data. The approach can also be sub-optimal in terms of parameter usage – for example, certain suffixes indicate more likely POS tags for these words, but this information gets encoded into each individual embedding as opposed to being shared between the whole vocabulary.

In this paper, we construct a task-independent neural network architecture for sequence labeling, and then extend it with two different approaches for integrating character-level information. By operating on individual characters, the model is able to infer representations for previously unseen words and share information about morpheme-level regularities. We propose a novel architecture for combining character-level representations with word embeddings using a gating mechanism, also referred to as *attention*, which allows the model to dynamically decide which source of information to use for each word. In addition, we describe a new objective for model training where the character-level representations are optimised to mimic the current state of word embeddings.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 309–318, Osaka, Japan, December 11-17 2016.

We evaluate the neural models on 8 datasets from the fields of NER, POS-tagging, chunking and error detection in learner texts. Our experiments show that including a character-based component in the sequence labeling model provides substantial performance improvements on all the benchmarks. In addition, the attention-based architecture achieves the best results on all evaluations, while requiring a smaller number of parameters.

2 Bidirectional LSTM for sequence labeling

We first describe a basic word-level neural network for sequence labeling, following the models described by Lample et al. (2016) and Rei and Yannakoudakis (2016), and then propose two alternative methods for incorporating character-level information.

Figure 1 shows the general architecture of the sequence labeling network. The model receives a sequence of tokens $(w_1, ..., w_T)$ as input, and predicts a label corresponding to each of the input tokens. The tokens are first mapped to a distributed vector space, resulting in a sequence of word embeddings $(x_1, ..., x_T)$. Next, the embeddings are given as input to two LSTM (Hochreiter and Schmidhuber, 1997) components moving in opposite directions through the text, creating context-specific representations. The respective forward- and backward-conditioned representations are concatenated for each word position, resulting in representations that are conditioned on the whole sequence:

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}}) \qquad \overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t+1}}) \qquad h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{1}$$

We include an extra narrow hidden layer on top of the LSTM, which proved to be a useful modification based on development experiments. An additional hidden layer allows the model to detect higher-level feature combinations, while constraining it to be small forces it to focus on more generalisable patterns:

$$d_t = tanh(W_d h_t) \tag{2}$$

where W_d is a weight matrix between the layers, and the size of d_t is intentionally kept small.

Finally, to produce label predictions, we use either a softmax layer or a conditional random field (CRF, Lafferty et al. (2001)). The softmax calculates a normalised probability distribution over all the possible labels for each word:

$$P(y_t = k|d_t) = \frac{e^{W_{o,k} d_t}}{\sum_{\tilde{k} \in K} e^{W_{o,\tilde{k}} d_t}} \tag{3}$$

where $P(y_t = k|d_t)$ is the probability of the label of the t-th word (y_t) being k, K is the set of all possible labels, and $W_{o,k}$ is the k-th row of output weight matrix W_o. To optimise this model, we minimise categorical crossentropy, which is equivalent to minimising the negative log-probability of the correct labels:

$$E = -\sum_{t=1}^{T} log(P(y_t|d_t)) \tag{4}$$

Following Huang et al. (2015), we can also use a CRF as the output layer, which conditions each prediction on the previously predicted label. In this architecture, the last hidden layer is used to predict confidence scores for the word having each of the possible labels. A separate weight matrix is used to learn transition probabilities between different labels, and the Viterbi algorithm is used to find an optimal sequence of weights. Given that y is a sequence of labels $[y_1, ..., y_T]$, then the CRF score for this sequence can be calculated as:

$$s(y) = \sum_{t=1}^{T} A_{t,y_t} + \sum_{t=0}^{T} B_{y_t,y_{t+1}} \tag{5}$$

$$A_{t,y_t} = W_{o,y_t} d_t \tag{6}$$

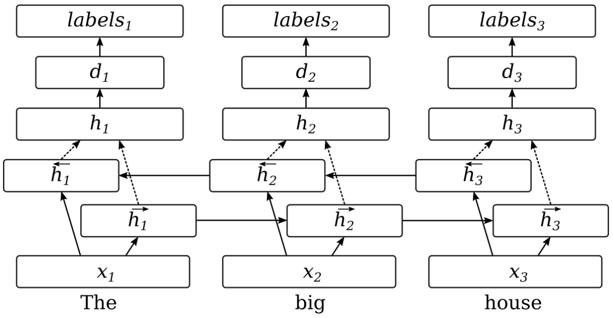

Figure 1: Neural sequence labeling model. Word embeddings are given as input; a bidirectional LSTM produces context-dependent representations; the information is passed through a hidden layer and the output layer. The outputs are either probability distributions for softmax, or confidence scores for CRF.

where A_{t,y_t} shows how confident the network is that the label on the t-th word is y_t. $B_{y_t,y_{t+1}}$ shows the likelihood of transitioning from label y_t to label y_{t+1}, and these values are optimised during training. The output from the model is the sequence of labels with the largest score $s(y)$, which can be found efficiently using the Viterbi algorithm. In order to optimise the CRF model, the loss function maximises the score for the correct label sequence, while minimising the scores for all other sequences:

$$E = -s(y) + log \sum_{\tilde{y} \in \widetilde{Y}} e^{s(\tilde{y})} \tag{7}$$

where $\widetilde{Y}$ is the set of all possible label sequences.

3 Character-level sequence labeling

Distributed embeddings map words into a space where semantically similar words have similar vector representations, allowing the models to generalise better. However, they still treat words as atomic units and ignore any surface- or morphological similarities between different words. By constructing models that operate over individual characters in each word, we can take advantage of these regularities. This can be particularly useful for handling unseen words – for example, if we have never seen the word *cabinets* before, a character-level model could still infer a representation for this word if it has previously seen the word *cabinet* and other words with the suffix *-s*. In contrast, a word-level model can only represent this word with a generic out-of-vocabulary representation, which is shared between all other unseen words.

Research into character-level models is still in fairly early stages, and models that operate exclusively on characters are not yet competitive to word-level models on most tasks. However, instead of fully replacing word embeddings, we are interested in combining the two approaches, thereby allowing the model to take advantage of information at both granularity levels. The general outline of our approach is shown in Figure 2. Each word is broken down into individual characters, these are then mapped to a sequence of character embeddings $(c_1, ..., c_R)$, which are passed through a bidirectional LSTM:

$$\overrightarrow{h_i^*} = LSTM(c_i, \overrightarrow{h_{i-1}^*}) \qquad \overleftarrow{h_i^*} = LSTM(c_i, \overleftarrow{h_{i+1}^*}) \tag{8}$$

We then use the last hidden vectors from each of the LSTM components, concatenate them together, and pass the result through a separate non-linear layer.

$$h^* = [\overrightarrow{h_R^*}; \overleftarrow{h_1^*}] \qquad m = tanh(W_m h^*) \tag{9}$$

where W_m is a weight matrix mapping the concatenated hidden vectors from both LSTMs into a joint word representation m, built from individual characters.

We now have two alternative feature representations for each word – x_t from Section 2 is an embedding learned on the word level, and $m^{(t)}$ is a representation dynamically built from individual characters in

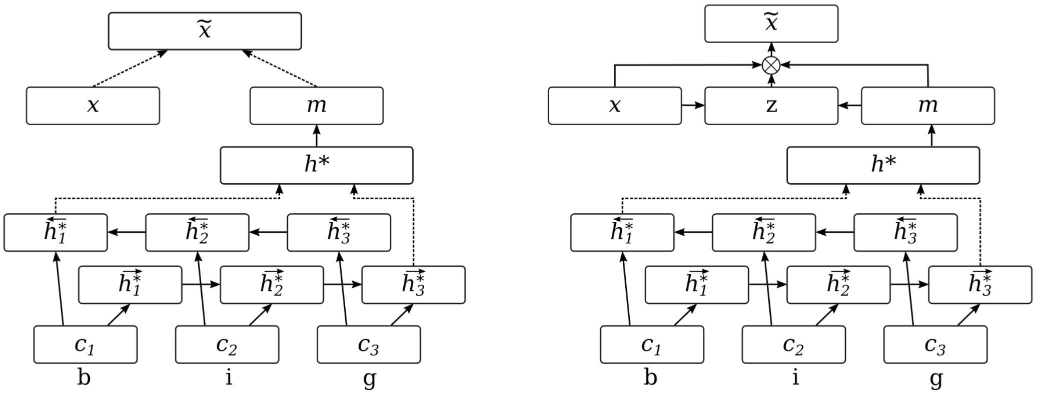

Figure 2: Left: concatenation-based character architecture. Right: attention-based character architecture. The dotted lines indicate vector concatenation.

the t-th word of the input text. Following Lample et al. (2016), one possible approach is to concatenate the two vectors and use this as the new word-level representation for the sequence labeling model:

$$\widetilde{x} = [x; m] \tag{10}$$

This approach, also illustrated in Figure 2, assumes that the word-level and character-level components learn somewhat disjoint information, and it is beneficial to give them separately as input to the sequence labeler.

4 Attention over character features

Alternatively, we can have the word embedding and the character-level component learn the same semantic features for each word. Instead of concatenating them as alternative feature sets, we specifically construct the network so that they would learn the same representations, and then allow the model to decide how to combine the information for each specific word.

We first construct the word representation from characters using the same architecture – a bidirectional LSTM operates over characters, and the last hidden states are used to create vector m for the input word. Instead of concatenating this with the word embedding, the two vectors are added together using a weighted sum, where the weights are predicted by a two-layer network:

$$z = \sigma(W_z^{(3)} tanh(W_z^{(1)} x + W_z^{(2)} m)) \qquad \widetilde{x} = z \cdot x + (1 - z) \cdot m \tag{11}$$

where $W_z^{(1)}$, $W_z^{(2)}$ and $W_z^{(3)}$ are weight matrices for calculating z, and $\sigma()$ is the logistic function with values in the range $[0, 1]$. The vector z has the same dimensions as x or m, acting as the weight between the two vectors. It allows the model to dynamically decide how much information to use from the character-level component or from the word embedding. This decision is done for each feature separately, which adds extra flexiblity – for example, words with regular suffixes can share some character-level features, whereas irregular words can store exceptions into word embeddings. Furthermore, previously unknown words are able to use character-level regularities whenever possible, and are still able to revert to using the generic OOV token when necessary.

The main benefits of character-level modeling are expected to come from improved handling of rare and unseen words, whereas frequent words are likely able to learn high-quality word-level embeddings directly. We would like to take advantage of this, and train the character component to predict these word embeddings. Our attention-based architecture requires the learned features in both word representations to align, and we can add in an extra constraint to encourage this. During training, we add a term to the loss function that optimises the vector m to be similar to the word embedding x:

Name	Task	# labels	# train tokens	# dev tokens	# test tokens
CoNLL00	Chunking	22	158,795	52,932	47,377
CoNLL03	NER	8	203,621	51,362	46,435
PTB-POS	POS	48	912,344	131,768	129,654
FCEPUBLIC	Error det	2	452,833	34,599	41,477
BC2GM	NER	3	355,405	71,042	143,465
CHEMDNER	NER	3	891,948	886,324	766,033
JNLPBA	NER	11	445,090	47,461	101,039
GENIA-POS	POS	42	397,690	52,697	50,556

Table 1: Details for each of the evaluation datasets.

$$\widetilde{E} = E + \sum_{t=1}^{T} g_t(1 - cos(m^{(t)}, x_t)) \qquad g_t = \begin{cases} 0, & \text{if } w_t = OOV \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

Equation 12 maximises the cosine similarity between $m^{(t)}$ and x_t. Importantly, this is done only for words that are not out-of-vocabulary – we want the character-level component to learn from the word embeddings, but this should exclude the OOV embedding, as it is shared between many words. We use g_t to set this cost component to 0 for any OOV tokens.

While the character component learns general regularities that are shared between all the words, individual word embeddings provide a way for the model to store word-specific information and any exceptions. Therefore, while we want the character-based model to shift towards predicting high-quality word embeddings, it is not desireable to optimise the word embeddings towards the character-level representations. This can be achieved by making sure that the optimisation is performed only in one direction; in Theano (Bergstra et al., 2010), the *disconnected_grad* function gives the desired effect.

5 Datasets

We evaluate the sequence labeling models and character architectures on 8 different datasets. Table 1 contains information about the number of labels and dataset sizes for each of them.

- **CoNLL00**: The CoNLL-2000 dataset (Tjong Kim Sang and Buchholz, 2000) is a frequently used benchmark for the task of chunking. Wall Street Journal Sections 15-18 from the Penn Treebank are used for training, and Section 20 as the test data. As there is no official development set, we separated some of the training set for this purpose.

- **CoNLL03**: The CoNLL-2003 corpus (Tjong Kim Sang and De Meulder, 2003) was created for the shared task on language-independent NER. We use the English section of the dataset, containing news stories from the Reuters Corpus[1].

- **PTB-POS**: The Penn Treebank POS-tag corpus (Marcus et al., 1993) contains texts from the Wall Street Journal, annotated for part-of-speech tags. The PTB label set includes 36 main tags and an additional 12 tags covering items such as punctuation.

- **FCEPUBLIC**: The publicly released subset of the First Certificate in English (FCE) dataset contains short essays written by language learners and manual corrections by examiners (Yannakoudakis et al., 2011). We use a version of this corpus converted into a binary error detection task, where each token is labeled as being correct or incorrect in the given context.

- **BC2GM**: The BioCreative II Gene Mention corpus (Smith et al., 2008) consists of 20,000 sentences from biomedical publication abstracts and is annotated for mentions of the names of genes, proteins and related entities using a single NE class.

[1] http://about.reuters.com/researchandstandards/corpus/

	CoNLL00		CoNLL03		PTB-POS		FCEPUBLIC	
	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
Word-based	91.48	91.23	86.89	79.86	96.29	96.42	46.58	41.24
Char concat	92.57	92.35	89.81	83.37	97.20	97.22	46.44	41.27
Char attention	**92.92**	**92.67**	**89.91**	**84.09**	**97.22**	**97.27**	**47.17**	**41.88**

	BC2GM		CHEMDNER		JNLPBA		GENIA-POS	
	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
Word-based	84.07	84.21	78.63	79.74	75.46	70.75	97.55	97.39
Char concat	87.54	87.75	82.80	83.56	76.82	72.24	98.59	98.49
Char attention	**87.98**	**87.99**	**83.75**	**84.53**	**77.38**	**72.70**	**98.67**	**98.60**

Table 2: Comparison of word-based and character-based sequence labeling architectures on 8 datasets. The evaluation measure used for each dataset is specified in Section 6.

- **CHEMDNER**: The BioCreative IV Chemical and Drug (Krallinger et al., 2015) NER corpus consists of 10,000 abstracts annotated for mentions of chemical and drug names using a single class. We make use of the official splits provided by the shared task organizers.

- **JNLPBA**: The JNLPBA corpus (Kim et al., 2004) consists of 2,404 biomedical abstracts and is annotated for mentions of five entity types: CELL LINE, CELL TYPE, DNA, RNA, and PROTEIN. The corpus was derived from GENIA corpus entity annotations for use in the shared task organized in conjuction with the BioNLP 2004 workshop.

- **GENIA-POS**: The GENIA corpus (Ohta et al., 2002) is one of the most widely used resources for biomedical NLP and has a rich set of annotations including parts of speech, phrase structure syntax, entity mentions, and events. Here, we make use of the GENIA POS annotations, which cover 2,000 PubMed abstracts (approx. 20,000 sentences). We use the same 210-document test set as Tsuruoka et al. (2005), and additionally split off a sample of 210 from the remaining documents as a development set.

6 Experiment settings

For data prepocessing, all digits were replaced with the character '0'. Any words that occurred only once in the training data were replaced by the generic OOV token for word embeddings, but were still used in the character-level components. The word embeddings were initialised with publicly available pretrained vectors, created using word2vec (Mikolov et al., 2013), and then fine-tuned during model training. For the general-domain datasets we used 300-dimensional vectors trained on Google News[2]; for the biomedical datasets we used 200-dimensional vectors trained on PubMed and PMC[3]. The embeddings for characters were set to length 50 and initialised randomly.

The LSTM layer size was set to 200 in each direction for both word- and character-level components. The hidden layer d has size 50, and the combined representation m has the same length as the word embeddings. CRF was used as the output layer for all the experiments – we found that this gave most benefits to tasks with larger numbers of possible labels. Parameters were optimised using AdaDelta (Zeiler, 2012) with default learning rate 1.0 and sentences were grouped into batches of size 64. Performance on the development set was measured at every epoch and training was stopped if performance had not improved for 7 epochs; the best-performing model on the development set was then used for evalua-

[2] https://code.google.com/archive/p/word2vec/

[3] http://bio.nlplab.org/

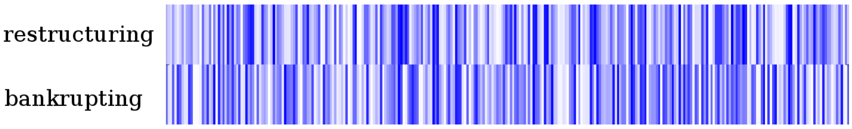

Figure 3: Visualisation of attention values for two words, trained on the PTB-POS dataset. Darker blue indicates features with higher weights for the character-level representation. *Restructuring* was present in the vocabulary, while *bankrupting* is an OOV.

tion on the test set. In order to avoid any outlier results due to randomness in the model initialisation, we trained each configuration with 10 different random seeds and present here the averaged results.

When evaluating on each dataset, we report the measures established in previous work. Token-level accuracy is used for PTB-POS and GENIA-POS; $F_{0.5}$ score over the erroneous words for FCEPUBLIC; the official evaluation script for BC2GM which allows for alternative correct entity spans; and microaveraged mention-level F_1 score for the remaining datasets.

7 Results

While optimising the hyperparameters for each dataset separately would likely improve individual performance, we conduct more controlled experiments on a task-independent model. Therefore, we use the same hyperparameters from Section 6 on all datasets, and the development set is only used for the stopping condition. With these experiments, we wish to determine 1) on which sequence labeling tasks do character-based models offer an advantange, and 2) which character-based architecture performs better.

Results for the different model architectures on all 8 datasets are shown in Table 2. As can be seen, including a character-based component in the sequence labeling architecture improves performance on every benchmark. The NER datasets have the largest absolute improvement – the model is able to learn character-level patterns for names, and also improve the handling of any previously unseen tokens.

Compared to concatenating the word- and character-level representations, the attention-based character model outperforms the former on all evaluations. The mechanism for dynamically deciding how much character-level information to use allows the model to better handle individual word representations, giving it an advantage in the experiments. Visualisation of the attention values in Figure 3 shows that the model is actively using character-based features, and the attention areas vary between different words.

The results of this general tagging architecture are competitive, even when compared to previous work using hand-crafted features. The network achieves 97.27% on PTB-POS compared to 97.55% by Huang et al. (2015), and 72.70% on JNLPBA compared to 72.55% by Zhou and Su (2004). In some cases, we are also able to beat the previous best results – 87.99% on BC2GM compared to 87.48% by Campos et al. (2015), and 41.88% on FCEPUBLIC compared to 41.1% by Rei and Yannakoudakis (2016). Lample et al. (2016) report a considerably higher result of 90.94% on CoNLL03, indicating that the chosen hyperparameters for the baseline system are suboptimal for this specific task. Compared to the experiments presented here, their model used the IOBES tagging scheme instead of the original IOB, and embeddings pretrained with a more specialised method that accounts for word order.

It is important to also compare the parameter counts of alternative neural architectures, as this shows their learning capacity and indicates their time requirements in practice. Table 3 contains the parameter counts on three representative datasets. While keeping the model hyperparameters constant, the character-level models require additional parameters for the character composition and character embeddings. However, the attention-based model uses fewer parameters compared to the concatenation approach. When the two representations are concatenated, the overall word representation size is increased, which in turn increases the number of parameters required for the word-level bidirectional LSTM. Therefore, the attention-based character architecture achieves improved results even with a smaller parameter footprint.

	CoNLL03		FCEPUBLIC		CHEMDNER	
	# total	# noemb	# total	# noemb	# total	# noemb
Word-based	4,507,658	1,230,158	2,972,052	1,230,252	5,862,878	1,070,278
Char concat	4,987,658	1,710,158	3,452,052	1,710,252	6,182,878	1,390,278
Char attention	4,687,958	1,410,458	3,152,352	1,410,552	5,943,078	1,150,478

Table 3: Comparison of trainable parameters in each of the neural model architectures. *# total* shows the total number of parameters; *# noemb* shows the parameter count excluding word embeddings, as only a small fraction of the embeddings are utilised at every iteration.

8 Related work

There is a wide range of previous work on constructing and optimising neural architectures applicable to sequence labeling. Collobert et al. (2011) described one of the first task-independent neural tagging models using convolutional neural networks. They were able to achieve good results on POS tagging, chunking, NER and semantic role labeling, without relying on hand-engineered features. Irsoy and Cardie (2014) experimented with multi-layer bidirectional Elman-style recurrent networks, and found that the deep models outperformed conditional random fields on the task of opinion mining. Huang et al. (2015) described a bidirectional LSTM model with a CRF layer, which included hand-crafted features specialised for the task of named entity recognition. Rei and Yannakoudakis (2016) evaluated a range of neural architectures, including convolutional and recurrent networks, on the task of error detection in learner writing. The word-level sequence labeling model described in this paper follows the previous work, combining useful design choices from each of them. In addition, we extended the model with two alternative character-level architectures, and evaluated its performance on 8 different datasets.

Character-level models have the potential of capturing morpheme patterns, thereby improving generalisation on both frequent and unseen words. In recent years, there has been an increase in research into these models, resulting in several interesting applications. Ling et al. (2015b) described a character-level neural model for machine translation, performing both encoding and decoding on individual characters. Kim et al. (2016) implemented a language model where encoding is performed by a convolutional network and LSTM over characters, whereas predictions are given on the word-level. Cao and Rei (2016) proposed a method for learning both word embeddings and morphological segmentation with a bidirectional recurrent network over characters. There is also research on performing parsing (Ballesteros et al., 2015) and text classification (Zhang et al., 2015) with character-level neural models. Ling et al. (2015a) proposed a neural architecture that replaces word embeddings with dynamically-constructed character-based representations. We applied a similar method for operating over characters, but combined them with word embeddings instead of replacing them, as this allows the model to benefit from both approaches. Lample et al. (2016) described a model where the character-level representation is combined with word embeddings through concatenation. In this work, we proposed an alternative architecture, where the representations are combined using an attention mechanism, and evaluated both approaches on a range of tasks and datasets. Recently, Miyamoto and Cho (2016) have also described a related method for the task of language modelling, combining characters and word embeddings using gating.

9 Conclusion

Developments in neural network research allow for model architectures that work well on a wide range of sequence labeling datasets without requiring hand-crafted data. While word-level representation learning is a powerful tool for automatically discovering useful features, these models still come with certain weaknesses – rare words have low-quality representations, previously unseen words cannot be modeled at all, and morpheme-level information is not shared with the whole vocabulary.

In this paper, we investigated character-level model components for a sequence labeling architecture, which allow the system to learn useful patterns from sub-word units. In addition to a bidirectional LSTM operating over words, a separate bidirectional LSTM is used to construct word representations from individual characters. We proposed a novel architecture for combining the character-based representation with the word embedding by using an attention mechanism, allowing the model to dynamically choose which information to use from each information source. In addition, the character-level composition function is augmented with a novel training objective, optimising it to predict representations that are similar to the word embeddings in the model.

The evaluation was performed on 8 different sequence labeling datasets, covering a range of tasks and domains. We found that incorporating character-level information into the model improved performance on every benchmark, indicating that capturing features regarding characters and morphmes is indeed useful in a general-purpose tagging system. In addition, the attention-based model for combining character representations outperformed the concatenation method used in previous work in all evaluations. Even though the proposed method requires fewer parameters, the added ability of controlling how much character-level information is used for each word has led to improved performance on a range of different tasks.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

James Bergstra, Olivier Breuleux, Frederic Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy).*

David Campos, Sergio Matos, and Jose L. Oliveira. 2015. A document processing pipeline for annotating chemical entities in scientific documents. *Journal of Cheminformatics*, 7.

Kris Cao and Marek Rei. 2016. A Joint Model for Word Embedding and Word Morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP (RepL4NLP-2016).*

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv:1508.01991.*

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the Bio-entity Recognition Task at JNLPBA. *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications.*

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. *In Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI16).*

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. CHEMDNER: The drugs and chemical names extraction challenge. *Journal of Cheminformatics*, 7(Suppl 1).

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.*

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016.*

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015a. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.*

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based Neural Machine Translation. *arXiv preprint arXiv:1511.04586*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

Tomáš Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated Word-Character Recurrent Language Model. *arXiv preprint arXiv:1606.01700*.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. *Proceedings of the second international conference on Human Language Technology Research*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, Manabu Torii, Hongfang Liu, Barry Haddow, Craig A Struble, Richard J Povinelli, Andreas Vlachos, William A Baumgartner, Lawrence Hunter, Bob Carpenter, Richard Tzong-Han Tsai, Hong-Jie Dai, Feng Liu, Yifei Chen, Chengjie Sun, Sophia Katrenko, Pieter Adriaans, Christian Blaschke, Rafael Torres, Mariana Neves, Preslav Nakov, Anna Divoli, Manuel Maña-López, Jacinto Mata, and W John Wilbur. 2008. Overview of BioCreative II gene mention recognition. *Genome biology*, 9 Suppl 2.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, 7.

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*.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Proceedings of Panhellenic Conference on Informatics*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*.

GuoDong Zhou and Jian Su. 2004. Exploring Deep Knowledge Resources in Biomedical Name Recognition. *Workshop on Natural Language Processing in Biomedicine and Its Applications at COLING*.