Learning Relation Classification using Attention only

Anonymous authors

Abstract

We present a novel approach for relation classification using attention mechanisms on top of token embeddings computed by ELMo. Our proposed approach consists of separating the entities taking part in a relation from its context and use these entities to pay attention to that context. We add the retrieved information on top of the entities and use that enriched abstraction to classify the relations. Because we only use the two entities enriched with information retrieved through attention, we can inspect where the added information comes from. Apart from achieving state of the art results on two established benchmarks, we also find that the two entities pay the most attention on tokens lying on the shortest dependency path between the two entities.

1. Introduction

A basic challenge in natural language understanding is relation extraction. Given that there exists enormous amount of information in natural language form, it is desirable to organize it via a structured representation. Such a representation has various applications in natural language understanding. Relation extraction provides the basic building blocks to create such structured representation, e.g. building knowledge graphs for a given text and populating knowledge bases [Grishman, 1997, Kumar, 2017, Zhang et al., 2017].

Relation extraction is about extracting triples of the form \(<entity 1, relation, entity 2>\) from unstructured text and relation classification is the task of classifying the relations, that is given a tuple \(<entity 1, entity 2>\), we want to determine the type of relation. For example, let us have a sentence ”The company produces mugs”. Let \(entity 1\) be ”company” and let \(entity 2\) be ”mugs”. Then we want to determine the relation between the two entities in that sentence, i.e. assign it a label like ”Product-Producer(e2, e1)” or ”produces(x,y)”.

Intuitively, the type of a given relation is determined by the two entities taking part in the relation and by additional context within a sentence the entities occur in. In this work, we would like to model the dependencies between this context and entities using attention mechanism before predicting the label.

Attention in general has received lots of interest in the last few years, firstly by extending RNNs and CNNs, pushing their impressive performance even further. It did not take long
until the first models arose that only focus on the attention mechanism and turning it into a stand-alone Neural Network architecture, called the Transformer [Vaswani et al., 2017]. This architecture does not rely on the sequence constraint of RNNs anymore, making it more parallelizable. Furthermore, an attention layer followed by a feedforward layer is still less complex than a CNN layer but yields shorter computational paths between words in a sequence, and thus makes information in the sequence more easily available. A side effect is that models become more interpretable given one has access to the attention scores and can highlight how much attention is paid to what.

In our case, we can inspect what the two entities of a relation pay attention to. We realized that the most important words they pay attention to often lie on the shortest dependency path between them, thus supporting the Shortest Path Hypothesis [Bunescu and Mooney, 2005]. These words also tend to lexicalize the annotated relation types to a certain degree (see Figure 4 and 5 in Section 3).

2. Related Work

In supervised relation classification, we work with a given sentence $S$ of length $n$ with tokens $w_1, w_2, \ldots, w_n$, two entities $e_1, e_2$ and their positions $p_{e1}, p_{e2}$ and an annotated label indicating what type of relation it is.

2.1 Input to the Network

The task then is to predict the relation (the label) between $e_1$ and $e_2$. Related work focuses on three different approaches for such a setting. The first approach is end-to-end systems which consume the whole sentence, the second one considers only the shortest dependency path between $e_1$ and $e_2$ and the last one works with the span starting at $p_{e1}$ until $p_{e2}$.

While working with the whole sentence as input, one faces long-distance dependencies and typically too much context not related to the relation between the two entities. To encode the two entities in the sentence, relative positions to the two entities are added to the input [Zeng et al., 2014, Nguyen and Grishman, 2015, Zhou et al., 2016, Bilan and Roth, 2018]. The challenge then lies in filtering for relevant information to determine the relation and dealing with long-distance dependencies.

One popular approach was framed by the Shortest Path Hypothesis, which states that the important information that determines the relation almost exclusively concentrates on the shortest dependency path between $e_1$ and $e_2$ [Bunescu and Mooney, 2005]. In such a setting, the only inputs to the network are tokens lying on that path [Xu et al., 2016, Zhang et al., 2017]. However, two caveats emerge: at times, necessary information to determine the relation does not lie on the shortest dependency path (e.g. negations), and also, automatically generating these parse trees is error prone and such mistakes are thus propagated into the relation classification system. A unifying approach is to consider the dependency parse for the whole sentence and given such structure, let the system learn by itself to prune out irrelevant information [Zhang et al., 2018b].

A last approach is to define a relation mention to start at $e_1$ and let it end after $e_2$ [Renslow and Neumann, 2018]. The input to the network then consists of this span. This approach can be seen as a compromise between considering only the shortest dependency path and considering the whole sentence, since the words on the shortest dependency path
mostly lie between $p_{e1}$ and $p_{e2}$, but we also consider other tokens in that span. In this work, we take a slightly modified version of this approach by defining an entity mention to start at the head of $e_1$ and to end at the head of $e_2$ and let the position of the heads be $p_{\text{head},e1}$ and $p_{\text{head},e2}$. For the SemEval-2010 training set (described in 4.1), we find that every second token in that span lies on the shortest dependency path between the two entities\(^1\), but 10% of the tokens on the shortest dependency path also lie outside that span.

### 2.2 Attention Mechanisms

We are not the first ones proposing to use attention mechanisms in relation classification. One interesting work uses the two entities to pay attention to an encoded sentence representation [Wang et al., 2016]. However, the results of this paper are not reproducible [Zhang et al., 2018a]. But, the idea of letting the entities attend over the context is worth exploring. Hence, we adapt a different model with this idea.

The Transformer encoder is already used in [Bilan and Roth, 2018]. They construct a representation of the input sequence using the Transformer encoder with only 1 layer. Afterwards, a max pooling operation over that representation yields a compressed vector which is used for classification. The two entities taking part in the relation are marked in the input sequence by indicating them via relative positions. In biological relation extraction, a combination of an attention encoder followed by a CNN produces representations for every token. These representations are projected into a head and a tail part and are mapped into a tensor through a bi-affine projection. The results get aggregated and yield relation scores for all the mentions in a sequence simultaneously. Additionally, the Transformer output is used to jointly learn NER [Verga et al., 2018].

Other work also incorporated position-aware attention mechanisms [Zhang et al., 2017], feeding information about the position and hoping that the network, having access to the entity positions, learns to pay attention to relevant information.

### 3. Model

As described earlier, we define a relation mention to start at the head of $e_1$ and to end at the head of $e_2$. We adapt a truncated Transformer version where we separate the two entity heads from the span lying between them and let these two heads pay attention to the context, that is the tokens lying between them. We illustrate our proposed architecture in Figure 2 next to the traditional Transformer architecture (Figure 1).

We encode all sentences using ELMo [Peters et al., 2018], a multi-layered bidirectional LSTM which consumes sequences character-wise. This is handy because ELMo is not restricted by a vocabulary and thus can yield embeddings for all the words. We use an ELMo model which is pretrained on language modelling tasks (1 Billion Word Benchmark) and yields deep, contextualized embeddings by returning all the internal states of the different LSTM layers. These embeddings have 1024 dimensions each. We precomputed the embeddings for all our models using the model on tensorflow hub\(^2\) and set the trainability of the ELMo model to false. We chose to precompute the embeddings because training

---

1. The dependency parses are computed using spaCy (https://spacy.io/)
2. https://tfhub.dev/google/elmo/2
the Transformer architecture successfully is cumbersome and requires large batch sizes\(^3\) for which computing the embeddings for every batch on the fly would become a bottleneck.

Afterwards, we retrieve the context of a relation, that is the word embeddings from position \(p_{head,e1} + 1\) until \(p_{head,e2} - 1\), and stack them in a context matrix. We also retrieve the two entity heads at position \(p_{head,e1}\) and \(p_{head,e2}\) and stack them in a query matrix.

Since we work with a Transformer version, we abandon recurrence or convolution altogether, and therefore we have to add information about the positions of the tokens. We do this by adding parameter-free positional embeddings, sine and cosine frequencies, on top of the word embeddings [Vaswani et al., 2017]:

\[
PE_{(pos, 2i)} = \sin (pos / 10000^{2i/d_{model}}) \\
PE_{(pos, 2i+1)} = \cos (pos / 10000^{2i/d_{model}})
\] (1)

where \(pos\) is the position, \(i\) the dimension and \(d_{model}\) the dimensionality of the input (1024 in this case). We also use relative positions; that is, we assign position 1 to the head of \(e_1\), the token following the head of \(e_1\) position 2, and so on:

\[
pos_i = pos_i - p_{head,e1} + 1
\] (2)

---

3.1 Attention

Afterwards, we perform multihead attention for both queries over the encoded context. The queries Q are the entries in the query matrix, that is the ELMo embeddings of the two entity heads with the added positional embeddings. The keys K and values V are both derived from the entries in the context matrix, that is the ELMo embeddings of the span between the two entity heads with the added positional embeddings:

\[
\text{MultiHead}(Q, K, V) = \text{ReLU}(\text{Concat}(\text{head}_1, \ldots, \text{head}_n)W^O)
\]

where \(\text{head}_i = \text{Attention}(\text{ReLU}(QW^Q_i), \text{ReLU}(KW^K_i), \text{ReLU}(VW^V_i))\) 

with Attention being defined as:

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
\]

Where we learn different parameter matrices \(W^Q_i \in \mathbb{R}^{d_{\text{model}} \times d_q}\), \(W^K_i \in \mathbb{R}^{d_{\text{model}} \times d_k}\), \(W^V_i \in \mathbb{R}^{d_{\text{model}} \times d_v}\) and \(W^O_i \in \mathbb{R}^{hd_v \times d_{\text{model}}}\) to map the embeddings into an appropriate space followed by a ReLU activation. We set \(d_{\text{model}} = 1024\) which is the dimensionality of the ELMo embeddings and \(d_q = d_k = d_v = 1024/h = 1024/2 = 512\), that is the dimensionality of the embeddings divided by the number of heads (with two heads).

To look at it from a different perspective: We map both entities into a query space and all the tokens from the context into a keys space and values space. Then we take the dot products for both entities with every token, yielding a \(2 \times n\) matrix where \(n\) is the length of the context. Each entry in that matrix is the dot product of entity \(i\) with token \(j\). We add a softmax to normalize the scores and take the weighted sum over the values (just a projection of the context words again). This is done for different attention heads (a hyperparameter) because different heads learn to watch out for different features in the sentence, as has been shown in [Vaswani et al., 2017]. In the end, the different heads are concatenated and followed by an output projection. We found that adding ReLUs after every projection increases performance, which is why we included them.

We used tensorflow as a framework to train our model. However, tensorflow requires the input sequences to be of the same length, so we pad them with zeros. When applying the attention, dot products for entities and padding tokens become 0. If all the other dot products between regular tokens and the entities are small, we would end up assigning a substantial attention mass to the padding tokens. To prevent that, we replace every dot product value between an entity and a padding token with \(-\infty\) before taking the softmax (similar to how the Transformer architecture in the decoding step performs future masking to block illegal connections).

3.2 Adding Context Information

As we described in the introduction, a relation is determined by the two entities taking part in it and additional context around the two entities. We already have information about the two entities (the embeddings of their heads) and after having applied the attention mechanism, we also find the most important additional information in the context. We
want to add this information on top of the entities and do so by applying the Add & Norm component from the original Transformer architecture.

$$x = \text{layernorm}(x + \text{subLayer}(x))$$ (5)

Where $x$ is the query matrix and the subLayer is the multihead attention over the context. If we would not apply the residual connection, we would lose all information about the entities taking part in a relation and only consider the context information.

We afterwards reshape the query matrix ($2 \times \text{dim}_{model}$) into a vector of size $2 \times \text{model}_{\text{dim}}$ and feed this vector into a feedforward layer taken from the Transformer architecture.

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$ (6)

This is a simple 2 layer feedforward network with a ReLU activation in the first layer. We again perform an Add & Norm component (equation 5) yielding a final vector which we feed into the classification layer.

We add three types of regularization for the model. After we added the positional embeddings to the ELMo representations, we add dropout. Secondly, we add dropout after every layer before feeding the layer’s output into the Add & Norm component and before the classification layer. We also employ label smoothing with $\epsilon = 0.1$. All these regularization techniques can be found in the Transformer paper [Vaswani et al., 2017].

3.3 Difference to the Standard Transformer

The architecture presented in section 3.1 includes only a subset of the modules from the full Transformer architecture (Figure 1). The missing parts are self attention over the context followed by a feedforward layer and a self attention module for the decoding step.

We experimented with including all these additional modules but we found that results are slightly worse. We think this happens because the encoder yields a contextualized representation of the input. Since we already get such a representation by ELMo, we do not need this step.

Similarly, adding self attention for the entities does not conceptually make much sense since the only information to be gained here is to look at the other entity. Only having a feedforward layer for the context does not add much which could not be learned by the projection matrices in the multihead attention module and also slightly decreases performance.

For these reasons, we decided to drop the described modules and go with a truncated version of the Transformer architecture. We also experimented with stacking several layers on top of each other, but similarly to [Bilan and Roth, 2018], we did not find any additional gain here which is why we stuck to only one layer and two attention heads.

3.4 What the Network Learned

Compared to other approaches, we incorporate a rather scarce set of features: Only the word embeddings produced by ELMo. For the TACRED dataset, we also have to incorporate NE (named entity) types and dependency path information (explained in section 4.3). We add parameter-free positional embeddings on top of the ELMo embeddings, but we believe their impact is well understood: they are sine and cosine frequencies helping the model to have a sense of position. Apart from these, we do not employ any additional features.
We take the representation of the two entities (the ELMo embeddings of the heads of the entities), we add context information on top of them and, after concatenating, use this vector to classify. However, compared to other neural network methods, we can highlight where the added information comes from. This can be done by inspecting the attention scores of the queries. Consider a sentence from the SemEval-2010 testset [Hendrickx et al., 2009] (Figure 3). The entities are "master" and "stick". Its relation type is Instrument-Agency(e2,e1), so "stick" is the instrument used by an agent, in this case "master".

The school **master** teaches the lesson with a **stick**.

Figure 3: Example sentence from SemEval-2010 testset

We encode the whole sentence using ELMo, but only consider the head of the two entities ("master" and "stick") and the span between them ("teaches the lesson with a") as input to our model. We separate the entities from the other words by stacking the embeddings of the entities in a query matrix and the embeddings of the words in the span between the two entities in a context matrix. We compute the relative positions (1 for "master", 2 for "teaches", 3 for "the" and so on) and add the positional embeddings on top. This already is the input for the attention layer.

Afterwards we perform the attention mechanism and see that both entities distribute all of their attention over the verb "teaches" and the preposition "with" (Figure 4). This corresponds exactly to the shortest dependency path between the two entities.

![Figure 4: Example of attention scores in SemEval-2010 testset](image)

If we would coin a relational phrase or describe the above relation in first-order logic, we would come up with a predicate like "teaches_with(master, stick)" or "teaches_with(x,y)".
By design choice, the arguments of the predicate are considered in our approach (the two queries), and what they learn to pay attention to is very close to the predicate or the relational phrase.

In Figure 5, we show a sentence from the TACRED development set [Zhang et al., 2017] and its relation label is "per:date_of_death". Again, we separate the head of the two entities ("Zapata" and "February") from the span between them ("…a dissident who died") and feed this as input to the model.

The Cuban government has accused the United States and the European Union of waging a smear campaign against the revolution in the wake of the death of Orlando Zapata, a dissident who died **February 23** after a prison hunger strike.

Figure 5: Example sentence from TACRED development set

In Figure 6, we show again what the two entity heads pay attention to. We find that the tokens which are paid attention to the most are ",," and "died". The token "died" describes the relation very accurately and we try to give an explanation about why the network decides to pay attention to ",," at the end of this section.

![Figure 6: Example of attention scores in TACRED development set](image)

In a quantitative analysis on the SemEval-2010 testset and the TACRED dev set, we gathered the attention scores for both queries and calculated how much attention is paid to tokens lying on the shortest dependency path (SDP) between two entities. The results are shown in Table 1. As stated in the Shortest Path Hypothesis (explained in section 2.1), we expect the model to mostly focus on tokens lying on that path. The last column in the table shows the percentage of tokens in the span between the two entities which in fact lie on the SDP between the two entities.
Learning Relation Classification using Attention only

<table>
<thead>
<tr>
<th>dataset</th>
<th>attention paid to tokens on SPD</th>
<th>tokens on SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemEval-2010 test set</td>
<td>94%</td>
<td>46%</td>
</tr>
<tr>
<td>TACRED dev set</td>
<td>30%</td>
<td>24%</td>
</tr>
</tbody>
</table>

Table 1: Attention mass paid on tokens on SPD

For the SemEval dataset, we find that if the Shortest Path Hypothesis is true, the most important information lies on the SPD between the two tokens and our network architecture learns to focus on such information, i.e. to pay attention to such tokens. Vice versa, if our network, while learning supervised relation classification, learns to pay the most attention towards tokens lying on the SPD, this seems to be evidence for the hypothesis to be true.

For the TACRED dataset, the numbers do not turn out as neatly. First of all, the spans between two entities are longer than in the SemEval dataset and not as many tokens lie on the SDP to begin with. Secondly, we find that one head of each entity learns to focus mostly on the token right after the left entity of the relation. This happens for 75% of the examples in the TACRED development set and also for the example in figure 6 where the token left of ”Zapata” is a comma. After inspecting the training set, we find that for 42% of the training examples, the token following the left entity head is a semantically rather empty token, e.g. tokens like ”,”, ”’s”, ”)”, ”(”, ”and”, ”:”, ”;”, ”’”, ”-” and ”.”. We expect the ELMo representation of such tokens to be similar to the left entity in terms of semantic content and observe an average cosine similarity of 0.4 between such tokens and the entity head before adding the positional embeddings and after adding them, we have an average cosine similarity of 0.77 between them. We think that because these two tokens often tend to be similar, the network decides to look at the token following the left entity, because it is close to looking at one of the entities taking part in the relation. Lastly, we expect the classes in the SemEval-10 set to be easier to learn given specific patterns, e.g. a third of the training examples for the class ”Cause-Effect” has the words ”caused by” in the sentence, making it easier for the network to pick up such patterns and point to them. We show additional examples of attention scores in appendix C.

4. Experiments

We tested our model on two established benchmarks in relation classification: the SemEval-2010 task 8 dataset [Hendrickx et al., 2009] and the TACRED corpus [Zhang et al., 2017]. In the following, we present the results and compare them to the current state of the art. For both datasets, we trained the final model five times and, following [Zhang et al., 2017], post results of the model with the median on the development set. In case of the SemEval-2010 dataset, we do not have a development set and just report the median of five runs on the testset. All training details can be found in appendix A. For the SemEval-10 data, we sampled the first 10% examples of every class from the training set and used this as a development set to tune hyperparameters. For TACRED, we optimized hyperparameters by tuning them on the official development set.
4.1 SemEval-2010 Dataset

The SemEval-2010 dataset is one of the benchmarks for relation classification. The dataset consists of 8000 training examples and 2717 test examples. The task focuses on the semantic relations between nominals and includes 19 classes (9 directed classes and one "Other" class), e.g. Cause-Effect, Product-Producer etc. The official metric is the macro-averaged F1 score for all classes except for the "Other" class which gets ignored during evaluation. There exists a vast amount of published results for this dataset and we only show an excerpt of what we believe to be the current state of the art (Table 2).

<table>
<thead>
<tr>
<th>System</th>
<th>Authors, Year</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRNNs with data augmentation + other features</td>
<td>Xu et al., 2016</td>
<td>86.1</td>
</tr>
<tr>
<td>Multi-Att-CNN</td>
<td>Wang et al., 2016</td>
<td>88.0</td>
</tr>
<tr>
<td>RNN with Attention &amp; Tensor Layers</td>
<td>Zhang et al., 2018</td>
<td>86.3</td>
</tr>
<tr>
<td>Graph Convolution</td>
<td>Zhang et al., 2018</td>
<td>84.8</td>
</tr>
<tr>
<td>truncated Transformer + ELMo Embeddings</td>
<td>our approach, 2018</td>
<td>86.7</td>
</tr>
</tbody>
</table>

Table 2: SemEval-2010 State of the Art

Our proposed architecture (truncated Transformer + ELMo embeddings) yields very competitive results and we report (to the best of our knowledge) the second highest results on the SemEval-2010 dataset and again refer to the discussion in footnote 3 about the best reported results on this dataset. We report the median of 5 randomly initialized runs on the test set and we also list the results for the other 4 runs in appendix B.

To measure the effect of the ELMo embeddings, we reimplemented a simple RNN baseline and trained it with GloVe [Pennington et al., 2014] and ELMo embeddings. We also tested our truncated Transformer architecture with GloVe embeddings (Table 3). The LSTM baseline is strongly inspired by [Zhang and Wang, 2015] and its performance (using GloVe embeddings) is almost equal to the results reported in the paper. It takes as input the span starting at \( e_1 \) and runs until \( e_2 \) has been consumed. It consists of a bidirectional LSTM and we perform a max-pooling operation over the hidden states of the sequence. We regularized the model by adding dropout at the input and before the classification layer. We also show results for the full Transformer architecture with GloVe and ELMo embeddings and performed ablation studies without positional embeddings and without the attention layer.

We believe that an LSTM trained with GloVe embeddings already builds a contextualized hidden representation for each token. Therefore it does not improve much if we train it with already contextualized embeddings and we observe only a small performance gain by training it with ELMo embeddings. We want to highlight that implementing the full Transformer architecture and training it with GloVe embeddings improves the performance over our proposed architecture with the same embeddings. Therefore, we expect the model to build a contextualized representation over the input via the Transformer encoder. However, if we use ELMo embeddings, we already have such a representation which can not be improved anymore and thus performance slightly drops by using the full Transformer architecture. Ablation studies without positional embeddings confirm that they are slightly

4. See [Zhang et al., 2018a] for a discussion about the reproducibility of that paper
<table>
<thead>
<tr>
<th>System</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN baseline [Zhang and Wang, 2015]</td>
<td>79.6</td>
</tr>
<tr>
<td>LSTM + GloVe embeddings</td>
<td>80.2</td>
</tr>
<tr>
<td>LSTM + ELMo embeddings</td>
<td>81.5</td>
</tr>
<tr>
<td>standard Transformer + GloVe embeddings</td>
<td>81.5</td>
</tr>
<tr>
<td>standard Transformer + ELMo embeddings</td>
<td>85.6</td>
</tr>
<tr>
<td>truncated Transformer + GloVe embeddings</td>
<td>80.9</td>
</tr>
<tr>
<td>truncated Transformer without positional embeddings</td>
<td>86.4</td>
</tr>
<tr>
<td>only classification on $e_1$ and $e_2$</td>
<td>83.2</td>
</tr>
<tr>
<td>truncated Transformer + ELMo embeddings</td>
<td>86.7</td>
</tr>
</tbody>
</table>

Table 3: Ablation studies on SemEval-10 test set

useful for the model. The second last row shows results when we only take the two contextualized representations of the entities, concatenate them and classify directly on them. This indicates that the contextualized representations of the two entities already are very indicative of the relation but that the attention layer still is useful.

4.2 TAC Relation Extraction Dataset

The TACRED (TAC Relation Extraction Dataset) is a new large-scale dataset for supervised relation classification. It contains 106,264 examples and 42 relations where 79.5% of the examples are instances of the "no_relation" class. The other 41 labels include relations like "per:city_of_birth", "per:title" and "org:founded_by". 25 of these types belong to people (all labels starting with "per:"), and 16 to organizations (labels starting with "org"). The established metric is the micro F1 score excluding the "no_relation" class. One caveat for the dataset is that entities within it can be quite long. Consider the sentence in Figure 7 where the subject of the relation spans from token 5 (aunt) until token 30 (sister) and the object is "actress Jamie Lynn":

As for being an aunt, it did take a little getting used to, says Spears, who was a little “shocked” while visiting her 17-year-old sister and actress Jamie Lynn in the hospital for the birth of her newborn daughter.

Figure 7: Example with very long entities from TACRED training set

Given the set-up of our model, we have to decide how to represent the two entities in a single vector each so that we can query the context with them. Because of long entities in the dataset (Figure 7), we decided to search for the heads of the entities and use their embedding to query the context. Whatever follows after the head of the left entity is added to the context and gets queried by the entity head in the attention layer. The tokens preceding the left entity head and the tokens following the right entity head are not considered. This dependency information is included in the distribution of the dataset.
4.3 Extension of the Model for TACRED

Since 25 of the relations belong to a subclass of type "per:" and 16 to a subclass of type "org:“, we add NE information for the two entities by concatenating their word embeddings with trainable NE embeddings of dimensionality 32 before feeding them into the attention layer. Thus, we increase the dimensionality of some of the model’s input and update the size of the output projection in the attention layer and feedforward layer to still be able to perform the residual connection. We only do this for the two entity heads and not for the tokens in between.

To counter the class imbalance (the dataset is dominated by the label "no_relation"), we balance the class weights for all classes.

\[
\text{class\_weights}_i = \frac{\text{num\_examples}}{\text{class\_counts}_i \times \text{num\_classes}}
\]  

(7)

Where \(\text{class\_counts}_i\) is the number of training examples for that class, \(\text{num\_examples}\) is the total number of training examples and \(\text{num\_classes}\) the number of classes. This yields class weights of 0.02 for the "no_relation" class and 270.33 for the "per:country\_of\_death" class. Since these values are rather extreme, we set a lower bound of 0.5 and an upper bound of 5 and replace values out of this range with the lower or upper bound, respectively.

Lastly, we perform early stopping based on the F1 score of the development set.

In Table 4, we present the result of our model and compare it to the current state of the art.

<table>
<thead>
<tr>
<th>System</th>
<th>Authors, Year</th>
<th>P</th>
<th>R</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position-aware LSTM (PA-LSTM)</td>
<td>Zhang et al., 2017</td>
<td>65.7</td>
<td>64.5</td>
<td>65.1</td>
</tr>
<tr>
<td>Contextualized Graph Convolution (C-GCN)</td>
<td>Zhang et al., 2018</td>
<td>69.9</td>
<td>63.3</td>
<td>66.4</td>
</tr>
<tr>
<td>Position-aware Self-attention (PSA)</td>
<td>Bilan &amp; Roth., 2018</td>
<td>64.6</td>
<td>68.6</td>
<td>66.5</td>
</tr>
<tr>
<td>truncated Transformer + ELMo embeddings</td>
<td>our approach, 2018</td>
<td>68.53</td>
<td>65.56</td>
<td>67.0</td>
</tr>
</tbody>
</table>

Table 4: Single model performance on the TACRED testset

We report the best single model performance (micro F1 score) on this dataset. C-GCN achieves a higher precision and PSA achieves a higher recall. Following [Zhang et al., 2017, Bilan and Roth, 2018], we also trained an ensemble of 5 models with different seeds and applied a majority voting strategy. We did this to compare our system’s performance in a setting where the prediction is based on evidence of more than one model and the results are shown in Table 5. We also report the highest results here, however only by a tiny margin.

In table 6, we report the results of ablation studies on the class weights, the positional embeddings, the NE types and the attention layer on the test set.

If we omit the class weights, we get a model with higher precision but lower recall and therefore the F1 drops to 65.2%. Again, without positional embeddings, we get slightly worse results which we already observed on the SemEval-2010 dataset. If we drop the attention module, we get a higher drop than in SemEval-2010, indicating that the context for the TACRED dataset is even more important and the entity representations are not as
Learning Relation Classification using Attention only

<table>
<thead>
<tr>
<th>System</th>
<th>Authors, Year</th>
<th>P</th>
<th>R</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble of PA-LSTMs</td>
<td>Zhang et al., 2017</td>
<td>70.1</td>
<td>64.6</td>
<td>67.2</td>
</tr>
<tr>
<td>Ensemble of PSA</td>
<td>Bilan &amp; Roth., 2018</td>
<td>65.1</td>
<td>69.7</td>
<td>67.3</td>
</tr>
<tr>
<td>C-GCN + PA-LSTM</td>
<td>Zhang et al., 2017 &amp; 2018</td>
<td>71.3</td>
<td>65.4</td>
<td>68.2</td>
</tr>
<tr>
<td>Ensemble of truncated Transformers</td>
<td>our approach, 2018</td>
<td>71.4</td>
<td>65.4</td>
<td>68.3</td>
</tr>
</tbody>
</table>

Table 5: Ensemble performance on the TACRED testset

<table>
<thead>
<tr>
<th>Ablation Studies</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>without class weights</td>
<td>65.2</td>
</tr>
<tr>
<td>without NE types</td>
<td>62.1</td>
</tr>
<tr>
<td>without positions</td>
<td>66.5</td>
</tr>
<tr>
<td>without the attention module</td>
<td>61.1</td>
</tr>
<tr>
<td>only ELMo embeddings of entities</td>
<td>58.3</td>
</tr>
<tr>
<td>truncated Transformer + ELMo embeddings</td>
<td>67.0</td>
</tr>
</tbody>
</table>

Table 6: Ablation studies on TACRED test set

indicative as in the other dataset. Without the NE types, performance also drops heavily. We expect this to happen because classes are partly conditioned on the types of the entities. If we only use the two representations of the entities yielded by ELMo and classify on them directly, we observe the largest drop in performance.

4.4 Discussion

We show on two established datasets that our architecture yields very competitive results and we achieve the best reported models for the TACRED dataset. In comparison to earlier work, we add two contributions. Firstly, we propose to split the queries from the context and let the queries learn what to pay attention to. This also allows us to track where and how information in the model flows. Secondly, we move away from tedious feature engineering for relation classification by using only contextualized representations for the SemEval-10 dataset and additionally limited external resources for the TACRED corpus. Given the nature of that corpus and our model, we have to include the dependency path information to find suitable representations for the entities (their heads) and given that labels are conditioned on NE types, we should use the NE types too. The NE types and dependency parses are already included in the distribution of the dataset.

5. Conclusion

We proposed a new approach to relation classification. Since a relation is determined by the two entities taking part in the relation and additional context in a sentence they occur in, we separate the entities from the context and enrich them with additional information found in the context. We extract this additional information by performing multihead attention using the two entities as queries and the context as keys and values. We add that retrieved
information on top of the entities. Afterwards we concatenate the two entity representations into a single vector and perform classification on top of that vector.

By using this approach and combining it with ELMo embeddings, we achieve State of the Art results on two established benchmarks in relation classification. Additionally, we believe our results to be more interpretable than existing approaches because we can easily track and highlight how information in the model is flowing. We gave two examples of what our trained models pay attention to and find that the words paid attention to by the network tend to lexicalize the relation and often lie on the shortest dependency path between the two entities. Given such patterns in manually inspected examples, we performed a quantitative analysis of how much attention is paid to tokens lying on that path and find that for the SemEval-10 dataset, the network almost exclusively focuses on such tokens. This is evidence for the Shortest Path Hypothesis and vice versa, assuming the hypothesis holds true, is evidence that our network learned what it was supposed to learn. For TACRED, we do not observe such clear overall patterns and explained what we believe the network to have learned instead.
References


Appendix A. Training Details

We used the default learning rate of 0.001 for the SemEval-10 models and trained all TACRED models with the adaptive learning rate schedule from [Vaswani et al., 2017].

\[ \text{lr}ate = d_{\text{model}}^{-0.5} \times \min(step\_num^{-0.5}, step\_num \times \text{warmup\_steps}^{-1.5}) \]  \hspace{1cm} (8)

In the following table, we list all the training details.

<table>
<thead>
<tr>
<th>parameter</th>
<th>SemEval-2010</th>
<th>TACRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>warmup_steps</td>
<td>100</td>
<td>300</td>
</tr>
<tr>
<td>num_steps</td>
<td>1000</td>
<td>3000</td>
</tr>
<tr>
<td>batch_size</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>num_heads</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>num_layers</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>model_dim</td>
<td>1024</td>
<td>1056 (ELMo + NE type embeddings)</td>
</tr>
<tr>
<td>hidden_units in MLP layer</td>
<td>4096</td>
<td>4224</td>
</tr>
<tr>
<td>embeddings_Q for each head</td>
<td>512</td>
<td>528</td>
</tr>
<tr>
<td>embeddings_K for each head</td>
<td>512</td>
<td>528</td>
</tr>
<tr>
<td>embeddings_V for each head</td>
<td>512</td>
<td>528</td>
</tr>
<tr>
<td>dropout input and before layer norm</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>dropout classification layer</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>layer norm trainable</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>max_span_length_training</td>
<td>max_span_length</td>
<td>30</td>
</tr>
<tr>
<td>class_weights</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>label_smoothing</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>early_stopping</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>use_NE_types</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>loss function</td>
<td>cross_entropy</td>
<td>cross_entropy</td>
</tr>
<tr>
<td>optimizer</td>
<td>adam</td>
<td>adam</td>
</tr>
<tr>
<td>adam_beta1</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>adam_beta2</td>
<td>0.98</td>
<td>0.999</td>
</tr>
<tr>
<td>adam_epsilon</td>
<td>1e-09</td>
<td>1e-09</td>
</tr>
</tbody>
</table>

Table 7: Training details

Appendix B. All Results for both Datasets

In table 8, we report the F1 results for all our 5 runs on the SemEval-2010 testset. In the paper, we reported the median (run 2).

And we do the same thing for the TACRED corpus (table 9). We report the testset value based on the median of the development set (run 2).
Table 8: results from all our five runs on the SemEval-2010 testset

<table>
<thead>
<tr>
<th>run</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>86.82</td>
</tr>
<tr>
<td>2</td>
<td><strong>86.66</strong></td>
</tr>
<tr>
<td>3</td>
<td>86.67</td>
</tr>
<tr>
<td>4</td>
<td>86.48</td>
</tr>
<tr>
<td>5</td>
<td>86.18</td>
</tr>
</tbody>
</table>

Table 9: results from all our five runs on the TACRED dataset

<table>
<thead>
<tr>
<th>run</th>
<th>F1 development set (%)</th>
<th>F1 testset (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>66.60</td>
<td>66.38</td>
</tr>
<tr>
<td>2</td>
<td><strong>66.91</strong></td>
<td><strong>67.02</strong></td>
</tr>
<tr>
<td>3</td>
<td>65.80</td>
<td>66.66</td>
</tr>
<tr>
<td>4</td>
<td>67.21</td>
<td>67.48</td>
</tr>
<tr>
<td>5</td>
<td>67.17</td>
<td>65.84</td>
</tr>
</tbody>
</table>

Appendix C. More Examples of Attention Scores

C.1 SemEval-2010 Test Set

In the following, we show some more examples of attention scores from the SemEval2-010 testset. Every test example is distributed among 3 consecutive lines. The first line is the true label, our prediction and whether our prediction is correct or not. The second line contains the attention scores for entity 1 (yellow), where the attention scores for the two different heads precede each token. The more attention is paid to a token by a head, the stronger its color. The numbers indicate how much attention the head pays to a certain token (rounded to one decimal point). The two entity heads are marked with a box and, because we separate them from the context, they can not pay attention to themselves (Figure 8-11).
Learning Relation Classification using Attention only

Figure 8: More examples of attention scores in SemEval-2010 testset

true label: “Content-Container(e2,e1)”; predicted label: “Content-Container(e2,e1)”; correct prediction

true label: “Instrument-Agent(e2,e1)”; predicted label: “Instrument-Agent(e2,e1)”; correct prediction

true label: “Entity-Origin(e1,e2)”; predicted label: “Entity-Origin(e1,e2)”; correct prediction

true label: “Entity-Destination(e1,e2)”; predicted label: “Entity-Destination(e1,e2)”; correct prediction

true label: “Product-Producer(e1,e2)”; predicted label: “Product-Producer(e1,e2)”; correct prediction

true label: “Component-Whole(e1,e2)”; predicted label: “Component-Whole(e1,e2)”; correct prediction

true label: “Component-Whole(e1,e2)”; predicted label: “Component-Whole(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

true label: “Cause-Effect(e1,e2)”; predicted label: “Cause-Effect(e1,e2)”; correct prediction

Figure 9: More examples of attention scores in SemEval-2010 testset
true label: "Message-Topic(e1,e2)"; predicted label: "Message-Topic(e1,e2)"; correct prediction

volume 1.0 1.0 analyzes 0.0 0.0 central concepts

true label: "Entity-Origin(e1,e2)"; predicted label: "Entity-Origin(e1,e2)"; correct prediction

defendant 0.0 0.0 was 0.0 0.0 released 1.0 1.0 from custody

defendant 0.0 0.0 was 0.0 0.0 released 1.0 1.0 from custody

true label: "Other"; predicted label: "Product-Producer(e2,e1)"; wrong prediction

plant 0.9 0.0 builds 0.0 0.0 four 0.0 0.0 - 0.0 0.0 cylinder 0.0 0.0 * 0.0 0.0 Ecotec 0.0 0.0 * 0.0 0.0 that 0.0 0.0 are 0.1 0.0 used 0.0 1.0 in vehicles

plan 0.9 0.0 builds 0.0 0.0 four 0.0 0.0 - 0.0 0.0 cylinder 0.0 0.0 * 0.0 0.0 Ecotec 0.0 0.0 * 0.0 0.0 that 0.0 0.0 are 0.1 0.0 used 0.1 1.0 in vehicles

true label: "Component-Whole(e1,e2)"; predicted label: "Content-Container(e1,e2)"; wrong prediction

cheese 0.2 0.0 as 0.2 0.0 an 0.2 0.0 element 0.4 1.0 in 0.0 0.0 a dish

cheese 0.1 0.0 as 0.1 0.0 an 0.0 0.0 element 0.7 1.0 in 0.0 0.0 a dish

true label: "Component-Whole(e1,e2)"; predicted label: "Other"; wrong prediction

castle 0.0 0.0 was 1.0 1.0 inside 0.0 0.0 a museum

castle 0.0 0.0 was 1.0 1.0 inside 0.0 0.0 a museum

true label: "Other"; predicted label: "Other"; correct prediction

clubs 0.0 0.0 are 0.0 0.0 falling 1.0 1.0 into 0.0 0.0 foreign ownership

clubs 0.0 0.0 are 0.0 0.0 falling 1.0 1.0 into 0.0 0.0 foreign ownership

true label: "Entity-Destination(e1,e2)"; predicted label: "Entity-Destination(e1,e2)"; correct prediction

man 0.0 0.0 was 0.0 0.0 carried 1.0 1.0 into 0.0 0.0 a 0.0 0.0 waiting 0.0 0.0 police car

man 0.0 0.0 was 0.0 0.0 carried 1.0 1.0 into 0.0 0.0 a 0.0 0.0 waiting 0.0 0.0 police car

Figure 10: More examples of attention scores in SemEval-2010 testset
C.2 TACRED Development Set

We also report some more attention scores from the TACRED development set (Figure 11-14). The setup is the same as described above, except that we explicitly add information about which entity we are talking about (for example "head entity 1: Flint looks at" indicates that in the following line, we show the attention scores for the head of entity 1, that is "Flint").

true label: "no_relation"; predicted label: "per:date_of_death"; wrong prediction
head entity 1: Flint looks at:

true label: "org:top_members/employees"; predicted label: "org:top_members/employees"; correct prediction
head entity 1: Overstreet looks at:

true label: "no_relation"; predicted label: "no_relation"; correct prediction
head entity 1: Rose looks at:

true label: "no_relation"; predicted label: "org:top_members/employees"; wrong prediction
head entity 1: Woman looks at:

Figure 11: More examples of attention scores in TACRED development set
Figure 12: More examples of attention scores in TACRED development set

Figure 13: More examples of attention scores in TACRED development set