Transfer Learning and Sentence Level Features for Named Entity Recognition on Tweets

Pius von Däniken
SpinningBytes AG

Mark Cieliebak
ZHAW

Abstract

We present our system for the WNUT 2017 Named Entity Recognition challenge on Twitter data. We describe two modifications of a basic neural network architecture for sequence tagging. First, we show how we exploit additional labeled data, where the Named Entity tags differ from the target task. Then, we propose a way to incorporate sentence level features. Our system uses both methods and ranked second for entity level annotations, achieving an F1-score of 40.78, and second for surface form annotations, achieving an F1-score of 39.33.

1 Introduction

Named Entity Recognition (NER) is an important Natural Language Processing task. Its goal is to tag entities such as names of people and locations in text. State-of-the-art systems can achieve F1-scores of up to 92 points on English news texts (Chiu and Nichols, 2015). Achieving good performance on more complex domains such as user generated texts on social media is still a hard problem. The best system submitted for the WNUT 2016 shared task achieved an F1-score of 52.41 on English Twitter data (Strauss et al., 2016).

In this work, we present our submission for the WNUT 2017 shared task on "Novel and Emerging Entity Recognition" (Derczynski et al., 2017). We extend a basic neural network architecture for sequence tagging (Chiu and Nichols, 2015; Collobert et al., 2011) by incorporating sentence level feature vectors and exploiting additional labeled data for transfer learning. We build on and take inspiration from recent work from (Falkner et al., 2017; Sileo et al., 2017) on NER for French Twitter data (Lopez et al., 2017).

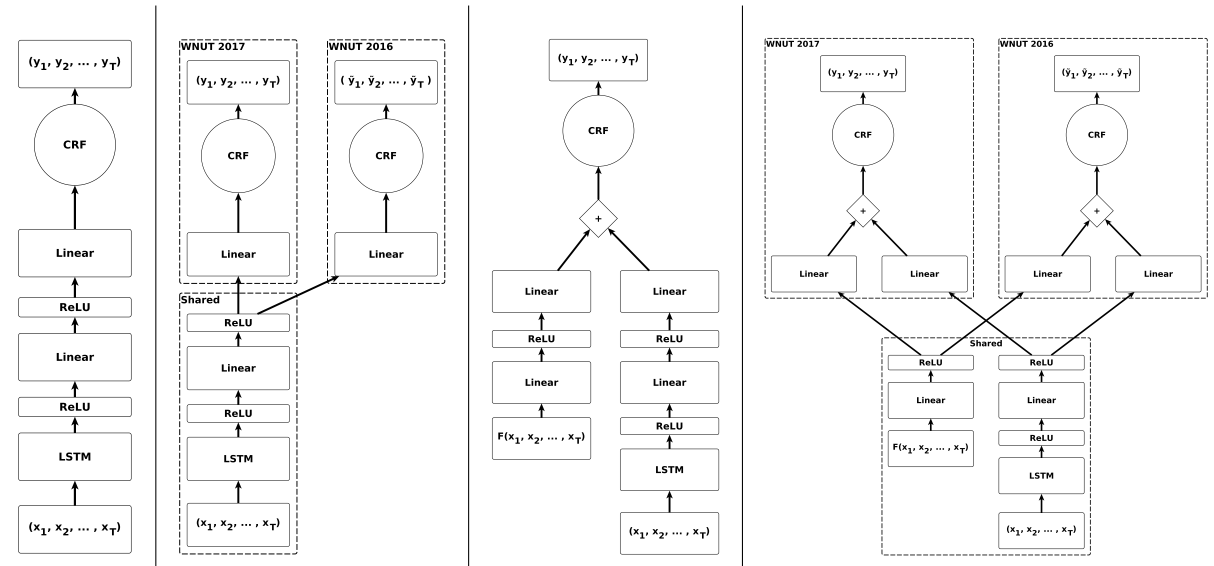
Our submitted solution reached an F1-score of 41.76 for entity level annotations and 57.98 on surface form annotations. This places us second on entity level annotations, where the best system achieved an F1-score of 41.90, and fourth on surface form annotations, where the best system achieved an F1-score of 66.59.

2 System Description

Our solution is based on a sequence labeling system that uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) which extracts features for training a Conditional Random Field (Sutton and McCallum, 2012). We apply a transfer learning approach, since previous research has shown that this can improve sequence labeling systems (Yang et al., 2017). More precisely, we modify the base system to allow for joint training on the WNUT 2016 corpus (Strauss et al., 2016), which uses a different tag set than our target task. In addition, we extend the system to incorporate sentence level feature vectors. All these methods are combined to build the system that we used for our submission to the WNUT 2017 shared task. Figure 1 shows an overview of the different architectures, which are described in detail in the following sections.

2.1 Basic Sequence Labeling System

Figure 1a shows an overview of our base system. We use a bidirectional Long Short Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) to learn the potential function for a linear chain Conditional Random Field (CRF) (Sutton and McCallum, 2012) to predict a sequence of Named Entity tags $y_{1:T}$ from a sequence of feature vectors $x_{1:T}$. This is based on an architecture previously used in (Chiu and Nichols, 2015), which



(a) Basic System (b) Transfer Learning Architecture (c) Incorporating Sentence Level Features (d) Architecture Using Transfer Learning and Sentence Level Features

Figure 1: Overview Of The Different Network Architectures Used

achieved state-of-the-art performance for Named Entity Recognition on the English CoNLL 2003 data set (Tjong Kim Sang and De Meulder, 2003).

Bidirectional LSTM: For every word in w_t in a given input sentence $w_{1:T}$, we first compute a feature vector x_t, which is the concatenation of all the word level features described in Section 2.5. The sequence of feature vectors $x_{1:T}$ is then fed to a bidirectional LSTM. The output of both the forward and backward LSTM are concatenated to get $o_{1:T}$, which get passed through a Rectified Linear Unit, ($ReLU$) (Nair and Hinton, 2010). Every $o_t \in o_{1:T}$ then gets passed through a fully connected feed-forward network with one hidden layer and $ReLU$ activation: $s_t = W_2\, relu(W_1 o_t + b_1) + b_2$. Let N_{tags} be the number of possible NER-tags, d_o the dimension of o_t and d_h the dimension of the hidden layer. The resulting vector $s_t \in \mathbb{R}^{N_{tags}}$ represents a score for every possible tag y at time step t. The values $W_1 \in \mathbb{R}^{d_h \times d_o}$, $b_1 \in \mathbb{R}^{d_h}$, $W_2 \in \mathbb{R}^{N_{tags} \times d_h}$ and $b_2 \in \mathbb{R}^{N_{tags}}$ are weights of the feed-forward network.

Conditional Random Field: A linear chain CRF models the conditional probability of an output sequence $y_{1:T}$ given an input sequence $x_{1:T}$ as:

$$p\left(y_{1:T}|x_{1:T}\right) = \frac{1}{Z(x_{1:T})} \prod_{t=1}^{T} e^{\phi(y_{t-1}, y_t, x_{1:T}, t, \Theta)} \tag{1}$$

where $Z\left(x_{1:T}\right)$ is a normalization constant:

$$Z\left(x_{1:T}\right) = \sum_{\forall y_{1:T}} \prod_{t=1}^{T} e^{\phi(y_{t-1}, y_t, x_{1:T}, t, \Theta)} \tag{2}$$

ϕ is a potential function parametrized by a set of parameters Θ. In our case we use:

$$\begin{aligned} \phi\left(y_{t-1}, y_t, x_{1:T}, t, \Theta = \{\theta, A\}\right) = \\ s_{\theta, y_t, t}\left(x_{1:T}\right) + A_{y_{t-1}, y_t} \end{aligned} \tag{3}$$

Let θ be the parameters of the network described above. Then $s_{\theta, y_t, t}\left(x_{1:T}\right)$ is the score that the network parametrized by θ outputs for tag y_t at time step t given the input sequence $x_{1:T}$. $A \in \mathbb{R}^{N_{tags} \times N_{tags}}$ is a matrix such that $A_{i,j}$ is the score of transitioning from tag i to tag j.

Training: During training we try to maximize the likelihood of the true tag sequence $y_{1:T}$ given the input feature vectors $x_{1:T}$. We use the Adam (Kingma and Ba, 2014) algorithm to optimize the parameters $\Theta = \{\theta, A\}$. Additionally we perform gradient clipping (Pascanu et al., 2012) and apply dropout (Srivastava et al., 2014) to the LSTM outputs $o_{1:T}$. The neural network parameters θ are randomly initialized from a normal distribution with mean zero and variance according to (Glorot and Bengio, 2010) (normal Glorot initialization). The transition scores A are initialized from a uniform distribution with mean zero and variance according to (Glorot and Bengio, 2010), (uniform Glorot initialization).

2.2 Transfer Learning

In this setting we use the WNUT 2016 corpus (Strauss et al., 2016) as an additional source of labeled data. The idea is to train the upper layers of the neural network on both datasets to improve its generalization ability. It was shown in (Yang et al., 2017) that this can improve the system performance. Figure 1b gives an overview of our transfer learning architecture.

Modified Architecture: We share all network layers except for the last linear projection to get separate tag scores for each data set:

$$\begin{aligned} s_t^{2016} &= W_2^{2016} relu(W_1 o_t + b_1) + b_2^{2016} \\ s_t^{2017} &= W_2^{2017} relu(W_1 o_t + b_1) + b_2^{2017} \end{aligned} \tag{4}$$

The resulting tag scores get fed to separate CRFs, which have separate transition matrices A^{2016} and A^{2017}.

Training: During training we alternately use a batch from each dataset and backpropagate the loss of the corresponding CRF.

2.3 Incorporating Sentence Level Features

Figure 1c shows how we include sentence level features into our architecture. In this setting we take an additional feature vector $f_{sent} = F(x_{1:T}) \in \mathbb{R}^{d_{sent}}$ for each input sentence $x_{1:T}$.

Modified Architecture: We use an additional feed-forward network to extract tag scores $s_{sent} \in \mathbb{R}^{N_{tags}}$ from the sentence feature vector f_{sent}:

$$s_{sent} = W_{2,sent}\, relu\left(W_{1,sent} f_{sent} + b_{1,sent}\right) + b_{2,sent}$$

The dimensions used are: $W_{1,sent} \in \mathbb{R}^{d_{h,sent} \times d_{sent}}$, $b_{1,sent} \in \mathbb{R}^{d_{h,sent}}$, $W_{2,sent} \in \mathbb{R}^{N_{tags} \times d_{h,sent}}$ and $b_{2,sent} \in \mathbb{R}^{N_{tags}}$. The value $d_{h,sent}$ is the dimension of the hidden layer of the feed-forward network. Let $s_{1:T,word}$ be the scores that the basic network described in Section 2.1 outputs for sequence $x_{1:T}$. To get the final scores $s_{1:T}$ fed to the CRF we add s_{sent} to every $s_{t,word} \in s_{1:T,word}$: $s_t = s_{sent} + s_{t,word}$.

2.4 Combined System

The combined system adds the sentence level features to the transfer learning architecture. We share all layers except the linear projections to tag scores for both sentence features and word features in a manner analogous to Sections 2.2 and 2.3. The resulting architecture is shown in Figure 1d.

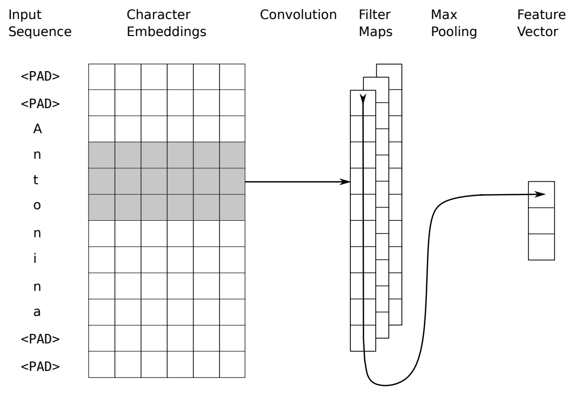


Figure 2: Neural Network used to extract character level features

2.5 Features

Word Embeddings: We use the FastText (Bojanowski et al., 2016) library to compute word embeddings. We train the model on a corpus of 200 million tweets and all tweets from the WNUT 2016 and WNUT 2017 corpora. The vocabulary contains all words occurring at least 10 times. Other parameters use the default values set by the library [1]. In particular, the size of the context window is set to 5 and the embedding dimension is 100.

This results in an embedding matrix $E_{word} \in \mathbb{R}^{N_{vocab} \times 100}$, where N_{vocab} is the number of unique tokens in the WNUT 2016 and WNUT 2017 corpora. FastText predicts embedding vectors for words that were out-of-vocabulary during training by considering character n-grams of the word. The embedding matrix E_{word} is not updated during training.

Word Capitalization Features: Following (Chiu and Nichols, 2015) we add explicit capitalization features, since capitalization information is lost during word embedding lookups. The 6 feature options are: *all capitalized*, *uppercase initial*, *all lower cased*, *mixed capitalization*, *emoji* and *other*. An embedding matrix $E_{wordCap} \in \mathbb{R}^{6 \times d_{wordCap}}$ is used to feed these features to the network and updated during training via back-propagation. $E_{wordCap}$ is initialized using normal Glorot initialization.

Character Convolution Features: A convolutional neural network is used to extract additional character level features. Its architecture is shown in Figure 2. First, we add special padding tokens

[1] https://github.com/facebookresearch/fastText

on both sides of the character sequence w, to extend it to a target length, $l_{w,max}$. If there is an odd number of paddings, the additional padding is added on the right. For sequences longer than $l_{w,max}$, only the first $l_{w,max}$ characters are used. An embedding matrix $E_{char} \in \mathbb{R}^{N_c \times d_c}$ maps characters to $\mathbb{R}^{d_c}$ vectors. N_c is the number of unique characters in the dataset with the addition of the padding token.

Using E_{char}, we embed the padded sequence w and get $C_w \in \mathbb{R}^{l_{w,max} \times d_c}$. A set of m convolution filters $\in \mathbb{R}^{d_c \times h}$ is then applied to C_w. This results in m feature maps $M_i \in \mathbb{R}^{l_{w,max}-h+1}$, which are passed through a $ReLU$ activation. The final feature vector $F \in \mathbb{R}^m$ is attained by max pooling, such that $F_i = \max M_i$.

The embedding matrix is initialized using uniform Glorot initialization. The m convolution filters are initialized using normal Glorot initialization.

Character Capitalization Convolution Features: Analogous to the word capitalization features, we use additional character capitalization features. The feature options are: *upper*, *lower*, *punctuation*, *numeric* and *other*. We apply a neural network with the same architecture as described above to extract the final character capitalization feature vector.

Sentence Embeddings: In (Pagliardini et al., 2017) the authors introduce sent2vec, a new method for computing sentence embeddings. They show that these embeddings provide improved performance for several downstream tasks.

To train the sent2vec model, we use the same training set as the one used for word embeddings and we use default values for all the model parameters[2]. In particular, the resulting sentence feature vectors are in $\mathbb{R}^{100}$.

3 Experiments

We implemented the system described in Section 2.4 using the Tensorflow framework [3].

We monitored the systems performance during training and aborted experiments that had an F1-score of less than 40 after two epochs (evaluated on the development set). We let successful experiments run for the full 6 epochs (cf. Section 3.2). For the submission to WNUT 2017, we ran 6 successful experiments and submitted the one which had the highest entity level F1-score on the development set.

Parameter	Value
$l_{w,max}$	30
N_{tags} WNUT 2016	21
N_{tags} WNUT 2017	13
$d_{wordCap}$	6
d_c	15
LSTM hidden units	64
$d_{h,word}$	128
$d_{h,sent}$	128
m	10
h	3
Dropout rate	0.3
Learning Rate	0.003
Gradient Clip Norm	2
Batch size	100
Number of epochs	6

Table 1: Model Parameters

3.1 Preprocessing

Tokenization: Since the WNUT 2016 and WNUT 2017 corpora are in the CoNLL format, they are already tokenized. To tokenize the additional tweets used for training word and sentence embeddings (cf. Section 2.5), we use the Twitter tokenizer provided by the Python NLTK library [4].

Token Substitution: We perform some simple pattern-based token substitutions. To normalize Twitter user handles, we substitute every word starting with an @ character by a special user token. Similarly, all words starting with the prefix *http* are replaced by a url token. Finally, for words longer than one character, we remove up to one initial # character.

3.2 Model Parameters

Table 1 shows the parameters used for training the model.

3.3 Experiments Performed After The Submission

Following the submission, we conducted additional experiments to investigate the influence of the transfer learning approach and sent2vec features on the system performance.

[2]https://github.com/epfml/sent2vec

[3]https://www.tensorflow.org/

[4]http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.casual

	Precision (%)		Recall (%)		F1	
	Mean	Stddev	Mean	Stddev	Mean	Stddev
Surface Forms	45.55	0.47	34.94	0.87	39.54	0.55
Entities Overall	47.23	0.55	36.33	0.83	41.06	0.52
Corporation	8.81	0.99	10.86	1.62	9.70	1.14
Creative Work	22.41	2.55	11.03	1.50	14.73	1.73
Group	39.27	7.47	9.49	2.20	15.13	2.86
Location	58.55	2.88	47.11	1.79	52.12	0.66
Person	57.82	1.60	63.60	1.10	60.55	0.84
Product	22.47	2.17	7.87	1.93	11.60	2.38

Table 2: Aggregated performance of all experiments, run before the submission, evaluated on the test set

	Precision (%)	Recall (%)	F1
Surface Forms	45.47	34.66	39.33
Entities Overall	47.09	35.96	40.78
Corporation	8.24	11.67	9.66
Creative Work	21.92	11.76	15.31
Group	31.71	9.22	14.29
Location	58.95	44.80	50.91
Person	57.67	61.97	59.74
Product	20.00	5.13	8.16

Table 3: Performance of the submitted annotations evaluated on the test set

For each of the 4 systems described in Section 2, we ran 6 experiments. We use the same parameters as shown in Section 3.2.

4 Results

Table 2 shows precision, recall and F1-score of our system. We compute the mean and standard deviations over the 6 successful experiments we considered for submission (cf. Section 3). Table 3 shows the breakdown of the performance of the annotations we submitted for the WNUT 2017 shared task.

Table 4 shows the performance of the different subsystems proposed in Section 2. We report the mean and standard deviation over the 6 experiments we performed after submission, for every system.

All reported scores were computed using the evaluation script provided by the task organizers.

5 Discussion

From table 4 we can see that using sent2vec features increases precision and decreases recall slightly, leading to an overall lower performance compared to the basic system. The transfer learning system shows a more substantial decrease in precision and increase in recall and overall performs best out of the 4 systems. Combination of the two approaches is counterproductive and outperforms the basic system only slightly.

During training we observed that restarting experiments as described in Section 3 was only necessary when using sent2vec features.

One weakness of our transfer learning setting is that the two datasets we used have almost identical samples and only differ in their annotations. The WNUT 2016 corpus uses 10 entity classes: *company*, *facility*, *Geo location*, *movie*, *music artist*, *other*, *person*, *product*, *sports team*, and *TV show*. Further work is needed to study the effect of using an unrelated data set for transfer learning.

6 Conclusion

We described a deep learning approach for Named Entity Recognition on Twitter data, which extends a basic neural network for sequence tagging by using sentence level features and transfer learning. Our approach achieved 2nd place at the WNUT 2017 shared task for Named Entity Recognition, obtaining an F1-score of 40.78.

For future work, we plan to explore the power of transfer learning for NER in more depth. For instance, it would be interesting to see how annotated NER data for other languages or other text types affects the system performance.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *ArXiv e-prints* https://arxiv.org/abs/1607.04606.

Jason P. C. Chiu and Eric Nichols. 2015. Named Entity Recognition with Bidirectional LSTM-CNNs. *ArXiv e-prints* https://arxiv.org/abs/1511.08308.

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* 12:2493–2537.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition. In *Proceedings of the 3rd Workshop on Noisy, User-generated Text (W-NUT) at EMNLP*. ACL.

Nicole Falkner, Stefano Dolce, Pius von Däniken, and Mark Cieliebak. 2017. Swiss Chocolate at CAp 2017 NER challenge: Partially annotated data and

	Entities						Surface Forms					
	Precision (%)		Recall (%)		F1		Precision (%)		Recall (%)		F1	
	Mean	Stddev	Mean	Stddev	Mean	Stddev	Mean	Stddev	Mean	Stddev	Mean	Stddev
Basic System	58.77	3.72	32.47	1.23	41.74	0.70	56.53	3.80	30.75	1.37	39.73	0.71
Transfer Learning	48.17	1.34	**37.55**	1.43	**42.16**	0.52	46.31	1.31	**35.86**	1.51	**40.38**	0.62
Sent2Vec Features	**59.51**	1.73	30.91	0.52	40.67	0.41	**57.30**	1.98	29.20	0.58	38.66	0.46
Combined System	50.41	3.19	36.04	2.10	41.88	0.69	48.60	3.00	34.50	2.36	40.20	0.99

Table 4: Performance of the different subsystems evaluated on the test set, after the submission

transfer learning. Conférence sur l'Apprentissage Automatique.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR, Chia Laguna Resort, Italy, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv e-prints* https://arxiv.org/abs/1412.6980.

Cdric Lopez, Ioannis Partalas, Georgios Balikas, Nadia Derbas, Amlie Martin, Coralie Reutenauer, Frdrique Segond, and Massih-Reza Amini. 2017. French named entity recognition in twitter challenge. Technical report.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Omnipress, pages 807–814.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *ArXiv e-prints* https://arxiv.org/abs/1703.02507.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training Recurrent Neural Networks. *ArXiv e-prints* https://arxiv.org/abs/1211.5063.

Damien Sileo, Camille Pradel, Philippe Muller, and Tim Van de Cruys. 2017. Synapse at CAp 2017 NER challenge: Fasttext crf. Conférence sur l'Apprentissage Automatique.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(1):1929–1958.

Benjamin Strauss, Bethany E. Toma, Alan Ritter, Marie-Catherine de Marneffe, and Wei Xu. 2016. Results of the WNUT16 named entity recognition shared task. In *The 2nd Workshop on Noisy User-generated Text*. pages 138–144.

Charles Sutton and Andrew McCallum. 2012. An introduction to conditional random fields. *Found. Trends Mach. Learn.* 4(4):267–373.

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 142–147.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. *ArXiv e-prints* https://arxiv.org/abs/1703.06345.