Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification

Heike Adel and Hinrich Schütze

Center for Information and Language Processing (CIS)
LMU Munich, Germany
heike@cis.lmu.de

Abstract

We introduce globally normalized convolutional neural networks for joint entity classification and relation extraction. In particular, we propose a way to utilize a linear-chain conditional random field output layer for predicting entity types and relations between entities at the same time. Our experiments show that global normalization outperforms a locally normalized softmax layer on a benchmark dataset.

1 Introduction

Named entity classification (EC) and relation extraction (RE) are important topics in natural language processing. They are relevant, e.g., for populating knowledge bases or answering questions from text, such as “Where does X live?”

Most approaches consider the two tasks independent from each other or treat them as a sequential pipeline by first applying a named entity recognition tool and then classifying relations between entity pairs. However, named entity types and relations are often mutually dependent. If the types of entities are known, the search space of possible relations between them can be reduced and vice versa. This can help, for example, to resolve ambiguities, such as in the case of “Mercedes”, which can be a person, organization and location. However, knowing that in the given context, it is the second argument for the relation “live in” helps concluding that it is a location. Therefore, we propose a single neural network (NN) for both tasks. In contrast to joint training and multitask learning, which calculate task-wise costs, we propose to learn a joint classification layer which is globally normalized on the outputs of both tasks. In particular, we train the NN parameters based on the loss of a linear-chain conditional random field (CRF) (Lafferty et al., 2001). CRF layers for NNs have been introduced for token-labeling tasks like named entity recognition (NER) or part-of-speech tagging (Collobert et al., 2011; Lample et al., 2016; Andor et al., 2016). Instead of labeling each input token as in previous work, we model the joint entity and relation classification problem as a sequence of length three for the CRF layer. In particular, we identify the types of two candidate entities (words or short phrases) given a sentence (we call this entity classification to distinguish it from the token-labeling task NER) as well as the relation between them.

To the best of our knowledge, this architecture for combining entity and relation classification in a single neural network is novel. Figure 1 shows an example of how we model the task: For each sentence, candidate entities are identified. Every possible combination of candidate entities (query entity pair) then forms the input to our model which predicts the classes for the two query entities as well as for the relation between them.
To sum up, our contributions are as follows: We introduce globally normalized convolutional neural networks for a sentence classification task. In particular, we present an architecture which allows us to model joint entity and relation classification with a single neural network and classify entities and relations at the same time, normalizing their scores globally. Our experiments confirm that a CNN with a CRF output layer outperforms a CNN with locally normalized softmax layers. Our source code is available at http://cistern.cis.lmu.de.

2 Related Work

Some work on joint entity and relation classification uses distant supervision for building their own datasets, e.g., (Yao et al., 2010; Yaghoobzadeh et al., 2016). Other studies, which are described in more detail in the following, use the “entity and relation recognition” (ERR) dataset from (Roth and Yih, 2004, 2007) as we do in this paper. Roth and Yih (2004) develop constraints and use linear programming to globally normalize entity types and relations. Giuliano et al. (2007) use entity type information for relation extraction but do not train both tasks jointly. Kate and Mooney (2010) train task-specific support vector machines and develop a card-pyramid parsing algorithm to jointly model both tasks. Miwa and Sasaki (2014) use the same dataset but model the tasks as a table filling problem (see Section 4.2). Their model uses both a local and a global scoring function. Recently, Gupta et al. (2016) apply recurrent neural networks to fill the table. They train them in a multitask fashion. Previous work also uses a variety of linguistic features, such as part-of-speech tags. In contrast, we use convolutional neural networks and only word embeddings as input. Furthermore, we are the first to adopt global normalization of neural networks for this task.

Several studies propose different variants of non-neural CRF models for information extraction tasks but model them as token-labeling problems (Sutton and McCallum, 2006; Sarawagi et al., 2004; Culotta et al., 2006; Zhu et al., 2005; Peng and McCallum, 2006). In contrast, we propose a simpler linear-chain CRF model which directly connects entity and relation classes instead of assigning a label to each token of the input sequence. This is more similar to the factor graph by Yao et al. (2010) but computationally simpler. Xu and Sarikaya (2013) also apply a CRF layer on top of continuous representations obtained by a CNN. However, they use it for a token labeling task (semantic slot filling) while we apply the model to a sentence classification task, motivated by the fact that a CNN creates single representations for whole phrases or sentences.

3 Model

3.1 Modeling Context and Entities

Figure 2 illustrates our model.

Input. Given an input sentence and two query entities, our model identifies the types of the entities and the relation between them; see Figure 1. The input tokens are represented by word embeddings trained on Wikipedia with word2vec (Mikolov et al., 2013). For identifying the class of an entity $e_k$, the model uses the context to its left, the words constituting $e_k$ and the context to its right. For classifying the relation between two entities $e_i$ and $e_j$, the sentence is split into six parts: left of $e_i$, left of $e_j$, right of $e_i$, left of $e_j$, right of $e_j$. For the example sentence in Figure 1 and the entity pair (“Anderson”, “chief”), the context split is: [Anderson], “chief”); the context split is: [Anderson], “chief”); the context split is: [Anderson], “chief”); the context split is: [Anderson], “chief”); the context split is: [Anderson], “chief”); the context split is: [Anderson], “chief”);

Sentence Representation. For representing the different parts of the input sentence, we use convolutional neural networks (CNNs). CNNs are suitable for RE since a relation is usually expressed by the semantics of a whole phrase or sentence. Moreover, they have proven effective for RE in previous work (Vu et al., 2016). We train one CNN layer for convolving the entities and one for the contexts. Using two CNN layers instead of one gives our model more flexibility. Since entities are usually shorter than contexts, the filter width for entities can be smaller than for contexts. Furthermore, this architecture simplifies changing the entity representation from words to characters in future work.

After convolution, we apply $k$-max pooling for both the entities and the contexts and concatenate the results. The concatenated vector $c_z \in \mathbb{R}^{C_z}$, $z \in \{EC, RE\}$ is forwarded to a task-specific hidden layer of size $H_z$ which learns patterns across the different input parts:

$$h_z = \tanh(V_z^T c_z + b_z)$$

(1)

The ERR dataset we use provides boundaries for entities to concentrate on the classification task (Roth and Yih, 2004).
with weights \( V_z \in \mathbb{R}^{C_z \times H_z} \) and bias \( b_z \in \mathbb{R}^{H_z} \).

### 3.2 Global Normalization Layer

For global normalization, we adopt the linear-chain CRF layer by Lample et al. (2016).\(^2\) It expects scores for the different classes as input. Therefore, we apply a linear layer first which maps the representations \( h_z \in \mathbb{R}^{H_z} \) to a vector \( v_z \) of the size of the output classes \( N = N_{EC} + N_{RE} \):

\[
v_z = W_z^T h_z
\]

with \( W_z \in \mathbb{R}^{H_z \times N} \). For a sentence classification task, the input sequence for the CRF layer is not inherently clear. Therefore, we propose to model the joint entity and relation classification problem with the following sequence of scores (cf., Figure 2):

\[
d = [v_{EC}(e_1), v_{RE}(r_{12}), v_{EC}(e_2)]
\]

with \( r_{ij} \) being the relation between \( e_i \) und \( e_j \). Thus, we approximate the joint probability of entity types \( T_{e_1}, T_{e_2} \) and relations \( R_{e_1e_2} \) as follows:

\[
P(T_{e_1}R_{e_1e_2}T_{e_2}) \approx P(T_{e_1}) \cdot P(R_{e_1e_2}|T_{e_1}) \cdot P(T_{e_2}|R_{e_1e_2})
\]

Our intuition is that the dependence between relation and entities is stronger than the dependence between the two entities.

The CRF layer pads its input of length \( n = 3 \) with begin and end tags and computes the following score for a sequence of predictions \( y \):

\[
s(y) = \sum_{i=0}^{n} Q_{y_i,y_{i+1}} + \sum_{i=1}^{n} d_{i,y_i}
\]

with \( Q_{k,l} \) being the transition score from class \( k \) to class \( l \) and \( d_{p,q} \) being the score of class \( q \) at position \( p \) in the sequence. The scores are summed because all the variables of the CRF layer live in the log space. The matrix of transition scores \( Q \in \mathbb{R}^{(n+2) \times (n+2)} \) is learned during training.\(^3\) For training, the forward algorithm computes the scores for all possible label sequences \( Y \) to get the log-probability of the correct label sequence \( \hat{y} \):

\[
\log(p(\hat{y})) = \frac{e^{s(\hat{y})}}{\sum_{y \in Y} e^{s(y)}}
\]

For testing, Viterbi is applied to obtain the label sequence \( y^* \) with the maximum score:

\[
y^* = \arg \max_{\hat{y} \in Y} s(\hat{y})
\]

### 4 Experiments and Analysis

#### 4.1 Data and Evaluation Measure

We use the “entity and relation recognition” (ERR) dataset from (Roth and Yih, 2004)\(^4\) with the train-test split by Gupta et al. (2016). We tune the parameters on a held-out part of train. The data is labeled with entity types and relations (see Table 1). For entity pairs without a relation, we use the label N. Dataset statistics and model parameters are provided in the appendix.

Following previous work, we compute \( F_1 \) of the individual classes for EC and RE, as well as a task-wise macro \( F_1 \) score. We also report the average of scores across tasks (Avg EC+RE).

---

\(^2\)https://github.com/glample/tagger

\(^3\)2 is added because of the padded begin and end tag

\(^4\)http://cogcomp.cs.illinois.edu/page/resource_view/43
4.2 Experimental Setups

Setup 1: Entity Pair Relations. Roth and Yih (2004, 2007); Kate and Mooney (2010) train separate models for EC and RE on the ERR dataset. For RE, they only identify relations between named entity pairs. In this setup, the query entities for our model are only named entity pairs. Note that this facilitates EC in our experiments.

Setup 2: Table Filling. Following Miwa and Sasaki (2014); Gupta et al. (2016), we also model the joint task of EC and RE as a table filling task. For a sentence with length \( m \), we create a quadratic table. Cell \((i, j)\) contains the relation between word \( i \) and word \( j \) (or N for no relation). A diagonal cell \((k, k)\) contains the entity type of word \( k \). Following previous work, we only predict classes for half of the table, i.e. for \( m(m + 1)/2 \) cells. Figure 3 shows the table for the example sentence from Figure 1. In this setup, each cell \((i, j)\) with \( i \neq j \) is a separate input query to our model. Our model outputs a prediction for cell \((i, j)\) (the relation between \( i \) and \( j \)) and predictions for cells \((i, i)\) and \((j, j)\) (the types of \( i \) and \( j \)). To fill the diagonal with entity classes, we aggregate all predictions for the particular entity by using majority vote. Section 4.4 shows that the individual predictions agree with the majority vote in almost all cases.

Setup 3: Table Filling Without Entity Boundaries. The table from setup 2 includes one row/column per multi-token entity, utilizing the given entity boundaries of the ERR dataset. In order to investigate the impact of the entity boundaries on the classification results, we also consider another table filling setup where we ignore the boundaries and assign one row/column per token. Note that this setup is also used by prior work on table filling (Miwa and Sasaki, 2014; Gupta et al., 2016). For evaluation, we follow Gupta et al. (2016) and score a multi-token entity as correct if at least one of its comprising cells has been classified correctly.

Comparison. The most important difference between setup 1 and setup 2 is the number of entity pairs with no relation (test set: \( \approx 3k \) for setup 1, \( \approx 121k \) for setup 2). This makes setup 2 more challenging. The same holds for setup 3 which considers the same number of entity pairs with no relation as setup 2. To cope with this, we randomly subsample negative instances in the train set of setup 2 and 3. Setup 3 considers the most query entity pairs in total since multi-token entities are split into their comprising tokens. However, setup 3 represents a more realistic scenario than setup 1 or setup 2 because in most cases, entity boundaries are not given. In order to apply setup 1 or 2 to another dataset without entity boundaries, a preprocessing step, such as entity boundary recognition or chunking would be required.

4.3 Experimental Results

Table 1 shows the results of our globally normalized model in comparison to the same model with locally normalized softmax output layers (one for EC and one for RE). For setup 1, the CRF layer performs comparable or better than the softmax layer. For setup 2 and 3, the improvements are more apparent. We assume that the model can benefit more from global normalization in the case of table filling because it is the more challenging setup. The comparison between setup 2 and setup 3 shows that the entity classification suffers from not given entity boundaries (in setup 3). A reason could be that the model cannot convolve the token embeddings of the multi-token entities anymore when computing the entity representation (context B and D in Figure 2). Nevertheless, the relation classification performance is comparable in setup 2 and setup 3. This shows that the model can internally account for potentially wrong entity classification results due to missing entity boundaries.

The overall results (Avg EC+RE) of the CRF are better than the results of the softmax layer for all three setups. To sum up, the improvements of the linear-chain CRF show that (i) joint EC and RE benefits from global normalization and (ii) our way of creating the input sequence for the CRF for joint EC and RE is effective.

Comparison to State of the Art. Table 2 shows our results in the context of state-of-the-art results: (Roth and Yih, 2007), (Kate and Mooney, 2010),...
Table 1: $F_1$ results for entity classification (EC) and relation extraction (RE) in the three setups

<table>
<thead>
<tr>
<th>Model</th>
<th>S</th>
<th>Feats</th>
<th>EC</th>
<th>RE</th>
<th>EC+RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R &amp; Y 2007</td>
<td>1</td>
<td>yes</td>
<td>85.8</td>
<td>58.1</td>
<td>72.0</td>
</tr>
<tr>
<td>K &amp; M 2010</td>
<td>1</td>
<td>yes</td>
<td>91.7</td>
<td>62.2</td>
<td>77.0</td>
</tr>
<tr>
<td>Ours (NN CRF)</td>
<td>1</td>
<td>no</td>
<td><strong>92.1</strong></td>
<td>65.3</td>
<td><strong>78.7</strong></td>
</tr>
<tr>
<td>Ours (NN CRF)</td>
<td>2</td>
<td>no</td>
<td><strong>88.2</strong></td>
<td>62.1</td>
<td><strong>75.2</strong></td>
</tr>
<tr>
<td>M &amp; S 2014</td>
<td>3</td>
<td>yes</td>
<td>92.3</td>
<td>71.0</td>
<td>81.7</td>
</tr>
<tr>
<td>G et al. 2016 (1)</td>
<td>3</td>
<td>yes</td>
<td>92.4</td>
<td>69.9</td>
<td>81.2</td>
</tr>
<tr>
<td>G et al. 2016 (2)</td>
<td>3</td>
<td>no</td>
<td><strong>88.8</strong></td>
<td>58.3</td>
<td><strong>73.6</strong></td>
</tr>
<tr>
<td>Ours (NN CRF)</td>
<td>3</td>
<td>no</td>
<td>82.1</td>
<td>62.5</td>
<td>72.3</td>
</tr>
</tbody>
</table>

Table 2: Comparison to state of the art (S: setup)

4.5 Analysis of CRF Transition Matrix

To analyze the CRF layer, we extract which transitions have scores above 0.5. Figure 4 shows that the layer has learned correct correlations between entity types and relations.

5 Conclusion and Future Work

In this paper, we presented the first study on global normalization of neural networks for a sentence classification task without transforming it into a token-labeling problem. We trained a convolutional neural network with a linear-chain conditional random field output layer on joint entity and relation classification and showed that it outperformed using a locally normalized softmax layer.

An interesting future direction is the extension of the linear-chain CRF to jointly normalize all predictions for table filling in a single model pass. Furthermore, we plan to verify our results on other datasets in future work.

Acknowledgments

Heike Adel is a recipient of the Google European Doctoral Fellowship in Natural Language Processing and this research is supported by this fellowship. This work was also supported by DFG (SCHU 2246/4-2).

Since we only input named entities into our model.
References


A Dataset Statistics

Table 3 provides statistics of the data composition in our different setups which are described in the paper. The N class of setup 2 and setup 3 has been subsampled in the training and development set as described in the paper.

<table>
<thead>
<tr>
<th></th>
<th>train</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peop</td>
<td>1146</td>
<td>224</td>
<td>321</td>
</tr>
<tr>
<td>Org</td>
<td>596</td>
<td>189</td>
<td>198</td>
</tr>
<tr>
<td>Loc</td>
<td>1204</td>
<td>335</td>
<td>427</td>
</tr>
<tr>
<td>Other</td>
<td>427</td>
<td>110</td>
<td>125</td>
</tr>
<tr>
<td>O</td>
<td>20338</td>
<td>5261</td>
<td>6313</td>
</tr>
<tr>
<td>Located_in</td>
<td>243</td>
<td>66</td>
<td>94</td>
</tr>
<tr>
<td>Work_for</td>
<td>243</td>
<td>82</td>
<td>76</td>
</tr>
<tr>
<td>OrgBased_in</td>
<td>239</td>
<td>106</td>
<td>105</td>
</tr>
<tr>
<td>Live_in</td>
<td>342</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>Kill</td>
<td>203</td>
<td>18</td>
<td>47</td>
</tr>
<tr>
<td>N (setup 1)</td>
<td>10742</td>
<td>2614</td>
<td>3344</td>
</tr>
<tr>
<td>N (setup 2/3)</td>
<td>123453</td>
<td>30757</td>
<td>120716</td>
</tr>
</tbody>
</table>

Table 3: Dataset statistics for our different experimental setups

Note that the sum of numbers of relation labels is slightly different to the numbers reported in (Roth and Yih, 2004). According to their website https://cogcomp.cs.illinois.edu/page/resource_view/43, they have updated the corpus.

B Hyperparameters

<table>
<thead>
<tr>
<th>Setup</th>
<th>Output layer</th>
<th>nkC</th>
<th>nkE</th>
<th>hC</th>
<th>hE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>softmax</td>
<td>500</td>
<td>100</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>softmax</td>
<td>500</td>
<td>100</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>softmax</td>
<td>500</td>
<td>100</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>CRF</td>
<td>200</td>
<td>50</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>CRF</td>
<td>500</td>
<td>100</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>CRF</td>
<td>500</td>
<td>100</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 4: Hyperparameter optimization results

Table 4 provides the hyperparameters we optimized on dev (nkC: number of convolutional filters for the CNN convolving the contexts, nkE: number of convolutional filters for the CNN convolving the entities; hC: number of hidden units for creating the final context representation, hE: number of hidden units for creating the final entity representation).

For all models, we use a filter width of 3 for the context CNN and a filter width of 2 for the entity CNN (tuned in prior experiments and fixed for the optimization of the parameters in Table 4). For training, we apply gradient descent with a batch size of 10 and an initial learning rate of 0.1. When the performance on dev decreases, we halve the learning rate. The model is trained with early stopping on dev, with a maximum number of 20 epochs. We apply L2 regularization with $\lambda = 10^{-3}$. 

1730