Learning Relation Classification using Attention only

Anonymous authors

Abstract

We present a novel approach for relation classification using attention mechanisms only. Our proposed approach consists of separating the entities taking part in a relation from its context and use the entities to pay attention to the context. We add the retrieved information on top of the entities and use that enriched abstraction to classify the relations. Apart from reducing computational complexity and achieving state of the art results on two established benchmarks, our method also makes results more interpretable because we can highlight what part of the context the queries pay attention to.

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 (also known as "slot filling") [Zhang et al., 2017]. 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. An important
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 $w_1, w_2, ..., w_n$, two entities $e_1, e_2$ and their positions $p_{e_1}, p_{e_2}$ 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 $p_{e_1}$ until $p_{e_2}$.

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 $p_{e_1}$ and $p_{e_2}$, but we also consider additional information in that span.
This is a rough way of filtering for important information, but these spans can also grow quite long. In such cases, we have the same problems as if we would consume the whole sentence as input. Additionally, sometimes relevant information is found outside that span, but then it can not be considered by the system. In this work, we took 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 at $p_{head,e_1}$ and $p_{head,e_2}$.

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 and simpler model with this idea.

There are first attempts at using the Transformer architecture [Vaswani et al., 2017] and encoding a sentence using self attention [Bilan and Roth, 2018], letting go of RNNs and CNNs (and hybrid approaches). However, the Transformer is a sequence to sequence network and to map it to a sequence classification problem, they apply max pooling over the encoded sequence to get a final representation to classify, not explicitly modelling the two entities taking part in a relation (apart from indicating the two entities via relative positions).

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 the two entity heads pay attention to the context. 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\(^1\) and set the trainability of the ELMo model to false. We chose to precompute the embeddings because training the transformer architecture successfully is cumbersome and requires large batch sizes\(^2\) for which computing the embeddings for every batch on the fly would become a bottleneck.

---

1. https://tfhub.dev/google/elmo/2
Afterwards, we retrieve the context of a relation, that is the word embeddings from position \( p_{\text{head},e1} + 1 \) until \( p_{\text{head},e2} - 1 \), and stack them in a context matrix. We also retrieve the two entity heads at position \( p_{\text{head},e1} \) and \( p_{\text{head},e2} \) 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]:

\[
\begin{align*}
PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{\text{model}}}) \\
PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{\text{model}}})
\end{align*}
\]  

(1)

where \( pos \) is the position, \( i \) the dimension and \( d_{\text{model}} \) 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 that head position 2, and so on:

\[
pos_i = pos_i - p_{\text{head},e1} + 1
\]  

(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
Learning Relation Classification using Attention only

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:

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

where \(\text{head}_i = \text{Attention}(\text{ReLU}(QW_i^Q), \text{ReLU}(KW_i^K), \text{ReLU}(VW_i^V))\) \hspace{1cm} (3)

with Attention being defined as:

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

Where we learn different parameter matrices \(W_i^Q \in \mathcal{R}^{d_{model} \times d_k}\), \(W_i^K \in \mathcal{R}^{d_{model} \times d_k}\), \(W_i^V \in \mathcal{R}^{d_{model} \times d_v}\) and \(W_i^O \in \mathcal{R}^{hd_v \times d_{model}}\) to map the embeddings into an appropriate space followed by a ReLU activation. We set \(d_{model} = 1024\) which is the dimensionality of the ELMo embeddings and \(d_k = d_v = 1024/4 = 256\), that is the dimensionality of the embeddings divided by the number of heads (which we set to 4).

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. 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.

\[
x = \text{layernorm}(x + \text{subLayer}(x)) \hspace{1cm} (5)
\]
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 ($2 \times d_{\text{model}}$) into a vector of size $2 \times \text{model}_\text{dim}$ and feed this vector into a feedforward layer taken from the Transformer architecture.

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

(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.

### 3.3 Regularization

We add five types of regularization for the model. After we added the positional embeddings to the input, we add a dropout. Secondly, we dropout values after every layer before feeding the layer’s output into the Add & Norm component. For both, we use a dropout rate of 0.5. Thirdly, we add l2-regularization in the classification layer with a $\lambda = 0.05$ and a dropout with rate 0.1 to the input of the classification layer. Since ELMo embeddings have a dimensionality of 1024, the final vector we classify on has dimensionality 2048 (the two entities with added information concatenated). Because we only have a finite amount of training examples, we impose heavy regularization on that layer. At last, we employ label smoothing with $\epsilon = 0.1$.

### 3.4 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) [Vaswani et al., 2017]. 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 found that while the full transformer architecture yields similar, slightly worse results, the resulting attention scores are no longer interpretable.

Adding self attention of the entities does not conceptually make much sense since the only information to be gained is to look at the other entity. This results in more uniform attention scores over both entities, but also results in slightly worse results and adds lots of trainable parameters. 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 four attention heads.

### 3.5 Interpretability of the Model

Compared to other approaches, we incorporate a very scarce set of features: Only the word embeddings produced by ELMo. We do add parameter-free positional embeddings on top,
but their impact is well understood: they are sine and cosine frequencies helping the model to have a sense of position. Apart from these two, we do not employ any additional features.

Secondly, what we actually do is take the two entities and add information on top of them. However, compared to other neural network methods, we can document where the added information comes from. This can be done by inspecting the attention scores. 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 them 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 most of their attention over the verb "teaches" and the preposition "with" (Figure 4). This corresponds to the shortest dependency path between the two entities. Complementary to this, a bit of attention gets distributed over "lesson" and the other words.

Figure 4: Example of attention scores in SemEval-2010 testset
If we would want to coin a relational phrase or describe the above relation in first-order
logic, we would probably 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, and what they learn to pay attention to is very close to the predicate or the
relational phrase. We find it exciting that these patterns emerge from learning supervised
relation classification.

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 "23") from the span between them ("a dissident who died February") 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 two tokens paid attention to the most ("died" and "February") describe the relation
very accurately. We list more examples of attention scores in the appendix C.

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

In comparison to RNNs, especially their gated, deep and bidirectional version, we do not
have long, incomprehensible information flow. Shorter paths also help in dealing with long
distance dependencies and in comprehending where information comes from. Compared
to CNNs, the queries have access to global information to help determine the relation and
again, we can highlight from where we add information.
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 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 test set. All training details can be found in appendix A.

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 1).

<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 1: 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 explicitly mention our evaluation procedure, that is we trained the model 5 times with different seeds and report the median of the 5 runs. 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. The 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 memory states of the sequence. We again regularize the model by adding dropout at the input and before the classification layer and l2 regularization on the classification layer. We also report results for our proposed architecture with GloVe embeddings and results for the full transformer architecture with GloVe and ELMo embeddings (Table 2).

3. 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>results reported from paper</td>
</tr>
<tr>
<td>LSTM + GloVe embeddings</td>
<td>our implementation</td>
</tr>
<tr>
<td>LSTM + ELMo embeddings</td>
<td>our implementation</td>
</tr>
<tr>
<td>standard Transformer + GloVe embeddings</td>
<td>our implementation</td>
</tr>
<tr>
<td>standard Transformer + ELMo embeddings</td>
<td>our implementation</td>
</tr>
<tr>
<td>truncated Transformer + GloVe embeddings</td>
<td>our implementation</td>
</tr>
<tr>
<td>truncated Transformer + ELMo embeddings</td>
<td>our implementation</td>
</tr>
</tbody>
</table>

Table 2: Impact of ELMo embeddings

We think that a bidirectional LSTM on top of ELMo embeddings (which are produced by bidirectional LSTMs too) can not extract much more information which it could not have learned by training it on GloVe embeddings in the first place. Therefore we only have a small performance gain by replacing the 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. We suspect that the full transformer module combined with ELMo embeddings has too many parameters and highly overfits the training set despite heavy regularization. We also think that in the GloVe setting, the full Transformer has to build its own contextualized representations over the input by performing self attention, while in the ELMo setting, we already have that information implicitly in the embeddings.

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 such long entities, we decided to search for the heads of the entities and use their embedding to query. Whatever follows after the head is added to the context and gets queried by the entity head.
in the attention layer. The tokens preceding the 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 the model and also update the number of outputs from the attention layer and the feedforward layer to still be able to perform the residual connection. We only do this for the entities and not for the tokens in between the two entity heads.

To counter the class imbalance (the dataset is dominated by the label "no relation"), we balance the class weights for all classes.

\[
\text{class_weights}_i = \frac{1}{\text{class_counts}_i/\text{num_examples}} \quad (7)
\]

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. For example, 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 quite 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 3, we compare the result of the single model with the median F1 score on the development set 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><strong>69.9</strong></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><strong>68.6</strong></td>
<td>66.5</td>
</tr>
<tr>
<td>truncated Transformer + ELMo Embeddings</td>
<td>our approach, 2018</td>
<td>67.3</td>
<td>67.0</td>
<td><strong>67.1</strong></td>
</tr>
</tbody>
</table>

Table 3: 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. However, we get a very balanced metric between the two and think that this is partly due to the way we set the class weights. 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 4.

Again, we report very competitive results here. C-GCN + PA-LSTM is the combined output of two different neural networks which complement each other, therefore we find it exciting to get very close to their results.
Table 4: Ensemble performance on the TACRED testset

<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 Transformer</td>
<td>our approach, 2018</td>
<td>69.3</td>
<td>66.9</td>
<td>68.1</td>
</tr>
</tbody>
</table>

4.4 Discussion
We show on two established datasets that our architecture yields very competitive results and we achieve the best reported single model result on the TACRED dataset. In comparison to earlier work, we add three major contributions. Firstly, we completely move away from tedious feature engineering and additional external resources (apart from the NE types and the position of the heads of entities in the TACRED corpus which are both included in the distribution of the dataset) and our only preprocessing step consists in computing the ELMo embeddings. Secondly, we make a step forward in the direction of more interpretable models where we can track how information flows and what information we actually use to predict the relation between two entities. Thirdly, we propose a slim and elegant architecture by separating the two entities and the additional context in a sentence and using the entities to query this additional context. This also saves computational power and time, with the bottleneck lying in precomputing the embeddings. But afterwards, using one single GPU (GTX 1080, 11GB), we can train one epoch on the TACRED training set in about one minute.

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 from 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 firstly lexicalize the relation, and secondly mostly lie on the Shortest Dependency Path between the two entities. This is evidence for the Shortest Path Hypothesis and vica versa, assuming the hypothesis holds true, is evidence that our network learned what it was supposed to learn.
References


Appendix A. Training Details

We trained all our models with the adaptive learning rate schedule from [Vaswani et al., 2017].

\[ lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \]  

(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>250</td>
<td>500</td>
</tr>
<tr>
<td>num_steps</td>
<td>1000</td>
<td>2000</td>
</tr>
<tr>
<td>batch_size</td>
<td>2000</td>
<td>4000</td>
</tr>
<tr>
<td>num_heads</td>
<td>4</td>
<td>4</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>2048</td>
<td>2112</td>
</tr>
<tr>
<td>embeddings_Q for each head</td>
<td>256</td>
<td>264</td>
</tr>
<tr>
<td>embeddings_K for each head</td>
<td>256</td>
<td>264</td>
</tr>
<tr>
<td>embeddings_V for each head</td>
<td>256</td>
<td>264</td>
</tr>
<tr>
<td>dropout input and before layer norm</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>dropout before layer norm</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>dropout classification layer</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>l2 lambda</td>
<td>0.05</td>
<td>0.05</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>normalize_batch_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>0.98</td>
</tr>
<tr>
<td>adam_epsilon</td>
<td>1e-09</td>
<td>1e-09</td>
</tr>
</tbody>
</table>

Table 5: Training details

Appendix B. All Results for both Datasets

In table 6, 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 7). We report the testset value based on the median of the development set (run 2).
Table 6: 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>86.31</td>
</tr>
<tr>
<td>2</td>
<td><strong>86.69</strong></td>
</tr>
<tr>
<td>3</td>
<td>86.74</td>
</tr>
<tr>
<td>4</td>
<td>86.60</td>
</tr>
<tr>
<td>5</td>
<td>86.98</td>
</tr>
</tbody>
</table>

Table 7: 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>68.03</td>
<td>67.01</td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><strong>67.79</strong></td>
<td><strong>67.14</strong></td>
</tr>
<tr>
<td>3</td>
<td>67.42</td>
<td>67.50</td>
</tr>
<tr>
<td>4</td>
<td>67.31</td>
<td>67.53</td>
</tr>
<tr>
<td>5</td>
<td>67.84</td>
<td>66.96</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 an indicator whether our prediction is correct or not. The second line contains the attention scores for entity 1 (yellow), where the attention scores for the four different heads precede each token. The more attention is paid to a token by a head, the more colored it is. 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).
true label: "Message-Topic(e1,e2)"; predicted label: "Message-Topic(e1,e2)"; correct prediction

```
 audits 0.6 0.4 0.6 0.4 were 0.4 0.6 0.4 0.6 about waste
 audits 0.7 0.6 0.4 0.3 were 0.3 0.4 0.6 0.7 about waste
```

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

```
 company 0.8 0.6 0.4 1.0 fabricates 0.2 0.4 0.6 0.0 plastic chairs
 company 0.2 0.1 0.6 1.0 fabricates 0.3 0.9 0.4 0.0 plastic chairs
```

true label: "Instrument-Agency(e2,e1)"; predicted label: "Instrument-Agency(e2,e1)"; correct prediction

```
 master 0.7 0.1 0.1 0.0 teaches 0.1 0.3 0.4 0.0 the 0.1 0.1 0.1 0.0 lesson 0.0 0.4 0.2 1.0 with 0.1 0.1 0.2 0.0 a stick
 master 0.6 0.1 0.1 0.0 teaches 0.0 0.3 0.1 0.0 the 0.3 0.3 0.1 0.0 lesson 0.0 0.3 0.5 1.0 with 0.1 0.1 0.3 0.0 a stick
```

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

```
 body 0.3 0.4 1.0 1.0 into 0.3 0.4 0.0 0.0 a 0.5 0.2 0.0 0.0 local reservoir
 body 0.1 0.1 0.6 1.0 into 0.6 0.8 0.3 0.0 a 0.3 0.1 0.1 0.0 local reservoir
```

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

true label: "Member-Collection(e2,e1)"; predicted label: "Member-Collection(e2,e1)"; correct prediction

```
 camp 0.2 0.4 0.8 1.0 of 0.8 0.6 0.2 0.0 Heine champions
 camp 0.1 0.3 0.9 1.0 of 0.0 0.2 0.1 0.0 Heine champions
```

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

```
 cartridge 0.2 0.1 0.2 0.0 was 0.2 0.1 0.2 0.0 marked 0.1 0.1 0.1 0.0 as 0.2 0.2 0.3 0.0 empty 0.1 0.2 0.0 0.0 0.1 0.2 0.1 1.0 with ink
 cartridge 0.1 0.0 0.1 0.0 was 0.5 0.1 0.2 0.0 marked 0.0 0.1 0.1 0.0 as 0.2 0.2 0.3 0.0 empty 0.0 0.1 0.0 0.0 0.1 0.2 0.0 0.0 0.0 0.4 0.0 with ink
```

true label: "Member-Collection(e2,e1)"; predicted label: "Member-Collection(e2,e1)"; correct prediction

```
 ped 0.5 0.8 0.9 1.0 of 0.4 0.5 0.1 0.0 sperm whales
 ped 0.5 0.8 0.9 1.0 of 0.5 0.2 0.1 0.0 sperm whales
```

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

```
 Roundworms 0.1 0.0 0.3 0.0 or 0.2 0.0 0.1 0.0 ascards 0.1 0.0 0.0 0.0 are 0.3 0.0 0.0 0.0 caused 0.0 0.0 0.0 0.1 1.0 by 0.0 0.0 0.4 0.0 an 0.3 0.0 0.0 0.0 intestinal parasite
 Roundworms 0.0 0.0 0.0 0.0 or 0.0 0.0 0.0 0.0 ascards 0.0 0.0 0.0 0.0 are 0.0 0.0 0.0 0.0 caused 0.0 0.0 0.0 0.0 0.0 by 0.0 0.0 0.3 0.0 an 0.0 0.0 0.0 0.0 intestinal parasite
```

Figure 9: More examples of attention scores in SemEval-2010 testset
true label: “Instrument-Agent(e, a1)”; predicted label: “Other”; wrong prediction

organizations: 0.0 0.1 0.0 1.0 effectively 0.3 0.5 0.0 0.3 manage 0.0 0.4 0.1 0.0 their resources
organizations: 0.4 0.2 0.0 0.0 effectively 0.5 0.7 0.0 0.0 manage 0.1 0.1 0.1 0.1 their resources

ture label: “Component-Whole(e, a2)”; predicted label: “Component-Whole(e, a2)”; correct prediction

Middle: 0.8 0.6 0.1 0.0 of 0.4 0.4 0.2 0.0 aquatic animals
Middle: 0.8 0.1 0.1 0.0 of 0.0 0.0 0.0 0.0 aquatic animals

ture label: “Member-Collection(e, a1)”; predicted label: “Member-Collection(e, a1)”; correct prediction

coffee 0.5 0.6 0.9 1.0 of 0.5 0.6 0.1 0.0 fourteen assed

coffee 0.3 0.2 0.1 0.0 of 0.7 0.5 0.0 0.0 fourteen assed

ture label: “Entity-Origin(e, a2)”; predicted label: “Entity-Origin(e, a2)”; correct prediction

colalled: 0.1 0.1 0.1 1.0 of interests

colalled: 0.1 0.1 0.1 1.0 of interests

Figure 10: More examples of attention scores in SemEval-2010 testset

true label: “Entity-Origin(e, a2)”; predicted label: “Entity-Origin(e, a2)”; correct prediction

Character: 0.1 0.3 0.1 0.1 from 0.0 0.1 0.1 0.0 Arabic Literature

Character: 0.1 0.3 0.1 0.1 from 0.0 0.1 0.1 0.0 Arabic Literature

ture label: “Entity-Origin(e, a2)”; predicted label: “Entity-Origin(e, a2)”; correct prediction

bourbon 0.2 0.0 0.2 0.0 was 0.0 0.0 0.1 0.0 removed 0.0 0.2 0.1 1.0 from 0.0 0.1 0.1 0.0 original barrel

bourbon 0.0 0.0 0.1 1.0 was 0.0 0.0 0.1 1.0 removed 0.0 0.2 0.1 1.0 from 0.0 0.0 0.1 1.0 original barrel

true label: “Component-Whole(e, a2)”; predicted label: “Component-Whole(e, a2)”; correct prediction

heads 0.6 0.5 0.4 0.3 of 0.4 0.5 0.6 0.2 the ferrrets

heads 0.4 0.2 0.3 1.0 of 0.0 0.0 0.7 0.1 the ferrrets

true label: “Cause-Effect(e, a2)”; predicted label: “Cause-Effect(e, a2)”; correct prediction

Lessons: 0.0 0.1 0.1 0.0 in 0.2 0.0 0.3 0.0 the 0.1 0.0 0.1 0.0 internal 0.0 0.0 0.0 0.0 capsule 0.0 0.0 0.1 1.0 caused 0.0 0.0 0.1 0.0 proportional 0.7 0.0 0.0 0.0 leg weakness

Lessons: 0.0 0.0 0.0 0.0 in 0.0 0.0 0.0 0.0 the 0.1 0.0 0.0 0.0 internal 0.1 0.0 0.0 0.0 capsule 0.4 0.0 0.1 0.0 caused 0.0 0.0 0.0 0.0 proportional 0.3 0.0 0.1 0.0 leg weakness

true label: “Instrument-Agent(e, a2)”; predicted label: “Instrument-Agent(e, a2)”; correct prediction

researcher 0.7 0.2 0.1 0.5 approached 0.2 0.1 0.4 0.0 the 0.1 0.1 0.0 0.0 data 0.0 0.0 0.1 0.5 with 0.0 0.5 0.0 0.0 predetermined Categories

researcher 0.6 0.1 0.3 0.0 approached 0.1 0.2 0.0 0.0 the 0.2 0.3 0.0 0.0 data 0.2 0.3 0.0 0.0 with 0.1 0.1 0.3 0.0 predetermined Categories

true label: “Content-Container(e, a2)”; predicted label: “Content-Container(e, a2)”; correct prediction

money 0.1 0.2 0.1 0.0 was 0.0 0.3 0.0 0.0 hidden 0.1 0.2 0.1 1.0 in 0.0 0.2 0.0 0.0 his bag

money 0.0 0.2 0.0 0.0 was 0.1 0.0 0.1 1.0 hidden 0.0 0.2 0.5 1.0 in 0.0 0.2 0.0 0.0 his bag

true label: “Cause-Effect(e, a2)”; predicted label: “Cause-Effect(e, a2)”; correct prediction

suffering 0.0 0.0 0.0 0.0 and 0.3 0.0 0.0 0.0 devastation 0.6 0.0 0.0 0.0 caused 0.0 0.0 0.1 1.0 by 0.0 0.0 0.1 0.0 the bomb

suffering 0.0 0.0 0.0 0.0 and 0.0 0.0 0.0 0.0 devastation 0.9 0.0 0.0 0.0 caused 0.0 0.0 0.5 1.0 by 0.0 0.0 0.5 0.0 the bomb

Figure 11: 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 12-15). 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").

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

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

Figure 14: More examples of attention scores in TACRED development set
true label: "no_relation"; predicted label: "no_relation"; correct prediction
head entry 1: Group looks at:

head entry 2: Industries looks at:

true label: "person-origin"; predicted label: "person-origin"; correct prediction
head entry 1: Rashard looks at:

head entry 2: Afghan looks at:

true label: "per date_of_death"; predicted label: "per date_of_death"; correct prediction
head entry 1: Anderson looks at:

head entry 2: Thursday looks at:

true label: "no_relation"; predicted label: "no_relation"; correct prediction
head entry 1: Monday looks at:

head entry 2: Center looks at:

Figure 15: More examples of attention scores in TACRED development set