Evaluating the Utility of Hand-crafted Features in Sequence Labelling

Minghao Wu, Fei Liu, Trevor Cohn

The University of Melbourne, Victoria, Australia
JD AI Research, Beijing, China

wuminghao@jd.com
fliu3@student.unimelb.edu.au
t.cohn@unimelb.edu.au

Abstract

Conventional wisdom is that hand-crafted features are redundant for deep learning models, as they already learn adequate representations of text automatically from corpora. In this work, we test this claim by proposing a new method for exploiting handcrafted features as part of a novel hybrid learning approach, incorporating a feature auto-encoder loss component. We evaluate on the task of named entity recognition (NER), where we show that including manual features for part-of-speech, word shapes and gazetteers can improve the performance of a neural CRF model. We obtain a $F_1$ of 91.89 for the CoNLL-2003 English shared task, which significantly outperforms a collection of highly competitive baseline models. We also present an ablation study showing the importance of auto-encoding, over using features as either inputs or outputs alone, and moreover, show including the autoencoder components reduces training requirements to 60%, while retaining the same predictive accuracy.

1 Introduction

Deep neural networks have been proven to be a powerful framework for natural language processing, and have demonstrated strong performance on a number of challenging tasks, ranging from machine translation (Cho et al., 2014b,a), to text categorisation (Zhang et al., 2015; Joulin et al., 2017; Liu et al., 2018b). Not only do such deep models outperform traditional machine learning methods, they also come with the benefit of not requiring difficult feature engineering. For instance, both Lample et al. (2016) and Ma and Hovy (2016) propose end-to-end models for sequence labelling task and achieve state-of-the-art results.

Orthogonal to the advances in deep learning is the effort spent on feature engineering. A representative example is the task of named entity recognition (NER), one that requires both lexical and syntactic knowledge, where, until recently, most models heavily rely on statistical sequential labelling models taking in manually engineered features (Florian et al., 2003; Chieu and Ng, 2002; Ando and Zhang, 2005). Typical features include POS and chunk tags, prefixes and suffixes, and external gazetteers, all of which represent years of accumulated knowledge in the field of computational linguistics.

The work of Collobert et al. (2011) started the trend of feature engineering-free modelling by learning internal representations of compositional components of text (e.g., word embeddings). Subsequent work has shown impressive progress through capturing syntactic and semantic knowledge with dense real-valued vectors trained on large unannotated corpora (Mikolov et al., 2013a,b; Pennington et al., 2014). Enabled by the powerful representational capacity of such embeddings and neural networks, feature engineering has largely been replaced with taking off-the-shelf pre-trained word embeddings as input, thereby making models fully end-to-end and the research focus has shifted to neural network architecture engineering.

More recently, there has been increasing recognition of the utility of linguistic features (Li et al., 2017; Chen et al., 2017; Wu et al., 2017; Liu et al., 2018a) where such features are integrated to improve model performance. Inspired by this, taking NER as a case study, we investigate the utility of hand-crafted features in deep learning models, challenging conventional wisdom in an attempt to refute the utility of manually-engineered features. Of particular interest to this paper is the work by Ma and Hovy (2016) where they
introduce a strong end-to-end model combining a bi-directional Long Short-Term Memory (Bi-LSTM) network with Convolutional Neural Network (CNN) character encoding in a Conditional Random Field (CRF). Their model is highly capable of capturing not only word- but also character-level features. We extend this model by integrating an auto-encoder loss, allowing the model to take hand-crafted features as input and re-construct them as output, and show that, even with such a highly competitive model, incorporating linguistic features is still beneficial. Perhaps the closest to this study is the works by Ammar et al. (2014) and Zhang et al. (2017), who show how CRFs can be framed as auto-encoders in unsupervised or semi-supervised settings.

With our proposed model, we achieve strong performance on the CoNLL 2003 English NER shared task with an F1 of 91.89, significantly outperforming an array of competitive baselines. We conduct an ablation study to better understand the impacts of each manually-crafted feature. Finally, we further provide an in-depth analysis of model performance when trained with varying amount of data and show that the proposed model is highly competent with only 60% of the training set.

2 Methodology

In this section, we first outline the model architecture, then the manually crafted features, and finally how they are incorporated into the model.

2.1 Model Architecture

We build on a highly competitive sequence labelling model, namely Bi-LSTM-CNN-CRF, first introduced by Ma and Hovy (2016). Given an input sequence of \( x = \{x_1, x_2, \ldots, x_T\} \) of length \( T \), the model is capable of tagging each input with a predicted label \( \hat{y} \), resulting in a sequence of \( \hat{y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\} \) closely matching the gold label sequence \( y = \{y_1, y_2, \ldots, y_T\} \). Here, we extend the model by incorporating an auto-encoder loss taking hand-crafted features as input/output, thereby forcing the model to preserve crucial information stored in such features and allowing us to evaluate the impacts of each feature on model performance. Specifically, our model, referred to as Neural-CRF+AE, consists of four major components: (1) a character-level CNN (char-CNN); (2) a word-level bi-directional LSTM (Bi-LSTM); (3) a conditional random field (CRF); and (4) an auto-encoder auxiliary loss. An illustration of the model architecture is presented in Figure 1.

### Char-CNN

Previous studies (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016) have demonstrated that CNNs are highly capable of capturing character-level features. Here, our character-level CNN is similar to that used in Ma and Hovy (2016) but differs in that we use a ReLU activation (Nair and Hinton, 2010).1

### Bi-LSTM

We use a Bi-LSTM to learn contextual information of a sequence of words. As inputs to the Bi-LSTM, we first concatenate the pre-trained embedding of each word \( w_i \) with its character-level representation \( c_{w_i} \) (the output of the char-CNN) and a vector of manually crafted features \( f_i \) (described in Section 2.2):

\[
\vec{h}_i = \text{LSTM}(\vec{h}_{i-1}, [w_i; c_{w_i}; f_i])
\]

\[
\hat{y}_i = \text{LSTM}(\hat{y}_{i+1}, [w_i; c_{w_i}; f_i])
\]

where \([;]\) denotes concatenation. The outputs of the forward and backward pass of the Bi-LSTM is then concatenated \( h_i = [\vec{h}_i; \hat{h}_i] \) to form the output of the Bi-LSTM, where dropout is also applied.

### CRF

For sequence labelling tasks, it is intuitive and beneficial to utilise information carried between neighbouring labels to predict the best sequence of labels for a given sentence. Therefore,

---

1While the hyperbolic tangent activation function results in comparable performance, the choice of ReLU is mainly due to faster convergence.
we employ a conditional random field layer (Lafferty et al., 2001) taking as input the output of the Bi-LSTM $h_i$. Training is carried out by maximising the log probability of the gold sequence:

$$L_{CRF} = \log p(y|x)$$

while decoding can be efficiently performed with the Viterbi algorithm.

**Auto-encoder loss.** Alongside sequence labelling as the primary task, we also deploy, as auxiliary tasks, three auto-encoders for reconstructing the hand-engineered feature vectors. To this end, we add multiple fully independent connected dense layers, all taking as input the Bi-LSTM output $h_i$ with each responsible for reconstructing a particular type of feature:

$$\hat{f}_t^i = \sigma(W^f_t h_i)$$

where $\sigma$ is the sigmoid activation function, $t$ denotes the type of feature, and $W^f_t$ is a trainable parameter matrix. More formally, we define the auto-encoder loss as:

$$L_{AE}^t = \sum_{i=0}^{T} \text{XEntropy}(f^i_t, \hat{f}^i_t).$$

**Model training.** Training is carried out by optimising the joint loss:

$$\mathcal{L} = L_{CRF} + \sum_t \lambda_t L_{AE}^t,$$

where, in addition to $L_{CRF}$, we also add the auto-encoder loss, weighted by $\lambda_t$. In all our experiments, we set $\lambda_t$ to 1 for all $t$.

### 2.2 Hand-crafted Features

We consider three categories of widely used features: (1) POS tags; (2) word shape; and (3) gazetteers and present an example in Table 1. While POS tags carry syntactic information regarding sentence structure, the word shape feature focuses on a more fine-grained level, encoding character-level knowledge to complement the loss of information caused by embedding lookup, such as capitalisation. Both features are based on the implementation of spaCy.\(^2\) For the gazetteer feature, we focus on PERSON and LOCATION and compile a list for each. The PERSON gazetteer is collected from U.S. census 2000, U.S. census 2010 and DBpedia whereas GeoNames is the main source for LOCATION, taking in both official and alternative names. All the tokens on both lists are then filtered to exclude frequently occurring common words.\(^3\) Each category is converted into a one-hot sparse feature vector $f^i_t$ and then concatenated to form a multi-hot vector $f_i = [f_i^{POS}; f_i^{shape}; f_i^{gazetteer}]$ for the $i$-th word. In addition, we also experimented with including the label of the incoming dependency edge to each word as a feature, but observed performance deterioration on the development set. While we still study and analyse the impacts of this feature in Table 3 and Section 3.2, it is excluded from our model configuration (not considered as part of $f_i$ unless indicated otherwise).

### 3 Experiments

In this section, we present our experimental setup and results for name entity recognition over the CoNLL 2003 English NER shared task dataset (Tjong Kim Sang and De Meulder, 2003).

#### 3.1 Experimental Setup

**Dataset.** We use the CoNLL 2003 NER shared task dataset, consisting of 14,041/3,250/3,453 sentences in the training/development/test set respectively, all extracted from Reuters news articles during the period from 1996 to 1997. The dataset is annotated with four categories of name entities: PERSON, LOCATION, ORGANIZATION and MISC. We use the IOBES tagging scheme, as previous study have shown that this scheme provides a modest improvement to the model performance (Ratinov and Roth, 2009; Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016).

---

\(^{2}\)Gazetteer data is included in the code release.

---

| x | U.N. | official | Ekeus | heads | for | Baghdad |.
|---|------|---------|-------|-------|-----|---------|
| POS | NNP | NN | NNP | VBZ | IN | NNP |.
| Word shape | X.X. | xxxx | compound | ROO | prep | Xxxx |.
| Dependency tags | O | O | compound | O | O | LOC | O.
| Gazetteer | B-ORG | O | B-PER | O | O | B-LOC | O.

Table 1: Example sentence (top), showing the different types of linguistic features used in this work as additional inputs and auxiliary outputs (middle), and its labelling (bottom).
Model configuration. Following the work of Ma and Hovy (2016), we initialise word embeddings with GloVe (Pennington et al., 2014) (300-dimensional, trained on a 6B-token corpus). Character embeddings are 30-dimensional and randomly initialised with a uniform distribution in the range \([-\sqrt{\frac{3}{\text{dim}}}, +\sqrt{\frac{3}{\text{dim}}}]\). Parameters are optimised with stochastic gradient descent (SGD) with an initial learning rate of \(\eta = 0.015\) and momentum of 0.9. Exponential learning rate decay is applied every 5 epochs with a factor of 0.8. To reduce the impact of exploding gradients, we employ gradient clipping at 5.0 (Pascanu et al., 2013).

We train our models on a single GeForce GTX TITAN X GPU. With the above hyper-parameter setting, training takes approximately 8 hours for a full run of 40 epochs.

Evaluation. We measure model performance with the official CoNLL evaluation script and report span-level named entity F-score on the test set using early stopping based on the performance on the validation set. We report average F-scores and standard deviation over 5 runs for our model.

Baseline. In addition to reporting a number of prior results of competitive baseline models, as listed in Table 2, we also re-implement the Bi-LSTM-CNN-CRF model by Ma and Hovy (2016) (referred to as Neural-CRF in Table 2) and report its average performance.

3.2 Results

The experimental results are presented in Table 2. Observe that Neural-CRF+AE, trained either on the training set only or with the addition of the development set, achieves substantial improvements in F-score in both settings, superior to all but one of the benchmark models, highlighting the utility of hand-crafted features incorporated with the proposed auto-encoder loss. Compared against the Neural-CRF, a very strong model in itself, our model significantly improves performance, showing the positive impact of our technique for exploiting manually-engineered features. Although Peters et al. (2018) report a higher F-score using their ELMo embedding technique, our approach here is orthogonal, and accordingly we would expect a performance increase if we were to incorporate their ELMo representations into our model.

Ablation Study. To gain a better understanding of the impacts of each feature, we perform an ablation study and present the results in Table 3. We observe performance degradation when eliminating POS, word shape and gazetteer features, showing that each feature contributes to NER performance beyond what is learned through deep learning alone. Interestingly, the contribution of gazetteers is much less than that of the other features, which is likely due to the noise introduced in the matching process, with many incorrectly identified false positives.

Including features based on dependency tags into our model decreases the performance slightly. This might be a result of our simple implementation (as illustrated in Table 1), which does not include dependency direction, nor parent-child relationships.

Next, we investigate the impact of different means of incorporating manually-engineered features into the model. To this end, we experiment with three configurations with features as: (1) input only; (2) output only (equivalent to multi-task learning); and (3) both input and output (Neural-CRF+AE) and present the results in Table 4. Simply using features as either input or output only improves model performance slightly, but insignificantly so. It is only when features are incorporated with the proposed auto-encoder loss do we observe a significant performance boost.

<table>
<thead>
<tr>
<th>Model</th>
<th>F(_\text{i})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chiu and Ng (2002)</td>
<td>88.31</td>
</tr>
<tr>
<td>Florian et al. (2003)</td>
<td>88.76</td>
</tr>
<tr>
<td>Ando and Zhang (2005)</td>
<td>89.31</td>
</tr>
<tr>
<td>Collobert et al. (2011)</td>
<td>89.59</td>
</tr>
<tr>
<td>Huang et al. (2015)</td>
<td>90.10</td>
</tr>
<tr>
<td>Passos et al. (2014)</td>
<td>90.90</td>
</tr>
<tr>
<td>Lampel et al. (2016)</td>
<td>90.94</td>
</tr>
<tr>
<td>Luo et al. (2015)</td>
<td>91.20</td>
</tr>
<tr>
<td>Ma and Hovy (2016)</td>
<td>91.21</td>
</tr>
<tr>
<td>Yang et al. (2017)</td>
<td>91.62</td>
</tr>
<tr>
<td>Peters et al. (2018)</td>
<td>90.15</td>
</tr>
<tr>
<td>Peters et al. (2018)+ELMo</td>
<td>(92.22 \pm 0.10)</td>
</tr>
<tr>
<td>Neural-CRF†</td>
<td>91.06 (\pm 0.18)</td>
</tr>
<tr>
<td>Neural-CRF+AE†*</td>
<td>91.89 (\pm 0.23)</td>
</tr>
<tr>
<td>Ratinov and Roth (2009)†</td>
<td>90.80</td>
</tr>
<tr>
<td>Chiu and Nichols (2016)†</td>
<td>91.62</td>
</tr>
<tr>
<td>Neural-CRF+AE† ‡</td>
<td>(92.29 \pm 0.20)</td>
</tr>
</tbody>
</table>

Table 2: NER Performance on the CoNLL 2003 English NER shared task test set. Bold highlights best performance. † marks models trained on both the training and development sets. ‡ indicates average performance over 5 runs. * indicates statistical significance on the test set against Neural-CRF by two-sample Student’s t-test at level \(\alpha = 0.05\).
Table 3: Ablation study. Average performance over 5 runs with standard deviation. + and – denote adding and removing a particular feature (to/from Neural-CRF+AE trained on the training set only with POS tagging, word shape and gazetteer features). ∗ indicates statistical significance on the test set against Neural-CRF+AE by two-sample Student’s t-test at level α = 0.05. Note that in this table, ∗ measures the drop in performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev $F_1$</th>
<th>Test $F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural-CRF+AE</td>
<td>94.87 (± 0.21)</td>
<td>91.89 (± 0.23)</td>
</tr>
<tr>
<td>– POS tagging ∗</td>
<td>94.78 (± 0.17)</td>
<td>91.30 (± 0.28)</td>
</tr>
<tr>
<td>– word shape ∗</td>
<td>94.83 (± 0.31)</td>
<td>91.36 (± 0.30)</td>
</tr>
<tr>
<td>– gazetteer</td>
<td>94.85 (± 0.20)</td>
<td>91.80 (± 0.19)</td>
</tr>
<tr>
<td>+ dependencies</td>
<td>94.74 (± 0.16)</td>
<td>91.66 (± 0.18)</td>
</tr>
</tbody>
</table>

Table 4: Average performance of Neural-CRF with different features configurations over 5 runs with standard deviation. Note that + input & output = Neural-CRF+AE. ∗ indicates statistical significance on the test set against Neural-CRF by two-sample Student’s t-test at level α = 0.05.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev $F_1$</th>
<th>Test $F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural-CRF</td>
<td>94.53 (± 0.21)</td>
<td>91.06 (± 0.18)</td>
</tr>
<tr>
<td>+ input</td>
<td>94.63 (± 0.23)</td>
<td>91.17 (± 0.25)</td>
</tr>
<tr>
<td>+ output</td>
<td>94.69 (± 0.22)</td>
<td>91.23 (± 0.19)</td>
</tr>
<tr>
<td>+ input &amp; output ∗</td>
<td>94.87 (± 0.21)</td>
<td>91.89 (± 0.23)</td>
</tr>
</tbody>
</table>

Training Requirements Neural systems typically require a large amount of annotated data. Here we measure the impact of training with varying amount of annotated data, as shown in Figure 2. With the proposed model architecture, the amount of labelled training data can be drastically reduced: our model, achieves comparable performance against the baseline Neural-CRF, with as little as 60% of the training data. Moreover, as we increase the amount of training text, the performance of Neural-CRF+AE continues to improve.

Hyperparameters Three extra hyperparameters are introduced into our model, controlling the weight of the autoencoder loss relative to the CRF loss, for each feature type. Figure 3 shows the effect of each hyperparameter on test performance. Observe that setting $\lambda_i = 1$ gives strong performance, and that the impact of the gazetteer is less marked than the other two feature types. While increasing $\lambda$ is mostly beneficial, performance drops if the $\lambda$s are overly large, that is, the auto-encoder loss overwhelms the main prediction task.

4 Conclusion
In this paper, we set out to investigate the utility of hand-crafted features. To this end, we have presented a hybrid neural architecture to validate this hypothesis extending a Bi-LSTM-CNN-CRF by incorporating an auto-encoder loss to take manual features as input and then reconstruct them. On the task of named entity recognition, we show significant improvements over a collection of competitive baselines, verifying the value of such features. Lastly, the method presented in this work can also be easily applied to other tasks and models, where hand-engineered features provide key insights about the data.


Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Wu, M; Liu, F; Cohn, T

Title:
Evaluating the utility of hand-crafted features in sequence labelling

Date:
2020-01-01

Citation:

Persistent Link:
http://hdl.handle.net/11343/240906

File Description:
Published version