Joint entity recognition and relation extraction as a multi-head selection problem

Giannis Bekoulis*, Johannes Deleu, Thomas Demeester, Chris Develder

Ghent University – imec, IDLab, Department of Information Technology,
Technologiepark Zwijnaarde 15, 9052 Ghent, Belgium

Abstract

State-of-the-art models for joint entity recognition and relation extraction strongly rely on external natural language processing (NLP) tools such as POS (part-of-speech) taggers and dependency parsers. Thus, the performance of such joint models depends on the quality of the features obtained from these NLP tools. However, these features are not always accurate for various languages and contexts. In this paper, we propose a joint neural model which performs entity recognition and relation extraction simultaneously, without the need of any manually extracted features or the use of any external tool. Specifically, we model the entity recognition task using a CRF (Conditional Random Fields) layer and the relation extraction task as a multi-head selection problem (i.e., potentially identify multiple relations for each entity). We present an extensive experimental setup, to demonstrate the effectiveness of our method using datasets from various contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch). Our model outperforms the previous neural models that use automatically extracted features, while it performs within a reasonable margin of feature-based neural models, or even beats them.

Keywords: entity recognition, relation extraction, multi-head selection, joint model, sequence labeling

*Corresponding author

Email addresses: giannis.bekoulis@ugent.be (Giannis Bekoulis), johannes.deleu@ugent.be (Johannes Deleu), thomas.demeester@ugent.be (Thomas Demeester), chris.develder@ugent.be (Chris Develder)

1. Introduction

The goal of the entity recognition and relation extraction is to discover relational structures of entity mentions from unstructured texts. It is a central problem in information extraction since it is critical for tasks such as knowledge base population and question answering.

The problem is traditionally approached as two separate subtasks, namely (i) named entity recognition (NER) (Nadeau & Sekine, 2007) and (ii) relation extraction (RE) (Bach & Badaskar, 2007), in a pipeline setting. The main limitations of the pipeline models are: (i) error propagation between the components (i.e., NER and RE) and (ii) possible useful information from the one task is not exploited by the other (e.g., identifying a *Works for* relation might be helpful for the NER module in detecting the *type* of the two entities, i.e., *PER*, *ORG* and vice versa). On the other hand, more recent studies propose to use joint models to detect entities and their relations overcoming the aforementioned issues and achieving state-of-the-art performance (Li & Ji, 2014; Miwa & Sasaki, 2014).

Pipeline缺点：
1.两个模块之间错误传播
2.一个模块中有用的信息不会被另一个模块充分利用，如works for关系决定了一个实体类型是PER，一个实体类型是ORG

早期的联合模型依赖于人工提取特征

The previous joint models heavily rely on hand-crafted features. Recent advances in neural networks alleviate the issue of manual feature engineering, but some of them still depend on NLP tools (e.g., POS taggers, dependency parsers). Miwa & Bansal (2016) propose a Recurrent Neural Network (RNN)-based joint model that uses a bidirectional sequential LSTM (Long Short Term Memory) to model the entities and a tree-LSTM that takes into account dependency tree information to model the relations between the entities. The dependency information is extracted using an external dependency parser. Similarly, in the work of Li et al. (2017) for entity and relation extraction from biomedical text, a model which also uses tree-LSTMs is applied to extract dependency information. Gupta et al. (2016) propose a method that relies on RNNs but uses a lot of hand-crafted features and additional NLP tools to extract features such as POS-tags, etc. Adel & Schütze (2017) replicate the context around the entities with Convolutional Neural Networks (CNNs). Note that the aforementioned works examine pairs of entities for relation extraction, rather than modeling the whole sentence directly. This means that relations of other pairs of entities in the same sentence —

以上说的几种方法都是基于局部实体关系，没有考虑整个句子，没有考虑到其他的实体对也会对关系产生影响

which could be helpful in deciding on the relation *type* for a particular pair — are not taken into account. Katiyar & Cardie (2017) propose a neural joint model based on LSTMs where they model the whole sentence at once, but still they do not have a principled way to deal with multiple relations. Bekoulis et al. (2018) introduce a quadratic scoring layer to model the two tasks simultaneously. The limitation of this approach is that only a single relation can be assigned to a token, while the time complexity for the entity recognition task is increased compared to the standard approaches with linear complexity.

In this work, we focus on a new general purpose joint model that performs the two tasks of entity recognition and relation extraction simultaneously, and that can handle multiple relations together. Our model achieves state-of-the-art performance in a number of different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch) without relying on any manually engineered features nor additional NLP tools. In summary, our proposed model (which will be detailed next in Section 3) solves several shortcomings that we identified in related works (Section 2) for joint entity recognition and relation extraction: (i) our model does not rely on external NLP tools nor hand-crafted features, (ii) entities and relations within the same text fragment (typically a sentence) are extracted simultaneously, where (iii) an entity can be involved in multiple relations at once.

1.不依赖于任何NLP工具和人工特征
2.同时抽取多种关系的实体

Specifically, the model of Miwa & Bansal (2016) depends on dependency parsers, which perform particularly well on specific languages (i.e., English) and contexts (i.e., news). Yet, our ambition is to develop a model that generalizes well in various setups, therefore using only automatically extracted features that are learned during training. For instance, Miwa & Bansal (2016) and Li et al. (2017) use exactly the same model in different contexts, i.e., news (ACE04) and biomedical data (ADE), respectively. Comparing our results to the ADE dataset, we obtain a 1.8% improvement on the NER task and ∼3% on the RE task. On the other hand, our model performs within a reasonable margin (∼0.6% in the NER task and ∼1% on the RE task) on the ACE04 dataset without the use of pre-calculated features. This shows that the model of Miwa & Bansal (2016) strongly relies on the features extracted by the dependency parsers and cannot generalize well into different contexts where dependency parser features are weak.

Comparing to Adel & Schütze (2017), we train our model by modeling all the entities and the relations of the sentence at once. This type of inference is beneficial in obtaining information about neighboring entities and relations instead of just examining a pair of entities each time. Finally, we solve the underlying problem of the models proposed by Katiyar & Cardie (2017) and Bekoulis et al. (2017), who essentially assume classes (i.e., relations) to be mutually exclusive: we solve this by phrasing the relation extraction component as a multi-label prediction problem.[1]

To demonstrate the effectiveness of the proposed method, we conduct the largest experimental evaluation to date (to the best of our knowledge) in jointly performing both entity recognition and relation extraction (see Section 4 and Section 5), using different datasets from various domains (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch). Specifically, we apply our method to four datasets, namely ACE04 (news), Adverse Drug Events (ADE), Dutch Real Estate Classifieds (DREC) and CoNLL'04 (news). Our method outperforms all state-of-the-art methods that do not rely on any additional features or tools, while performance is very close (or even better in the biomedical dataset) compared to methods that do exploit hand-engineered features or NLP tools.

2. Related work

The tasks of entity recognition and relation extraction can be applied either one by one in a pipeline setting (Fundel et al., 2007; Gurulingappa et al., 2012a; Bekoulis et al., 2017) or in a joint model (Miwa & Sasaki, 2014; Miwa & Bansal, 2016; Bekoulis et al., 2018). In this section, we present related work for each task (i.e., named entity recognition and relation extraction) as well as prior work into joint entity and relation extraction.

[1]Note that another difference is that we use a CRF layer for the NER part, while Katiyar & Cardie (2017) uses a softmax and Bekoulis et al. (2017) uses a quadratic scoring layer; see further, when we discuss performance comparison results in Section 5.

2.1. Named entity recognition

In our work, NER is the first task which we solve in order to address the end-to-end relation extraction problem. A number of different methods for the NER task that are based on hand-crafted features have been proposed, such as CRFs (Lafferty et al., 2001), Maximum Margin Markov Networks (Taskar et al., 2003) and support vector machines (SVMs) for structured output (Tsochantaridis et al., 2004), to name just a few. Recently, deep learning methods such as CNN- and RNN-based models have been combined with CRF loss functions (Collobert et al., 2011; Huang et al., 2015; Lample et al., 2016; Ma & Hovy, 2016) for NER. These methods achieve state-of-the-art performance on publicly available NER datasets without relying on hand-crafted features.

2.2. Relation extraction

We consider relation extraction as the second task of our joint model. The main approaches for relation extraction rely either on hand-crafted features (Zelenko et al., 2003; Kambhatla, 2004) or neural networks (Socher et al., 2012; Zeng et al., 2014). Feature-based methods focus on obtaining effective hand-crafted features, for instance defining kernel functions (Zelenko et al., 2003; Culotta & Sorensen, 2004) and designing lexical, syntactic, semantic features, etc. (Kambhatla, 2004; Rink & Harabagiu, 2010). Neural network models have been proposed to overcome the issue of manually designing hand-crafted features leading to improved performance. CNN- (Zeng et al., 2014; Xu et al., 2015a; dos Santos et al., 2015) and RNN-based (Socher et al., 2013; Zhang & Wang, 2015; Xu et al., 2015b) models have been introduced to automatically extract lexical and sentence level features leading to a deeper language understanding. Vu et al. (2016) combine CNNs and RNNs using an ensemble scheme to achieve state-of-the-art results.

2.3. Joint entity and relation extraction

Entity and relation extraction includes the task of (i) identifying the entities (described in Section 2.1) and (ii) extracting the relations among them (described in Section 2.2). Feature-based joint models (Kate & Mooney, 2010; Yang & Cardie, 2013; Li & Ji,

2014; Miwa & Sasaki, 2014) have been proposed to simultaneously solve the entity recognition and relation extraction (RE) subtasks. These methods rely on the availability of NLP tools (e.g., POS taggers) or manually designed features and thus (i) require additional effort for the data preprocessing, (ii) perform poorly in different application and language settings where the NLP tools are not reliable, and (iii) increase the computational complexity. In this paper, we introduce a joint neural network model to overcome the aforementioned issues and to automatically perform end-to-end relation extraction without the need of any manual feature engineering or the use of additional NLP components.

Neural network approaches have been considered to address the problem in a joint setting (end-to-end relation extraction) and typically include the use of RNNs and CNNs (Miwa & Bansal, 2016; Zheng et al., 2017; Li et al., 2017). Specifically, Miwa & Bansal (2016) propose the use of bidirectional tree-structured RNNs to capture dependency tree information (where parse trees are extracted using state-of-the-art dependency parsers) which has been proven beneficial for relation extraction (Xu et al., 2015a,b). Li et al. (2017) apply the work of Miwa & Bansal (2016) to biomedical text, reporting state-of-the-art performance for two biomedical datasets. Gupta et al. (2016) propose the use of a lot of hand-crafted features along with RNNs. Adel & Schütze (2017) solve the entity classification task (which is different from NER since in entity classification the boundaries of the entities are known and only the *type* of the entity should be predicted) and relation extraction problems using an approximation of a global normalization objective (i.e., CRF): they replicate the context of the sentence (left and right part of the entities) to feed one entity pair at a time to a CNN for relation extraction. Thus, they do not simultaneously infer other potential entities and relations within the same sentence. Katiyar & Cardie (2017) and Bekoulis et al. (2018) investigate RNNs with attention for extracting relations between entity mentions without using any dependency parse tree features. Different from Katiyar & Cardie (2017), in this work, we frame the problem as a multi-head selection problem by using a sigmoid loss to obtain multiple relations and a CRF loss for the NER component. This way, we are able to independently predict classes that are not mutually exclusive, instead of assigning equal probability values among the tokens. We overcome the issue of addi-

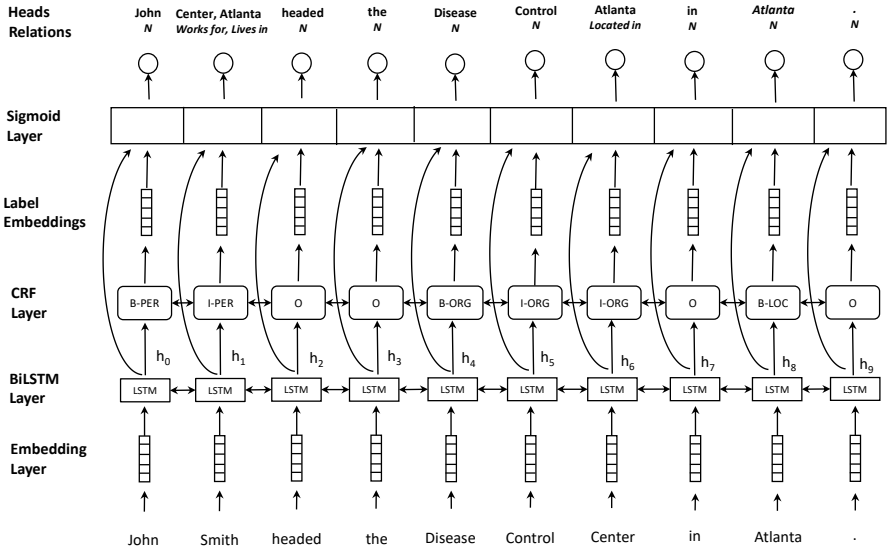
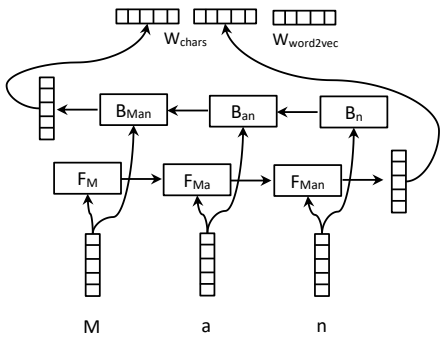


Figure 1: The multi-head selection model for joint entity and relation extraction. The input of our model is the words of the sentence which are then represented as word vectors (i.e., embeddings). The BiLSTM layer extracts a more complex representation for each word. Then the CRF and the sigmoid layers are able to produce the outputs for the two tasks. The outputs for each token (e.g., Smith) are: (i) an entity recognition label (e.g., I-*PER*) and (ii) a set of tuples comprising the head tokens of the entity and the types of relations between them (e.g., {(Center, *Works for*), (Atlanta, *Lives in*)}).

tional complexity described by Bekoulis et al. (2018), by dividing the loss functions into a NER and a relation extraction component. Moreover, we are able to handle multiple relations instead of just predicting single ones, as was described for the application of structured real estate advertisements of Bekoulis et al. (2018).

3. Joint model

In this section, we present our multi-head joint model illustrated in Fig. 1. The model is able to simultaneously identify the entities (i.e., types and boundaries) and all the possible relations between them at once. We formulate the problem as a multi-head selection problem extending previous work (Zhang et al., 2017; Bekoulis et al., 2018) as described in Section 2.3. By multi-head, we mean that any particular entity may be involved in multiple relations with other entities. The basic layers of the model, shown in Fig. 1, are: (i) embedding layer, (ii) bidirectional sequential LSTM (BiLSTM) layer, (iii) CRF layer and the (iv) sigmoid scoring layer. In Fig. 1, an example sentence from



embedding中用了char_embedding获取单词前缀和后缀信息

Figure 2: Embedding layer in detail. The characters of the word "Man" are represented by character vectors (i.e., embeddings) that are learned during training. The character embeddings are fed to a BiLSTM and the two final states (forward and backward) are concatenated. The vector w_{chars} is the character-level representation of the word. This vector is then further concatenated to the word-level representation $w_{word2vec}$ to obtain the complete word embedding vector.

the CoNLL04 dataset is presented. The input of our model is a sequence of tokens (i.e., words of the sentence) which are then represented as word vectors (i.e., word embeddings). The BiLSTM layer is able to extract a more complex representation for each word that incorporates the context via the RNN structure. Then the CRF and the sigmoid layers are able to produce the outputs for the two tasks. The outputs for each token (e.g., Smith) are twofold: (i) an entity recognition label (e.g., I-*PER*, denoting the token is inside a named entity of *type PER*) and (ii) a set of tuples comprising the head tokens of the entity and the types of relations between them (e.g., {(Center, *Works for*), (Atlanta, *Lives in*)}). Since we assume token-based encoding, we consider only the last token of the entity as head of another token, eliminating redundant relations. For instance, there is a *Works for* relation between entities "John Smith" and "Disease Control Center". Instead of connecting all tokens of the entities, we connect only "Smith" with "Center". Also, for the case of no relation, we introduce the "N" label and we predict the token itself as the head.

3.1. Embedding layer

应用了char_embedding,对中文没有用

Given a sentence $w = w_1, ..., w_n$ as a sequence of tokens, the word embedding layer is responsible to map each token to a word vector ($w_{word2vec}$). We use pre-trained word embeddings using the Skip-Gram word2vec model (Mikolov et al., 2013).

In this work, we also use character embeddings since they are commonly applied to neural NER (Ma & Hovy, 2016; Lample et al., 2016). This type of embeddings is able to capture morphological features such as prefixes and suffixes. For instance, in the Adverse Drug Events (ADE) dataset, the suffix "toxicity" can specify an *adverse drug event* entity such as "neurotoxicity" or "hepatotoxicity" and thus it is very informative. Another example might be the Dutch suffix "kamer" ("room" in English) in the Dutch Real Estate Classifieds (DREC) dataset which is used to specify the *space* entities "badkamer" ("bathroom" in English) and "slaapkamer" ("bedroom" in English). Character-level embeddings are learned during training, similar to Ma & Hovy (2016) and Lample et al. (2016). In the work of Lample et al. (2016), character embeddings lead to a performance improvement of up to 3% in terms of NER F_1 score. In our work, by incorporating character embeddings, we report in Table 2 an increase of $\sim$2% overall F_1 scoring points. For more details, see Section 5.2.

Figure 2 illustrates the neural architecture for word embedding generation based on its characters. The characters of each word are represented by character vectors (i.e., embeddings). The character embeddings are fed to a BiLSTM and the two final states (forward and backward) are concatenated. The vector w_{chars} is the character-level representation of the word. This vector is then further concatenated to the word-level representation $w_{word2vec}$ to obtain the complete word embedding vector.

3.2. Bidirectional LSTM encoding layer 双向LSTM编码

RNNs are commonly used in modeling sequential data and have been successfully applied in various NLP tasks (Sutskever et al., 2014; Lample et al., 2016; Miwa & Bansal, 2016). In this work, we use multi-layer LSTMs, a specific kind of RNNs which are able to capture long term dependencies well (Bengio et al., 1994; Pascanu et al., 2013). We employ a BiLSTM which is able to encode information from left to right (past to future) and right to left (future to past). This way, we can combine bidirectional information for each word by concatenating the forward ($\vec{h_i}$) and the backward ($\overleftarrow{h_i}$) output at timestep i. The BiLSTM output at timestep i can be written as:

$$h_i = [\vec{h_i}; \overleftarrow{h_i}], \quad i = 0, ..., n \tag{1}$$

3.3. Named entity recognition

We formulate the entity identification task as a sequence labeling problem, similar to previous work on joint learning models (Miwa & Bansal, 2016; Li et al., 2017; Katiyar & Cardie, 2017) and named entity recognition (Lample et al., 2016; Ma & Hovy, 2016) using the BIO (Beginning, Inside, Outside) encoding scheme. Each entity consists of multiple sequential tokens within the sentence and we should assign a tag for every token in the sentence. That way we are able to identify the entity arguments (start and end position) and its *type* (e.g., *ORG*). To do so, we assign the B-*type* (beginning) to the first token of the entity, the I-*type* (inside) to every other token within the entity and the O tag (outside) if a token is not part of an entity. Fig. 1 shows an example of the BIO encoding tags assigned to the tokens of the sentence. In the CRF layer, one can observe that we assign the B-*ORG* and I-*ORG* tags to indicate the beginning and the inside tokens of the entity "Disease Control Center", respectively. On top of the BiLSTM layer, we employ either a softmax or a CRF layer to calculate the most probable entity tag for each token. We calculate the score of each token w_i for each entity tag:

通过BIO编码，确定实体的始终位置

$$s^{(e)}(h_i) = V^{(e)} f(U^{(e)} h_i + b^{(e)}) \tag{2}$$

where the superscript (e) is used for the notation of the NER task, $f(\cdot)$ is an element-wise activation function (i.e., *relu*, *tanh*), $V^{(e)} \in \mathbb{R}^{p \times l}$, $U^{(e)} \in \mathbb{R}^{l \times 2d}$, $b^{(e)} \in \mathbb{R}^{l}$, with d as the hidden size of the LSTM, p the number of NER tags (e.g., B-*ORG*) and l the layer width. We calculate the probabilities of all the candidate tags for a given token w_i as $\Pr(tag \mid w_i) = \text{softmax}(s(h_i))$ where $\Pr(tag \mid w_i) \in \mathbb{R}^{p}$. In this work, we employ the softmax approach only for the entity classification (EC) task (which is similar to NER) where we need to predict only the entity *types* (e.g., *PER*) for each token assuming boundaries are given. The CRF approach is used for the NER task which includes both entity *type* and boundaries recognition.

e是任务标记，d是LSTM隐层大小，p是标签个数，l是层宽也就是batch_size

In the softmax approach, we assign entity *types* to tokens in a greedy way at prediction time (i.e., the selected tag is just the highest scoring tag over all possible set of tags). Although assuming an independent tag distribution is beneficial for entity classification tasks (e.g., POS tagging), this is not the case when there are strong de-

pendencies between the tags. Specifically, in NER, the BIO tagging scheme forces several restrictions (e.g., B-*LOC* cannot be followed by I–*PER*). The softmax method allows local decisions (i.e., for the tag of each token w_i) even though the BiLSTM captures information about the neighboring words. Still, the neighboring tags are not taken into account for the tag decision of a specific token. For example, in the entity "John Smith", tagging "Smith" as *PER* is useful for deciding that "John" is B-*PER*. To this end, for NER, we use a linear-chain CRF, similar to Lample et al. (2016) where an improvement of $\sim$1% F_1 NER points is reported when using CRF. In our case, with the use of CRF we also report a $\sim$1% overall performance improvement as observed in Table 2 (see Section 5.2). Assuming the word vector w, a sequence of score vectors $s_1^{(e)}, ..., s_n^{(e)}$ and a vector of tag predictions $y_1^{(e)}, ..., y_n^{(e)}$, the linear-chain CRF score is defined as:

$$S\left(y_1^{(e)}, \ldots, y_n^{(e)}\right) = \sum_{i=0}^{n} s^{(e)}_{i, y_i^{(e)}} + \sum_{i=1}^{n-1} T_{y_i^{(e)}, y_{i+1}^{(e)}} \tag{3}$$

where $S \in \mathbb{R}$, $s^{(e)}_{i, y_i^{(e)}}$ is the score of the predicted tag for token w_i, T is a square transition matrix in which each entry represents transition scores from one tag to another. $T \in \mathbb{R}^{(p+2)\times(p+2)}$ because $y_0^{(e)}$ and $y_n^{(e)}$ are two auxiliary tags that represent the starting and the ending tags of the sentence, respectively. Then, the probability of a given sequence of tags over all possible tag sequences for the input sentence w is defined as:

$$\Pr\left(y_1^{(e)}, \ldots, y_n^{(e)} \;\middle|\; w\right) = \frac{\mathrm{e}^{S(y_1^{(e)}, \ldots, y_n^{(e)})}}{\sum\limits_{\tilde{y_1}^{(e)}, \ldots, \tilde{y_n}^{(e)}} \mathrm{e}^{S(\tilde{y_1}^{(e)}, \ldots, \tilde{y_n}^{(e)})}} \tag{4}$$

We apply Viterbi to obtain the tag sequence $\hat{y}^{(e)}$ with the highest score. We train both the softmax (for the EC task) and the CRF layer (for NER) by minimizing the cross-entropy loss $\mathcal{L}_{\text{NER}}$. We also use the entity tags as input to our relation extraction layer by learning label embeddings, motivated by Miwa & Bansal (2016) where an improvement of 2% F_1 is reported (with the use of label embeddings). In our case, label embeddings lead to an increase of 1% F_1 score as reported in Table 2 (see Section 5.2). The input to the next layer is twofold: the output states of the LSTM and the learned label embedding representation, encoding the intuition that knowledge of named enti-

ties can be useful for relation extraction. During training, we use the gold entity tags, while at prediction time we use the predicted entity tags as input to the next layer. The input to the next layer is the concatenation of the hidden LSTM state h_i with the label embedding g_i for token w_i:

$$z_i = [h_i; g_i], \quad i = 0, ..., n \tag{5}$$

3.4. Relation extraction as multi-head selection

In this subsection, we describe the relation extraction task, formulated as a multi-head selection problem (Zhang et al., 2017; Bekoulis et al., 2018). In the general formulation of our method, each token w_i can have multiple heads (i.e., multiple relations with other tokens). We predict the tuple $(\hat{y}_i, \hat{c}_i)$ where $\hat{y}_i$ is the vector of heads and $\hat{c}_i$ is the vector of the corresponding relations for each token w_i. This is different for the previous standard head selection for dependency parsing method (Zhang et al., 2017) since (i) it is extended to predict multiple heads and (ii) the decisions for the heads and the relations are jointly taken (i.e., instead of first predicting the heads and then in a next step the relations by using an additional classifier). Given as input a token sequence w and a set of relation labels $\mathcal{R}$, our goal is to identify for each token w_i, $i \in \{0, ..., n\}$ the vector of the most probable heads $\hat{y}_i \subseteq w$ and the vector of the most probable corresponding relation labels $\hat{r}_i \subseteq \mathcal{R}$. We calculate the score between tokens w_i and w_j given a label r_k as follows:

每个词与其他的词有多种关系

1.预测多个head，
2.head和relation的决策是同时进行

$$s^{(r)}(z_j, z_i, r_k) = V^{(r)} f(U^{(r)} z_j + W^{(r)} z_i + b^{(r)}) \tag{6}$$

where the superscript (r) is used for the notation of the relation task, $f(\cdot)$ is an element-wise activation function (i.e., *relu*, *tanh*), $V^{(r)} \in \mathbb{R}^l$, $U^{(r)} \in \mathbb{R}^{l \times (2d+b)}$, $W^{(r)} \in \mathbb{R}^{l \times (2d+b)}$, $b^{(r)} \in \mathbb{R}^l$, d is the hidden size of the LSTM, b is the size of the label embeddings and l the layer width. We define

$$\Pr(head = w_j, label = r_k \mid w_i) = \sigma(s^{(r)}(z_j, z_i, r_k)) \tag{7}$$

to be the probability of token w_j to be selected as the head of token w_i with the relation label r_k between them, where $\sigma(.)$ stands for the sigmoid function. We minimize the

cross-entropy loss $\mathcal{L}_{\text{rel}}$ during training:

$$\mathcal{L}_{\text{rel}} = \sum_{i=0}^{n} \sum_{j=0}^{m} -\log \Pr(head = y_{i,j},\, relation = r_{i,j} \mid w_i) \tag{8}$$

where $y_i \subseteq w$ and $r_i \subseteq \mathcal{R}$ are the ground truth vectors of heads and associated relation labels of w_i and m is the number of relations (heads) for w_i. After training, we keep the combination of heads $\hat{y}_i$ and relation labels $\hat{r}_i$ exceeding a threshold based on the estimated joint probability as defined in Eq. (7). Unlike previous work on joint models (Katiyar & Cardie, 2017), we are able to predict multiple relations considering the classes as independent and not mutually exclusive (the probabilities do not necessarily sum to 1 for different classes). For the joint entity and relation extraction task, we calculate the final objective as $\mathcal{L}_{\text{NER}} + \mathcal{L}_{\text{rel}}$.

考虑到类别独立，这个概率之和不是1

3.5. Edmonds' algorithm

Our model is able to simultaneously extract entity mentions and the relations between them. To demonstrate the effectiveness and the general purpose nature of our model, we also test it on the recently introduced Dutch real estate classifieds (DREC) dataset (Bekoulis et al., 2017) where the entities need to form a tree structure. By using thresholded inference, a tree structure of relations is not guaranteed. Thus we should enforce tree structure constraints to our model. To this end, we post-process the output of our system with Edmonds' maximum spanning tree algorithm for directed graphs (Chu & Liu, 1965; Edmonds, 1967). A fully connected directed graph $G = (V, E)$ is constructed, where the vertices V represent the last tokens of the identified entities (as predicted by NER) and the edges E represent the highest scoring relations with their scores as weights. Edmonds' algorithm is applied in cases a tree is not already formed by thresholded inference.

4. Experimental setup

4.1. Datasets and evaluation metrics

We conduct experiments on four datasets: (i) Automatic Content Extraction, ACE04 (Doddington et al., 2004), (ii) Adverse Drug Events, ADE (Gurulingappa et al., 2012b),

(iii) Dutch Real Estate Classifieds, DREC (Bekoulis et al., 2017) and (iv) the CoNLL'04 dataset with entity and relation recognition corpora (Roth & Yih, 2004). Our code is available in our github codebase.[2]

ACE04: There are seven main entity *types* namely Person (*PER*), Organization (*ORG*), Geographical Entities (*GPE*), Location (*LOC*), Facility (*FAC*), Weapon (*WEA*) and Vehicle (*VEH*). Also, the dataset defines seven relation *types*: Physical (*PHYS*), Person-Social (*PER-SOC*), Employment-Membership-Subsidiary (*EMP-ORG*), Agent-Artifact (*ART*), PER-ORG affiliation (*Other-AFF*), GPE affiliation (*GPE-AFF*), and Discourse (*DISC*). We follow the cross-validation setting of Li & Ji (2014) and Miwa & Bansal (2016). We removed DISC and did 5-fold cross-validation on the *bnews* and *nwire* subsets (348 documents). We obtained the preprocessing script from Miwa's github codebase.[3] We measure the performance of our system using micro F_1 scores, Precision and Recall on both entities and relations. We treat an entity as correct when the entity type and the region of its head are correct. We treat a relation as correct when its type and argument entities are correct, similar to Miwa & Bansal (2016) and Katiyar & Cardie (2017). We refer to this type of evaluation as *strict*.[4] We select the best hyperparameter values on a randomly selected validation set for each fold, selected from the training set (15% of the data) since there are no official train and validation splits in the work of Miwa & Bansal (2016).

CoNLL04: There are four entity *types* in the dataset (*Location*, *Organization*, *Person*, and *Other*) and five relation *types* (*Kill*, *Live in*, *Located in*, *OrgBased in* and *Work for*). We use the splits defined by Gupta et al. (2016) and Adel & Schütze (2017). The dataset consists of 910 training instances, 243 for validation and 288 for testing.[5] We measure the performance by computing the F_1 score on the test set. We adopt two evaluation settings to compare to previous work. Specifically, we perform an EC task assuming the entity boundaries are given similar to Gupta et al. (2016) and Adel &

[2] https://github.com/bekou/multihead_joint_entity_relation_extraction

[3] https://github.com/tticoin/LSTM-ER/tree/master/data/ace2004

[4] For the CoNLL04, DREC and ADE datasets, the head region covers the whole entity (start and end boundaries). The ACE04 already defines the head region of an entity.

[5] http://cistern.cis.lmu.de/globalNormalization/globalNormalization_all.zip

Schütze (2017). To obtain comparable results, we omit the entity class "Other" when computing the EC score. We score a multi-token entity as correct if at least one of its comprising token *types* is correct assuming that the boundaries are given; a relation is correct when the *type* of the relation and the argument entities are both correct. We report macro-average F_1 scores for EC and RE to obtain comparable results to previous studies. Moreover, we perform actual NER evaluation instead of just EC, reporting results using the *strict* evaluation metric.

DREC: The dataset consists of 2,318 classifieds as described in the work of Bekoulis et al. (2018). There are 9 entity *types*: *Neighborhood*, *Floor*, *Extra building*, *Subspace*, *Invalid*, *Field*, *Other*, *Space* and *Property*. Also, there are two relation classes *Part-of* and *Equivalent*. The goal is to identify important entities of a property (e.g., floors, spaces) from classifieds and structuring them into a tree format to get the structured description of the property. For the evaluation, we use 70% for training, 15% for validation and 15% as test set in the same splits as defined in Bekoulis et al. (2018). We measure the performance by computing the F_1 score on the test set. To compare our results with previous work (Bekoulis et al., 2018), we use the *boundaries* evaluation setting. In this setting, we count an entity as correct if the boundaries of the entity are correct. A relation is correct when the relation is correct and the argument entities are both correct. Also, we report results using the *strict* evaluation for future reference.

ADE: There are two *types* of entities (*drugs* and *diseases*) in this dataset and the aim of the task is to identify the *types* of entities and relate each *drug* with a *disease* (adverse drug events). There are 6,821 sentences in total and similar to previous work (Li et al., 2016, 2017), we remove $\sim$130 relations with overlapping entities (e.g., "lithium" is a drug which is related to "lithium intoxication"). Since there are no official sets, we evaluate our model using 10-fold cross-validation where 10% of the data was used as validation and 10% for test set similar to Li et al. (2017). The final results are displayed in F_1 metric as a macro-average across the folds. The dataset consists of 10,652 entities and 6,682 relations. We report results similar to previous work on this dataset using the *strict* evaluation metric.

4.2. Word embeddings

We use pre-trained word2vec embeddings used in previous work, so as to retain the same inputs for our model and to obtain comparable results that are not affected by the input embeddings. Specifically, we use the 200-dimensional word embeddings used in the work of Miwa & Bansal (2016) for the ACE04 dataset[6] trained on Wikipedia. We obtained the 50-dimensional word embeddings used by Adel & Schütze (2017)[5] trained also on Wikipedia for the CoNLL04 corpus. We use the 128-dimensional word2vec embeddings used by Bekoulis et al. (2018) trained on a large collection of 887k Dutch property advertisements[7] for the DREC dataset. Finally, for the ADE dataset, we used 200-dimensional embeddings used by Li et al. (2017) and trained on a combination of PubMed and PMC texts with texts extracted from English Wikipedia (Moen & Ananiadou, 2013)[8].

4.3. Hyperparameters and implementation details

We have developed our joint model by using Python with the TensorFlow machine learning library (Abadi et al., 2016). Training is performed using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-3}. We fix the size of the LSTM to $d = 64$ and the layer width of the neural network to $l = 64$ (both for the entity and the relation scoring layers). We use dropout (Srivastava et al., 2014) to regularize our network. Dropout is applied in the input embeddings and in between the hidden layers for both tasks. Different dropout rates have been applied but the best dropout values (0.2 to 0.4) for each dataset have been used. The hidden dimension for the character-based LSTMs is 25 (for each direction). We also fixed our label embeddings to be of size $b = 25$ for all the datasets except for CoNLL04 where the label embeddings were not beneficial and thus were not used. We experimented with *tanh* and *relu* activation functions (recall that this is the function $f(\cdot)$ from the model description). We use the

[6] http://tti-coin.jp/data/wikipedia200.bin

[7] https://drive.google.com/uc?id=1Dvibr-Ps4G_GI6eDx9bMXnJphGhH_M1z&export=download

[8] http://evexdb.org/pmresources/vec-space-models/wikipedia-pubmed-and-PMC-w2v.bin

	Settings	Pre-calculated Features	Evaluation	Entity			Relation			Overall F_1
				P	R	F_1	P	R	F_1	
ACE 04	Miwa & Bansal (2016)	✓	*strict*	80.80	82.90	81.80	48.70	48.10	48.40	65.10
	Katiyar & Cardie (2017)	✗	*strict*	81.20	78.10	79.60	46.40	45.53	45.70	62.65
	multi-head	✗	*strict*	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
CoNLL 04	Gupta et al. (2016)	✓	*relaxed*	92.50	92.10	92.40	78.50	63.00	69.90	81.15
	Gupta et al. (2016)	✗	*relaxed*	88.50	88.90	88.80	64.60	53.10	58.30	73.60
	Adel & Schütze (2017)	✗	*relaxed*	-	-	82.10	-	-	62.50	72.30
	multi-head EC	✗	*relaxed*	93.41	93.15	**93.26**	72.99	63.37	**67.01**	**80.14**
	Miwa & Sasaki (2014)	✓	*strict*	81.20	80.20	80.70	76.00	50.90	61.00	70.85
	multi-head	✗	*strict*	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
DREC	Bekoulis et al. (2018)	✗	*boundaries*	77.93	80.31	79.11	49.24	50.17	49.70	64.41
	multi-head+E	✗	*boundaries*	79.84	84.92	**82.30**	50.52	55.30	**52.81**	**67.56**
	single-head	✗	*strict*	78.80	84.26	**81.43**	50.57	54.30	**52.37**	**66.90**
	multi-head	✗	*strict*	78.97	83.98	81.39	50.00	54.73	52.26	66.83
ADE	Li et al. (2016)	✓	*strict*	79.50	79.60	79.50	64.00	62.90	63.40	71.45
	Li et al. (2017)	✓	*strict*	82.70	86.70	84.60	67.50	75.80	71.40	78.00
	multi-head	✗	*strict*	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**

Table 1: Comparison of our method (multi-head) with the state-of-the-art on the ACE04, CoNLL04, DREC and ADE datasets. The models: (i) multi-head+E (the model + the Edmond algorithm to produce a tree-structured output), (ii) single-head (the model predicts only one head per token) and (iii) multi-head EC (the model predicts only the entity classes assuming that the boundaries are given) are slight variations of the multi-head model adapted for each dataset and evaluation. The ✓and ✗ symbols indicate whether or not the models rely on any hand-crafted features or additional tools. Note that all the variations of our models do not rely on any additional features. We include here different evaluation types (*strict*, *relaxed* and *boundaries*) to be able to compare our results against previous studies. Finally, we report results in terms of Precision, Recall, F_1 for the two subtasks as well as overall F_1, averaging over both subtasks. Bold entries indicate the best result among models that only consider automatically learned features.

relu activation only in the ACE04 and *tanh* in all other datasets. We employ the technique of early stopping based on the validation set. In all the datasets examined in this study, we obtain the best hyperparameters after 60 to 200 epochs depending on the size of the dataset. We select the best epoch according to the results in the validation set. For more details about the effect of each hyperparameter to the model performance see the Appendix.

5. Results and discussion

5.1. Results

In Table 1, we present the results of our analysis. The first column indicates the considered dataset. In the second column, we denote the model which is applied (i.e., previous work and the proposed models). The proposed models are the following: (i) *multi-head* is the proposed model with the CRF layer for NER and the sigmoid loss for multiple head prediction, (ii) *multi-head+E* is the proposed model with addition of Edmonds' algorithm to guarantee a tree-structured output for the DREC dataset,

(iii) *single-head* is the proposed method but it predicts only one head per token using a softmax loss instead of a sigmoid, and (iv) *multi-head EC* is the proposed method with a softmax to predict the entity classes assuming that the boundaries are given, and the sigmoid loss for multiple head selection. Table 1 also indicates whether the different settings include hand-crafted features or features derived from NLP tools (e.g., POS taggers, dependency parsers). We use the ✓ symbol to denote that the model includes this kind of additional features and the ✗ symbol to denote that the model is only based on automatically extracted features. Note that all the variations of our model do not rely on any additional features. In the next column, we declare the type of evaluation conducted for each experiment. We include here different evaluation types to be able to compare our results against previous studies. Specifically, we use three evaluation types, namely:

(i) *Strict*: an entity is considered correct if the boundaries and the *type* of the entity are both correct; a relation is correct when the *type* of the relation and the argument entities are both correct,

(ii) *Boundaries*: an entity is considered correct if only the boundaries of the entity are correct (entity *type* is not considered); a relation is correct when the *type* of the relation and the argument entities are both correct and

(iii) *Relaxed*: we score a multi-token entity as correct if at least one of its comprising token *types* is correct assuming that the boundaries are given; a relation is correct when the *type* of the relation and the argument entities are both correct.

In the next three columns, we present the results for the entity identification task (Precision, Recall, F_1) and then (in the subsequent three columns) the results of the relation extraction task (Precision, Recall, F_1). Finally, in the last column, we report an additional F_1 measure which is the average F_1 performance of the two subtasks. We mark with bold font in Table 1, the best result for each dataset among those models that use only automatically extracted features.

Considering the results in the ACE04, we observe that our model outperforms the model of Katiyar & Cardie (2017) by $\sim$2% in both tasks. This improvement can be

explained by the use of the multi-head selection method which can naturally capture multiple relations and model them as a multi-label problem. Unlike the work of Katiyar & Cardie (2017), the class probabilities do not necessarily sum up to one since the classes are considered independent. Moreover, we use a CRF-layer to model the NER task to capture dependencies between sequential tokens. Finally, we obtain more effective word representations by using character-level embeddings. On the other hand, our model performs within a reasonable margin ($\sim$0.5% for the NER task and $\sim$1% for the RE task) compared to Miwa & Bansal (2016). This difference is explained by the fact that the model of Miwa & Bansal (2016) relies on POS tagging and syntactic features derived by dependency parsing. However, this kind of features relies on NLP tools that are not always accurate for various languages and contexts. For instance, the same model is adopted by the work of Li et al. (2017) for the ADE biomedical dataset and in this dataset our model reports more than 3% improvement in the RE task. This shows that our model is able to produce automatically extracted features which perform reasonably well in all contexts (e.g., news, biomedical).

For the CoNLL04 dataset, there are two different evaluation settings, namely *relaxed* and *strict*. In the *relaxed* setting, we perform an EC task instead of NER assuming that the boundaries of the entities are given. We adopt this setting to produce comparable results with previous studies (Gupta et al., 2016; Adel & Schütze, 2017). Similar to Adel & Schütze (2017), we present results of single models and no ensembles. We observe that our model outperforms all previous models that do not rely on complex hand-crafted features by a large margin (>4% for both tasks). Unlike these previous studies that consider pairs of entities to obtain the entity types and the corresponding relations, we model the whole sentence at once. That way, our method is able to directly infer all entities and relations of a sentence and benefit from their possible interactions that cannot be modeled when training is performed for each entity pair individually, one at a time. In the same setting, we also report the results of Gupta et al. (2016) in which they use multiple complicated hand-crafted features coming from NLP tools. Our model performs slightly better for the EC task and within a margin of 1% in terms of overall F_1 score. The difference in the overall performance is due to the fact that our model uses only automatically generated features. We also report re-

sults on the same dataset conducting NER (i.e., predicting entity types and boundaries) and evaluating using the *strict* evaluation measure, similar to Miwa & Sasaki (2014). Our results are not directly comparable to the work of Miwa & Sasaki (2014) because we use the splits provided by Gupta et al. (2016). However, in this setting we present the results from Miwa & Sasaki (2014) as reference. We report an improvement of $\sim$2% overall F_1 score, which suggests that our neural model is able to extract more informative representations compared to feature-based approaches.

We also report results for the DREC dataset, with two different evaluation settings. Specifically, we use the *boundaries* and the *strict* settings. We transform the previous results from Bekoulis et al. (2018) to the *boundaries* setting to make them comparable to our model since in their work, they report token-based F_1 score, which is not a common evaluation metric in relation extraction problems. Also, in their work, they focus on identifying only the boundaries of the entities and not the *types* (e.g., *Floor*, *Space*). In the *boundaries* evaluation, we achieve $\sim$3% improvement for both tasks. This is due to the fact that their quadratic scoring layer is beneficial for the RE task, yet complicates NER, which is usually modeled as a sequence labeling task. Moreover, we report results using the *strict* evaluation which is used in most related works. Using the prior knowledge that each entity has only one head, we can simplify our model and predict only one head each time (i.e., using a softmax loss). The difference between the single and the multi-head models is marginal ($<$0.1% for both tasks). This shows that our model (multi-head) can adapt to various environments, even if the setting is single head (in terms of the application, and thus also in both training and test data).

Finally, we compare our model with previous work (Li et al., 2016, 2017) on the ADE dataset. The previous models (Li et al., 2016, 2017) both use hand-crafted features or features derived from NLP tools. However, our model is able to outperform both models using the *strict* evaluation metric. We report an improvement of $\sim$2% in the NER and $\sim$3% in the RE tasks, respectively. The work of Li et al. (2017) is similar to Miwa & Bansal (2016) and strongly relies on dependency parsers to extract syntactic information. A possible explanation for the better result obtained from our model is that the pre-calculated syntactic information obtained using external tools either is not so accurate or important for biomedical data.

Settings	Entity			Relation			Overall F_1
	P	R	F_1	P	R	F_1	
Multi-head	81.01	81.31	81.16	50.14	44.48	47.14	64.15
−Label embeddings	80.61	80.91	80.77	50.00	42.92	46.18	63.48
−Character embeddings	80.42	79.52	79.97	49.06	41.62	45.04	62.50
−CRF loss	80.47	81.50	80.98	47.34	42.84	44.98	62.98

Table 2: Ablation tests on the ACE04 test dataset.

5.2. Analysis of feature contribution

We conduct ablation tests on the ACE04 dataset reported in Table 2 to analyze the effectiveness of the various parts of our joint model. The performance of the RE task decreases ($\sim$1% in terms of F_1 score) when we remove the label embeddings layer and only use the LSTM hidden states as inputs for the RE task. This shows that the NER labels, as expected, provide meaningful information for the RE component.

Removing character embeddings also degrades the performance of both NER ($\sim$1%) and RE ($\sim$2%) tasks by a relatively large margin. This illustrates that composing words by the representation of characters is effective, and our method benefits from additional information such as capital letters, suffixes and prefixes within the token (i.e., its character sequences).

Finally, we conduct experiments for the NER task by removing the CRF loss layer and substituting it with a softmax. Assuming independent distribution of labels (i.e., softmax) leads to a slight decrease in the F_1 performance of the NER module and a $\sim$2% decrease in the performance of the RE task. This happens because the CRF loss is able to capture the strong tag dependencies (e.g., I-*LOC* cannot follow B-*PER*) that are present in the dataset instead of just assuming that the tag decision for each token is independent from tag decisions of neighboring tokens.

6. Conclusion

In this work, we present a joint neural model to simultaneously extract entities and relations from textual data. Our model comprises a CRF layer for the entity recognition task and a sigmoid layer for the relation extraction task. Specifically, we model the relation extraction task as a multi-head selection problem since one entity can have

multiple relations. Previous models on this task rely heavily on external NLP tools (i.e., POS taggers, dependency parsers). Thus, the performance of these models is affected by the accuracy of the extracted features. Unlike previous studies, our model produces automatically generated features rather than relying on hand-crafted ones, or existing NLP tools. Given its independence from such NLP or other feature generating tools, our approach can be easily adopted for any language and context. We demonstrate the effectiveness of our approach by conducting a large scale experimental study. Our model is able to outperform neural methods that automatically generate features while the results are marginally similar (or sometimes better) compared to feature-based neural network approaches.

As future work, we aim to explore the effectiveness of entity pre-training for the entity recognition module. This approach has been proven beneficial in the work of Miwa & Bansal (2016) for both the entity and the relation extraction modules. In addition, we are planning to explore a way to reduce the calculations in the quadratic relation scoring layer. For instance, a straightforward way to do so is to use in the sigmoid layer only the tokens that have been identified as entities.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (pp. 265–283). Berkeley, CA, USA.

Adel, H., & Schütze, H. (2017). Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics.

Bach, N., & Badaskar, S. (2007). A review of relation extraction. *Literature review for Language and Statistics II*, .

Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2017). Reconstructing the house from the ad: Structured prediction on real estate classifieds. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 2, Short Papers)* (pp. 274–279). Valencia, Spain.

Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2018). An attentive neural architecture for joint segmentation and parsing and its application to real estate ads. *Expert Systems with Applications*, *102*, 100 – 112. doi:10.1016/j.eswa.2018.02.031.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Transactions on neural networks*, *5*(2), 157–166. doi:10.1109/72.279181.

Chu, Y.-J., & Liu, T.-H. (1965). On shortest arborescence of a directed graph. *Scientia Sinica*, *14*, 1396–1400.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Culotta, A., & Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (pp. 423–429). Barcelona, Spain. doi:10.3115/1218955.1219009.

Doddington, G. R., Mitchell, A., Przybocki, M. A., Ramshaw, L. A., Strassel, S., & Weischedel, R. M. (2004). The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proceedings Fourth International Conference on Language Resources and Evaluation* (p. 1). Lisbon, Portugal volume 2.

Edmonds, J. (1967). Optimum branchings. *Journal of research of the National Bureau of Standards*, *71B*(4), 233–240.

Fundel, K., Kffner, R., & Zimmer, R. (2007). Relex-relation extraction using dependency parse trees. *Bioinformatics*, *23*(3), 365–371. doi:10.1093/bioinformatics/btl616.

Gupta, P., Schütze, H., & Andrassy, B. (2016). Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (pp. 2537–2547).

Gurulingappa, H., MateenRajpu, A., & Toldo, L. (2012a). Extraction of potential adverse drug events from medical case reports. *Journal of Biomedical Semantics*, *3*(1), 1–15. doi:10.1186/2041-1480-3-15.

Gurulingappa, H., Rajput, A. M., Roberts, A., Fluck, J., Hofmann-Apitius, M., & Toldo, L. (2012b). Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, *45*(5), 885 – 892. doi:https://doi.org/10.1016/j.jbi.2012.04.008.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, .

Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics on Interactive poster and demonstration sessions*. Barcelona, Spain. doi:10.3115/1219044.1219066.

Kate, R. J., & Mooney, R. (2010). Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the 14th Conference on Computational Natural Language Learning* (pp. 203–212). Uppsala, Sweden: Association for Computational Linguistics.

Katiyar, A., & Cardie, C. (2017). Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada.

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*. San Diego, USA.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning* (pp. 282–289). San Francisco, USA: Morgan Kaufmann.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 260–270). San Diego, California.

Li, F., Zhang, M., Fu, G., & Ji, D. (2017). A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinformatics*, *18*(1), 1–11. doi:10.1186/s12859-017-1609-9.

Li, F., Zhang, Y., Zhang, M., & Ji, D. (2016). Joint models for extracting adverse drug events from biomedical text. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 2838–2844). New York, USA: IJCAI/AAAI Press.

Li, Q., & Ji, H. (2014). Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 402–412). Baltimore, USA.

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1064–1074). Berlin, Germany.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems* (pp. 3111–3119). Nevada, United States: Curran Associates, Inc.

Miwa, M., & Bansal, M. (2016). End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the As-*

sociation for Computational Linguistics (Volume 1: Long Papers) (pp. 1105–1116). Berlin, Germany.

Miwa, M., & Sasaki, Y. (2014). Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1858–1869). Doha, Qatar: Association for Computational Linguistics.

Moen, S., & Ananiadou, T. S. S. (2013). Distributional semantics resources for biomedical text processing. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine* (pp. 39–43). Tokyo, Japan.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26. doi:10.1075/li.30.1.03nad.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning* (pp. 1310–1318). Atlanta, USA: JMLR.org.

Rink, B., & Harabagiu, S. (2010). Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (pp. 256–259). Los Angeles, California: Association for Computational Linguistics.

Roth, D., & Yih, W.-t. (2004). A linear programming formulation for global inference in natural language tasks. In *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)* (pp. 1–8). Boston, USA: Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/W04-2401.

dos Santos, C., Xiang, B., & Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 626–634). Beijing, China.

Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems* (pp. 926–934). Nevada, United States: Curran Associates, Inc.

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1201–1211). Jeju Island, Korea: Association for Computational Linguistics.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems* (pp. 3104–3112). Montreal, Canada: MIT Press.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. In *Proceedings of the 16th International Conference on Neural Information Processing Systems* (pp. 25–32). Bangkok, Thailand: MIT Press.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning* (pp. 104–112). Helsinki, Finland: ACM. doi:10.1145/1015330.1015341.

Vu, N. T., Adel, H., Gupta, P., & Schütze, H. (2016). Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 534–539). San Diego, California. URL: http://www.aclweb.org/anthology/N16-1065.

Xu, K., Feng, Y., Huang, S., & Zhao, D. (2015a). Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 536–540). Lisbon, Portugal: Association for Computational Linguistics. URL: http://aclweb.org/anthology/D15-1062.

Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H., & Jin, Z. (2015b). Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1785–1794). Lisbon, Portugal: Association for Computational Linguistics.

Yang, B., & Cardie, C. (2013). Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1640–1649). Sofia, Bulgaria. URL: http://www.aclweb.org/anthology/P13-1161.

Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, *3*, 1083–1106. doi:10.3115/1118693.1118703.

Zeng, D., Liu, K., Lai, S., Zhou, G., & Zhao, J. (2014). Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 2335–2344).

Zhang, D., & Wang, D. (2015). Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*, .

Zhang, X., Cheng, J., & Lapata, M. (2017). Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 1, Long Papers)* (pp. 665–676). Valencia, Spain.

Zheng, S., Hao, Y., Lu, D., Bao, H., Xu, J., Hao, H., & Xu, B. (2017). Joint entity and

relation extraction based on a hybrid neural network. *Neurocomputing*, *257*, 59 – 66. doi:10.1016/j.neucom.2016.12.075.

Appendix

In this section, we report additional results for our multi-head selection framework. Specifically, we (i) compare our model with the model of Lample et al. (2016) (i.e., optimize only over the NER task), (ii) explore several hyperparameters of the network (e.g., dropout, LSTM size, character embeddings size), and (iii) report F_1 score using different word embeddings compared to the embeddings used in previous works.

In Table 1 of the main paper, we focused on comparing our model against other *joint* models that are able to solve the two tasks (i.e., NER and relation extraction) simultaneously, mainly demonstrating superiority of phrasing the relation extraction as a multi-head selection problem (enabling the extraction of multiple relations at once). Here, in Table A1, we evaluate the performance of just the first module of our joint multi-head model: we compare the performance of the NER component of our model against the state-of-the-art NER model of Lample et al. (2016). The results indicate a marginal performance improvement of our model over Lample's NER baseline in 3 out of 4 datasets. The improvement of our model's NER part is not substantial, since (i) our NER part is almost identical to Lample's, and (ii) recent advances in NER performance among neural systems are relatively small (improvements in the order of few 0.1 F_1 points – for instance, the contribution of Ma & Hovy (2016) and Lample et al. (2016) on the CoNLL-2003 test set is 0.01% and 0.17% F_1 points, respectively). This slight improvement suggests that the interaction of the two components by sharing the underlying LSTM layer is indeed beneficial (e.g., identifying a *Works for* relation might be helpful for the NER module in detecting the *type* of the two entities, i.e., *PER*, *ORG* and vice versa). Note that improving NER in isolation was not the objective of our multi-head model, but we rather aimed to compare our model against other joint models that solve the task of entity recognition and relation identification *simultaneously*. We thus did not envision to claim or achieve state-of-the-art performance in each of the individual building blocks of our joint model.

Tables A2, A3 and A4 show the performance of our model on the test set for different values of the embedding dropout, LSTM layer dropout and the LSTM output dropout hyperparameters, respectively. Note that the hyperparameter values used for

the results in Section 5 were obtained by tuning over the development set, and these are indicated in bold face in the tables below. We vary one hyperparameter at a time in order to assess the effect of a particular hyperparameter. The main outcomes from these tables are twofold: (i) low dropout values (e.g., 0, 0.1) lead to a performance decrease in the overall F_1 score (see Table A3 where a $\sim$3% F_1 decrease is reported on the ACE04 dataset) and (ii) average dropout values (i.e., 0.2-0.4) lead to consistently similar results.

In Tables A5, A6, A7 and A8, we report results for different values of the LSTM size, the size of the character embeddings, the size of the label embeddings and the layer width of the neural network l (both for the entity and the relation scoring layers), respectively. The reported results show that different hyperparameters settings do lead to noticeable performance differences, but we do not observe any clear trend. Moreover, we have not observed any significant performance improvement that affects the overall ranking of the models as reported in Table 1. On the other hand, the results indicate that increasing (character and label) embedding size and layer dimensions leads to a slight decrease in performance for the CoNLL04 dataset. This can be explained by the fact that the CoNLL04 dataset is relatively small and using more trainable model parameters (i.e., larger hyperparameter values) can make our multi-head selection method to overfit quickly on the training set. In almost any other case, variation of the hyperparameters does not affect the ranking of the models reported in Table 1.

		Entity		
	Model	P	R	F_1
ACE 04	NER baseline	81.06	81.13	81.10
	multi-head	81.01	81.31	**81.16**
CoNLL 04	NER baseline	84.38	83.13	83.75
	multi-head	83.75	84.06	**83.90**
DREC	NER baseline	78.22	84.89	**81.42**
	multi-head	78.97	83.98	81.39
ADE	NER baseline	83.97	88.59	86.22
	multi-head	84.72	88.16	**86.40**

Table A1: Comparison of the multi-head selection model (only the NER component) against the NER baseline of Lample et al. (2016). Bold font indicates the best results for each dataset.

In the main results (see Section 5), to guarantee a fair comparison to previous work and to obtain comparable results that are not affected by the input embeddings, we use embeddings used also in prior studies. To assess the performance of our system to input

	Embedding Dropout	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	0.5	80.66	81.03	80.84	47.66	43.28	45.37	63.10
	0.4	80.97	81.39	81.18	49.90	43.55	46.51	63.84
	0.3	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	0.2	81.15	81.54	81.34	49.81	42.45	45.84	63.59
	0.1	80.86	81.06	80.96	47.74	42.92	45.20	63.08
	0	80.21	80.45	80.32	47.00	43.55	45.21	62.77
CoNLL 04	0.5	82.53	83.60	83.06	69.28	52.37	59.65	71.36
	0.4	83.66	83.04	83.35	65.17	51.42	57.48	70.42
	0.3	82.19	84.24	83.20	64.72	57.82	61.08	72.14
	0.2	84.07	84.62	84.34	71.96	54.74	62.18	73.26
	0.1	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	0	82.79	84.71	83.74	66.21	56.64	61.05	72.39
DREC	0.5	78.19	84.51	81.23	51.12	53.87	52.46	66.85
	0.4	78.47	84.73	81.48	51.87	53.57	52.71	67.10
	0.3	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	0.2	78.16	84.11	81.02	51.60	54.19	52.86	66.94
	0.1	78.83	83.34	81.02	49.38	52.69	50.99	66.01
	0	78.42	82.34	80.33	50.62	52.61	51.59	65.96
ADE	0.5	84.73	88.68	86.66	72.63	78.87	75.62	81.14
	0.4	84.51	88.21	86.32	71.93	77.90	74.80	80.56
	0.3	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	0.2	84.66	87.98	86.29	72.39	77.37	74.80	80.54
	0.1	85.10	87.43	86.25	72.91	76.71	74.76	80.51
	0	83.67	87.01	85.31	71.04	75.98	73.43	79.37

Table A2: Model performance for different embedding dropout values. Bold entries indicate the result reported in Section 5.

	LSTM Dropout	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	0.5	80.27	80.08	80.18	48.25	38.86	43.05	61.61
	0.4	81.18	81.36	81.27	50.54	42.06	45.91	63.59
	0.3	81.19	81.63	81.41	50.31	44.12	47.01	64.21
	0.2	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	0.1	81.27	81.32	81.29	48.20	41.52	44.61	62.95
	0	80.54	79.94	80.24	46.73	39.32	42.71	61.47
CoNLL 04	0.5	84.18	86.28	85.22	59.35	60.19	59.76	72.49
	0.4	84.43	85.45	84.94	63.77	62.56	63.16	74.05
	0.3	86.44	85.73	86.09	65.14	60.66	62.82	74.45
	0.2	84.73	85.91	85.32	68.02	59.48	63.46	74.39
	0.1	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	0	84.16	82.76	83.45	65.09	52.13	57.89	70.67
DREC	0.5	77.76	84.83	81.15	49.43	53.61	51.44	66.30
	0.4	78.66	83.98	81.23	50.63	54.64	52.56	66.89
	0.3	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	0.2	77.85	83.68	80.66	49.21	53.79	51.39	66.03
	0.1	78.94	83.62	81.21	51.37	53.10	52.22	66.71
	0	78.59	80.18	79.38	50.39	49.96	50.18	64.78
ADE	0.5	85.01	88.29	86.62	72.72	78.15	75.34	80.98
	0.4	84.66	88.37	86.47	72.20	78.00	74.99	80.73
	0.3	84.60	88.66	86.58	72.21	78.86	75.39	80.98
	0.2	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	0.1	84.36	87.98	86.13	72.03	77.51	74.66	80.40
	0	83.80	87.64	85.68	70.50	76.99	73.61	79.64

Table A3: Model performance for different LSTM layer dropout values. Bold entries indicate the result reported in Section 5.

	LSTM output Dropout	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	0.5	81.25	81.79	81.52	51.16	41.94	46.09	63.81
	0.4	81.23	81.70	81.47	51.44	42.77	46.71	64.09
	0.3	81.31	81.72	81.51	48.69	44.21	46.35	63.93
	0.2	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	0.1	81.01	81.12	81.07	47.55	42.82	45.06	63.07
	0	80.10	80.69	80.39	47.20	40.54	43.61	62.00
CoNLL 04	0.5	85.81	86.84	86.32	64.18	59.01	61.48	73.90
	0.4	83.27	84.89	84.08	66.07	61.37	63.63	73.85
	0.3	85.13	84.89	85.01	64.82	55.45	59.77	72.39
	0.2	84.13	84.52	84.32	66.03	57.58	61.52	72.92
	0.1	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	0	83.65	84.89	84.27	65.23	53.79	58.96	71.61
DREC	0.5	78.74	84.22	81.39	51.24	52.69	51.96	66.68
	0.4	78.45	85.20	81.69	50.34	55.45	52.77	67.23
	0.3	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	0.2	77.82	84.68	81.11	51.05	54.19	52.57	66.84
	0.1	78.84	83.75	81.22	51.74	54.75	53.20	67.21
	0	77.63	83.85	80.62	51.16	51.39	51.28	65.95
ADE	0.5	84.33	87.95	86.10	71.54	77.27	74.29	80.20
	0.4	85.16	88.16	86.63	72.87	77.81	75.26	80.95
	0.3	84.27	88.00	86.10	71.83	77.42	74.52	80.31
	0.2	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	0.1	84.65	88.04	86.31	72.38	77.49	74.85	80.58
	0	84.44	88.14	86.25	71.64	77.82	74.61	80.43

Table A4: Model performance for different LSTM output dropout values. Bold entries indicate the best result reported in Section 5.

	LSTM Size	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	32	80.99	81.25	81.12	50.33	42.60	46.14	63.63
	64	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	128	80.31	80.87	80.59	47.30	41.77	44.36	62.47
CoNLL 04	32	82.83	83.13	82.98	65.78	58.29	61.81	72.39
	64	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	128	82.43	83.04	82.73	64.86	53.79	58.81	70.77
DREC	32	77.74	85.43	81.40	50.92	52.31	51.60	66.50
	64	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	128	79.04	83.49	81.20	51.27	53.64	52.42	66.81
ADE	32	83.89	87.78	85.79	70.46	76.89	73.54	79.66
	64	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	128	84.27	87.87	86.04	71.36	76.77	73.97	80.00

Table A5: Model performance for different LSTM size values. Bold entries indicate the result reported in Section 5.

	Character Embeddings	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	15	81.02	81.57	81.29	47.87	44.78	46.27	63.78
	25	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	50	81.32	81.54	81.43	49.77	44.02	46.72	64.07
CoNLL 04	15	83.33	84.34	83.83	66.03	57.11	61.25	72.54
	25	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	50	85.15	82.95	84.04	59.84	52.61	55.99	70.01
DREC	15	79.73	84.17	81.89	52.52	55.30	53.88	67.89
	25	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	50	78.08	84.80	81.30	51.03	54.28	52.60	66.95
ADE	15	84.80	88.00	86.37	72.74	77.51	75.05	80.71
	25	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	50	84.65	88.08	86.33	72.17	77.45	74.72	80.52

Table A6: Model performance for different character embeddings size values. Bold entries indicate the result reported in Section 5.

	Label Embeddings	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	15	80.95	81.27	81.11	49.27	43.80	46.37	63.74
	25	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	50	81.17	81.61	81.39	48.01	44.48	46.18	63.78
CoNLL 04	15	84.68	83.50	84.08	62.21	56.16	59.03	71.56
	0	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	50	82.32	84.15	83.23	59.30	55.92	57.56	70.39
DREC	15	78.48	84.81	81.53	51.83	53.21	52.51	67.02
	25	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	50	78.92	84.88	81.79	51.35	53.23	52.27	67.03
ADE	15	84.47	88.18	86.29	71.93	77.49	74.61	80.45
	25	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	50	84.81	88.65	86.69	72.46	78.68	75.44	81.06

Table A7: Model performance for different label embeddings size values. Bold entries indicate the result reported in Section 5.

variations, we also report results using different word embeddings (see Table A9) (i.e., Adel & Schütze (2017); Li et al. (2017)) on the ACE04 dataset. Our results showcase that our model, even when using different word embeddings, is still performing better compared to other works that, like ours, do not rely on additional NLP tools.

	Hidden layer Size	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	32	81.01	81.02	81.02	48.81	43.26	45.87	63.44
	64	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
	128	81.30	81.32	81.31	51.58	43.68	47.30	64.31
CoNLL 04	32	82.26	84.24	83.24	65.96	59.24	62.42	72.83
	64	83.75	84.06	**83.90**	63.75	60.43	**62.04**	**72.97**
	128	82.69	83.69	83.19	64.46	55.45	59.62	71.40
DREC	32	79.66	84.23	81.89	52.42	51.45	51.93	66.91
	64	78.97	83.98	**81.39**	50.00	54.73	**52.26**	**66.83**
	128	78.35	84.47	81.30	48.53	53.08	50.70	66.00
ADE	32	84.31	88.56	86.38	71.67	78.51	74.93	80.66
	64	84.72	88.16	**86.40**	72.10	77.24	**74.58**	**80.49**
	128	84.81	88.54	86.63	72.29	78.20	75.13	80.87

Table A8: Model performance for different layer widths l of the neural network (both for the entity and the relation scoring layers). Bold entries indicate the result reported in Section 5.

Embeddings	Size	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
Miwa & Bansal (2016)	200	81.01	81.31	**81.16**	50.14	44.48	**47.14**	**64.15**
Adel & Schütze (2017)	50	82.18	79.83	80.99	49.10	41.40	44.92	62.96
Li et al. (2017)	200	81.51	81.35	81.43	46.59	44.43	45.49	63.46

Table A9: Model performance for different embeddings on the ACE04 dataset. Bold entries indicate the result reported in Section 5.