Human Language Engineering

Final Project Report

Academic year: 2022/2023

Author: Joaquim Marset Alsina

Contents

1 Introduction

The present report contains the final project from the subject *Human Language Engineering*, which is about comparing different methods to perform relation extraction (RE) on the *SemEval-2010 Task 8* [4] dataset.

Relation extraction extracts and classifies relations between pairs of entities in a sentence. It is an important sub-task of other NLP tasks like information extraction (IE) or question answering (QA). Typically, the sentences are already annotated with the entities to classify, so the methods are not end-to-end and consist of a multi-class classification problem.

For example, we can have the sentence "The <e1>pollution</e1> was caused by the <e2>shipwreck</e2>", which would come annotated with the entities "pollution" and "shipwreck", and the class "Cause-Effect(e2, e1)". That is, the first entity is an effect caused by the second. To predict the relationship between this pair of entities, we only need to focus on the verb "was caused", as it gives away the relation class. We can also see how we have tags enclosing the entities, which is a way to indicate the entities' position that some models use.

2 Project Goals

The idea of this project was to compare multiple methods to tackle the problem of RE. BERT-based methods outperform all previous approaches in most NLP tasks, and RE is not exempt. Thus, simply trying to solve the classification problem presented by SemEval was kind of out of the question, as with a simple BERT, we can achieve very good results.

Another motivation for comparing multiple methods is that it allows us to learn and implement how to solve a real NLP challenge using different approaches based on deep learning. In the previous subjects of the same line in the master, we worked with real challenges, but we only applied classical machine learning.

When we talk about implementing existing methods, it means reaching similar results to those reported by the authors, and then trying to improve their results using different hyperparameters, pre-trained embeddings, and features. With features, we refer to either adding handcrafted features as extra input or changing some components in the architecture to generate some hidden features that could benefit the model.

Then, once we have all the methods implemented, the idea would be to perform the comparison with the selected dataset. This comparison should consider the metrics reported by the model and the errors each method commits on the different classes. For example, looking for patterns in the wrongly classified sentences.

3 State of the Art

To tackle the problem of RE, we usually find three types of methods: supervised methods, methods using distant supervision, and unsupervised methods. Supervised methods are the most common but require having enough annotated data for the model to train. Distant supervision is the group of methods augmenting the dataset with data extracted from a knowledge base. However, as we saw in class, the extracted data is very noisy because of the label assumptions and hinders training the models. Finally, unsupervised methods do not need annotated data, so the model needs to unravel the possible classes and learn how to recognize them.

As supervised methods are more common, we have decided to focus on them. We can classify them as follows:

- Methods based on classic machine learning
- Methods based on neural networks
- Methods based on computing the shortest dependency path
- Methods based on transformers like BERT [2]

The first type relies on classic machine learning approaches like a support vector machine (SVM). This type depends completely on handcrafted features like Wordnet synset, part of speech (PoS) tagging, lemmas, or features from a dependency tree. To compute them, we rely on NLP tools, which can have errors that we propagate through the model. Also, computing some of these features is very costly, for example, the dependency tree. However, these methods are fast to train and can give decent results. The winner of the SemEval competition from the dataset we use, was based on an SVM using 45 handcrafted features [10].

The second type uses neural networks to extract information from words automatically. Neural networks outperform classic machine learning approaches and do not rely on handcrafted features. However, using these handcrafted features can further boost performance by taking the extra cost to compute them. To solve RE, we usually use networks based on a CNN [17, 11] or an RNN [18, 5], and we can further combine them with attention mechanisms [11, 18, 5]. We use attention mechanisms to generate a sentence representation (i.e. a vector summarizing the sentence) containing the relevant information to predict the relation class. When only considering the words as the single input, we usually need to specify the entities' position to the model. We want to predict a relationship between the entities, so the entities should give important information to predict the class. Also, words closer to the entities should be more important than those far away. Thus, we can either use tags to enclose the entities (like we saw in section 1) or use some positional information like relative positions.

The third type is based on dependency parsers and computing the shortest dependency path (SDP) between the entities and their lowest common subsumer (LCS) [16, 15]. In Figure 1, we show an example of what the SDPs of the sentence "A trillion gallons of water have been poured into an empty region of outer space" would be.

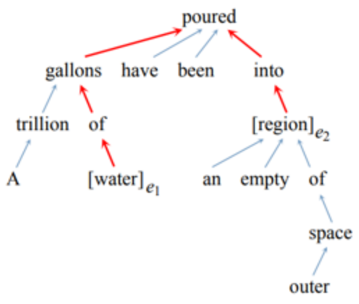


Figure 1: Example of shortest dependency paths (taken from [16]

The SDP-based method only considers the words in each sub-path and discards the others. It follows the idea that words close and in between the entities should be the most important to classify the relation. Thus, it only considers the relevant words in the sentence and removes the irrelevant ones. Also, thanks to only processing the sub-paths, we do not need any extra positional information, as the model can easily infer that the entities are always at the extremes. SDP-based methods mostly use neural networks (i.e. CNNs [15] and RNNs [16]), processing each sub-path independently and then merging the features of both sub-paths to predict the class. The main problem is the total dependency on computing the dependency tree, which is exponential on the sentence length and can have errors later propagated through the model. In terms of performance, it can achieve better results than methods from the previous type.

The last type leverages transformers, and in particular, pre-trained language models like BERT [2]. As in other NLP tasks, pre-trained language models have made their way, outperforming older approaches. In RE, this is also the case, and one can easily check the scores on the SemEval dataset to see that the top-10 methods use BERT [14, 1, 6]. We can use BERT to generate contextual embeddings of the different words instead of the older distributional ones. However, we can also use the embedding of the [CLS] token to perform classification problems like this one. And even combine it with the entities embedding to add more information that could help solve this problem we are facing. In BERT-based methods, we also need to specify the entities' position to the model, either with some enclosing tags or some extra positional features.

Among the transformer-based methods, we want to mention [6], which is a method that can predict multiple relations in a sentence, even overlapping relations between the same pair of entities. It relies on computing all the possible relations between all the possible words in a sentence and then filtering out those pairs of words that are not entities. Thus, the method can become costly if we have a lot of relation types and very long sentences. We also mention this method because our dataset only contains annotations for one pair of entities. Therefore, a model trained on this dataset can only predict one relation even if the sentence has more than one annotated. This dataset almost has no sentences with multiple or overlapping relations, but there are others like NYT [9] (built using distant supervision) and Web-NLG [3], which have them.

4 Methods

We have implemented four methods that we explain below. Three of them leverage neural networks and the other BERT. As we have explained, we have started by reproducing the authors' results and then improving them. We have used Pytorch to implement all the neural network-based methods and HuggingFace to implement the BERT-based one.

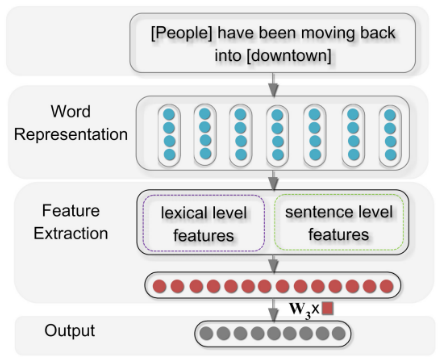
4.1 CNN

[17] uses a convolutional network to merge the information from neighbour words, similar to what a recurrent layer does when sequentially encoding a sentence. The convolution operation considers the current word and some local context with a size depending on the kernel size.

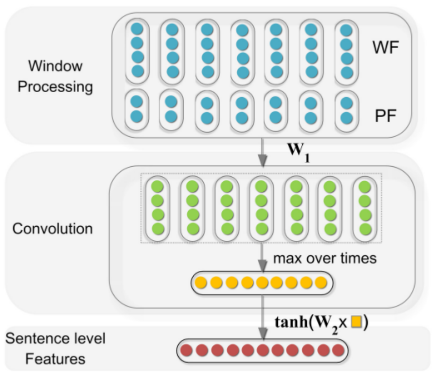
However, we cannot integrate the information of different contexts with only the convolution. For this reason, we have a max pooling layer across the sequence length plus a non-linear layer to better merge the word information in a vector summarizing the sentence. The input to this sub-model consists of the word embeddings and relative position embeddings to indicate the relative position to each entity (concatenating both embeddings).

Besides this sentence representation, which the authors call sentence-level features, the model also computes some features only from the entities. In particular, the entity word embedding, the embedding of the two adjacent words, and some WordNet hypernyms when the entity is a noun.

Finally, they concatenate both types of features and feed it to a softmax layer to predict the class. In Figure 2, we show the complete model architecture.



(a) General architecture



(b) Sentence-level features architecture

Figure 2: CNN model architecture (taken from [17])

As stated in the paper, this method achieved a macro-average F1 score (the official dataset score) of 82.7%. They leveraged the pre-trained embeddings from [12], which generates vectors of fifty elements. In terms of hyperparameters, they used an Adam optimizer with a default learning rate, a convolution with 100 filters and a kernel size of 3, and 200 neurons in the non-linear layer.

In our implementation, we did not consider the WordNet hypernyms. We wanted to start with few handcrafted features as possible, and once we could reach a similar performance to the authors', adding them to boost performance further. We also tried different and more modern pre-trained embeddings. In particular, we used GloVe [8] with multiple dimensions and Google's word2vec [7]. We also tried changing the kernel size and adding more convolutions with different kernel sizes to generate different-sized contexts.

Our best performance was 80.06%, quite far from the reported results. We used the same basic parameters as the authors. That is, parameters like learning rate, optimizer, dense units, and filters, as we considered it more important to change the others. The best pre-trained embedding was GloVe with dimension 100. Increasing it more than this point became counter-productive, as we gave too much weight to the lexical features compared to the sentence representation. Regarding the changes in the convolutions and kernel sizes, we did not get any improvement by changing the kernel size or adding multiple convolutions and concatenating their features. We believe this is because we lose information about certain words when using larger contexts.

We were quite surprised with this method because we saw it as conceptually simple, but getting good results became very difficult. We have indeed ignored one feature, but we believe the problem is also because CNNs are not as good at capturing relations and processing sentences as RNNs. Given that we spent a lot of time to reach that performance and wanted to implement more methods, we decided to let it at that performance and move on.

4.2 Attention Bi-LSTM

[18] is another method based on neural networks, and in particular, LSTMs. As the name implies, its main components are a bidirectional LSTM and an attention mechanism. With the bidirectional LSTM, we can sequentially encode the sentence starting from the beginning and end, taking into account past and future words. The attention mechanism the authors proposed follows the same idea of wanting to generate a context vector weighting each word according to its importance. However, unlike the most common uses of the attention mechanism in machine translation, in which we generate a context vector for each word, here we get a single vector that summarizes the whole sentence. The idea is that this sentence representation contains the information from the relevant words to predict the relationship between the pair of entities.

We present the architecture in Figure 3. As we can see, the only input this model needs is the sentence. We feed it to an embedding layer to generate word embeddings. Then it follows the Bi-LSTM and the attention mechanism. Finally, we feed the mentioned sentence representation to a softmax layer to predict the class.

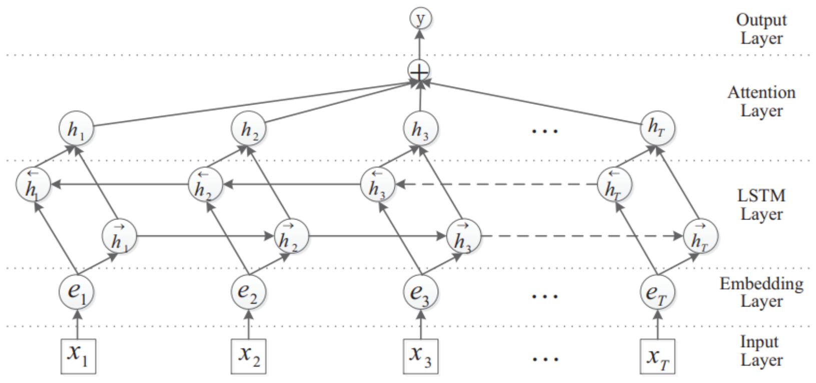

y
Output Layer
Attention Layer
h_1 h_2 h_3 ... h_T
LSTM Layer
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
Embedding Layer
e_1 e_2 e_3 ... e_T
Input Layer
x_1 x_2 x_3 ... x_T

Figure 3: Attention Bi-LSTM architecture (taken from [18])

However, we do not have any positional information to help the model know the entities' location. And as we have explained, we need this information to predict the relation class between them. For this reason, the authors proposed using the enclosing tags we mentioned in section 1 to locate the entities. We do not have relative positions, so the model needs to infer which words are closer. However, the recurrent behaviour of LSTMs should help in that aspect.

Regarding the embedding layer, the authors used GloVe embeddings with dimension 100. They also tried the same as the ones used in [17], but they got better results with GloVe, which is in line with what we saw in that method. Regarding other common hyperparameters, the authors used an AdaDelta optimizer with a learning rate of 1 and a batch size of 10. They also used dropout to regularize the output of the embedding layer, Bi-LSTM, and attention mechanism. They also used an L2 loss to improve generalization.

With this setting, the authors achieved an 84.0% macro-average F1 score. If we compare it with the performance of the CNN reported by the authors, this one achieved better performance without the need for any handcrafted feature.

As this method does not have any handcrafted features, we started by implementing it the same way the authors did. We tried different sizes of GloVe and word2vec, but we did not get any significant change. In terms of other hyperparameters, we tried changing the LSTM units, the optimizer, the learning rate, and the batch size. We achieved the best performance by using the same parameters the authors reported, given that we implemented it the same way. With the same setting the authors used, we got an 83.27% macro-average F1 score.

We wanted to try some handcrafted features, but we ended up putting them aside after we could reproduce all the methods. In the end, we ran out of time and did not try it. And this is something that also happened with the following method. However, what we tried, as it was very simple to implement, was changing the distributional embeddings for contextual embeddings. In this case, we used the uncased BERT base model [1], which we also use in the following methods. By only changing the pre-trained embedding matrix, we could improve our performance up to 87.42%.

4.3 Entity-Aware Attention

[5] is based on the previous one, but adding some additional components and changing the attention mechanism. We present the architecture in Figure 4.

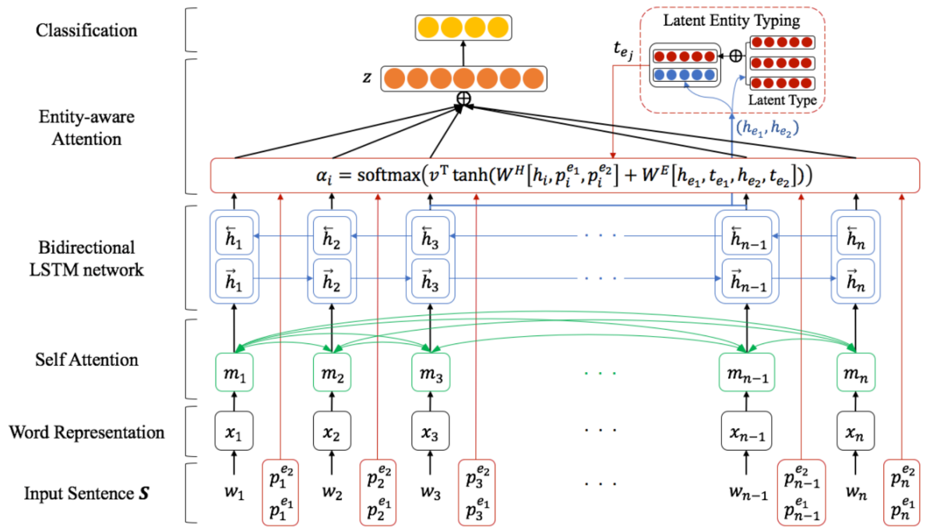


Figure 4: Entity-Aware Attention architecture (taken from [5])

The input consists of both the sentence and relative positions. As in the CNN method, they compute the relative position of each word to each entity as a way to know which words are closer to the entities. Like the previous method, the entities are enclosed with tags to indicate their position.

They generate word embeddings using a pre-trained GloVe with dimension 300 and also generate relative position embeddings by randomly initializing the embedding matrix. In this case, they used vectors of dimension 50. After generating the word embeddings, they go through a multi-head self-attention [13] to improve the embeddings taking into account the context words in the sentence.

Then it follows the same Bi-LSTM we saw in the previous method and an attention mechanism, which has the same purpose of generating a sentence representation focusing on the important words to predict the class. However, the authors decided to modify the attention mechanism, giving it the name entity-aware attention.

The authors say that previous methods did not fully exploit the entities' information. And entities should give important hints to classify the relation class, as the relation happens between them. For this reason,

[1]`https://huggingface.co/bert-base-uncased`

when computing the attention weights, we always consider the entities' hidden states and what they call latent types. These latent types are latent variables that represent entity types. Thus, instead of using the typical ones (e.g. PER, LOC, ORG), they decided to learn particular types that could better adapt to the dataset. Then, to compute the type of an entity, they compute it as a weighted sum (similar to soft clustering).

With these three new components, as we can see in the ablation they performed in the paper, the authors improved the performance of the previous method, which served as a baseline. In particular, the biggest improvement came from entity-aware attention, as the one from the relative positions and the self-attention were quite small.

Regarding the common hyperparameters, they used very similar ones to the previous method, only changing the size of the word embeddings. Regarding the hyperparameters of the new components, they decided to use four attention heads and three latent types. With this setting, they could achieve a macro-average F1 score of 85.2%.

To implement this method, we started again by implementing it the same way the authors did, without any modification in the architecture, but by trying different hyperparameters. We saw that we could get very similar results by using a smaller word embedding dimension, so we set it to 100 without losing a lot of performance but gaining some training time. Again, we tried changing the optimizer, learning rate, batch size, and network size. However, we did not improve from the author's setting. Also, we got the same best results when changing the number of heads and latent types. With this setting, we achieved 84.61%.

Again, we did not have time to test other input features. However, we tried changing the distributional word embeddings for contextual, using the same BERT as the previous method. We also decided to remove the self-attention mechanism. It was redundant and hurt performance in the few epochs we trained the model. With this new setting, we achieved an 88.54% macro-average F1 score, again improving almost a 4%.

The good point of this method is the interpretability of the different components. We not only have the attention weights to know which words are relevant to classify the relation, but we also have the self-attention weights and the weights when computing the type of an entity. Therefore, we can know which words another one focuses on when improving its embeddings. And we can also know which of the learned types is more important when computing the entity type.

4.4 R-BERT

[14] leverages BERT to predict the relation class as a simple classification downstream task. It uses the uncased base BERT, as the previous methods. We present the architecture in Figure 5:

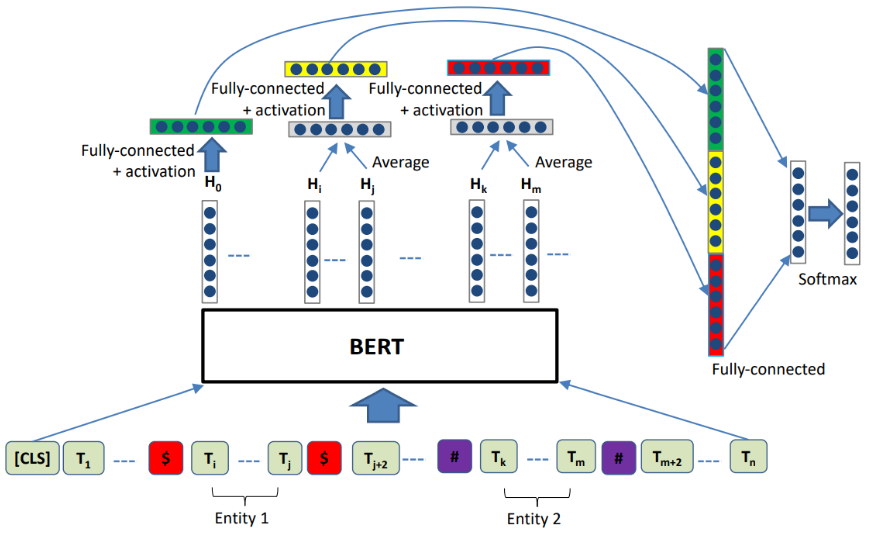


Figure 5: R-BERT architecture (taken from [14])

It receives the sentence as input but with some modifications. First, we need to add the specific BERT tokens (i.e. [CLS] and [SEP] tokens). In this case, we do not need to separate two sentences, so we do not need the [SEP] token.

Second, as Attention Bi-LSTM did, it adds specific tags to enclose the entities and indicate their position. Without them, the model cannot infer the entities' location and use it to incorporate the entities' information in the [CLS] embedding (i.e. the sentence representation). R-BERT uses $ to enclose the first entity and # for the second one.

With only the embedding of the [CLS] token, we could feed a classifier to predict the relation. However, the authors found that the entities' embedding can provide richer information to predict the class. For this reason, R-BERT also uses those entities' embeddings to feed the softmax classifier. Each embedding is first fed through a non-linear layer to improve its features, and then the three embeddings are concatenated into one vector. An entity can be composed of multiple words. In that case, they compute an average of the embeddings.

With this simple model, the authors achieved an 89.25% macro-average F1 score. The current state of the art for the SemEval dataset is 91.9%, also based on a BERT model.

For this method, we only tried to reproduce the results from the authors. Thus, we used the same hyperparameters. And we did not try any modification given that, as we saw in class, modifying BERT to add more input features is not that easy. With the same setting as the authors, we got an 88.74% macro-average F1 score.

5 Dataset

We have evaluated our methods in the SemEval 2010 Task 8 dataset. The dataset contains 10717 sentences, split into training (8000) and test (2717) subsets. Each sentence is annotated with a pair of entities and their relation type. Therefore, it does not consider sentences having multiple or overlapping relations.

The dataset features nine relation types, plus another one ("Other") to indicate some unknown relationship. The known types are:

- Cause-Effect: One entity denotes an object or event that generates an effect denoted by the other entity.
- Component-Whole: One entity represents a component of a larger whole represented by the other entity. In this class, we assume the component has some functionality inside the whole.
- Communication-Topic: One entity denotes some communication act, either spoken or written. The other entity represents the topic of the communication act.
- Content-Container: One entity represents an object physically stored in a delineated space, represented by the other entity.
- Entity-Destination: One entity is moved towards the other. The latter represents a destination.
- Entity-Origin: One entity comes or originates from the other that represents its origin.
- Instrument-Agency: One entity denotes an agent that uses the second entity that represents an instrument.
- Member-Collection: One entity denotes a member of a bigger collection represented by the second entity. In this class, the member does not have any functionality.
- Product-Producer: One entity denotes one product that the other entity, which represents a producer, has created.

Each relation has directionality, meaning we distinguish which entity represents each part of the relation. For example, if we consider the class "Cause-Effect", we have to distinguish between "Cause-Effect(e1, e2)" and "Cause-Effect(e2, e1)". That is, we have to distinguish which entity represents the cause, and which the effect.

Also, the entities are enclosed with special tags that some models, like the Attention Bi-LSTM, use. These tags are "<e1></e1>" for the first sentence, and "<e2></e2>" for the second. They can be used to indicate the entities' position and to identify which entity is which.

We have used the test subset to evaluate hyperparameters and the model's generalization. We could have created a validation subset, but as we do not need to deploy any model and we do not need to evaluate it in more data, we decided to keep all the training data.

The official metric to evaluate this dataset is the macro-average F1 score of all the classes except "Other". To compute this score, we need to consider the directionality and not only the base class. That is, we cannot mark a true positive if we predict correctly "Cause-Effect", but "Cause-Effect(e1, e2)" instead of "Cause-Effect(e2, e1)".

6 Results and Comparison

As we explained in the project goals, what we wanted to do in this project is compare different existing methods to tackle the problem of RE in the SemEval dataset. Therefore, the most important part is evaluating the performance of those models in the test subset. In particular, we have compared the official score but also focused on the errors each method commits in the different classes. Thus, we also report the confusion matrices and show some sentences where some models predict them correctly, but others fail.

6.1 Macro-average F1 scores

We start by comparing the results we obtained in our implementation with those reported by the authors of each method. They only show the official score, so we only use it to perform this first comparison. In Table 1, we show the scores reported by the authors, together with the winner of the SemEval competition and the current state of the art according to [2]. In Table 2, we show the results obtained with our implementations.

SVM (Winner)	CNN	Attention Bi-LSTM	Entity-Aware Attention	R-BERT	SOTA
82.19	82.7	84.0	85.2	89.25	91.9

Table 1: Performance of the different methods reported by their authors

CNN	Attention Bi-LSTM	Entity-Aware Attention	Attention Bi-LSTM BERT	Entity-Aware Attention BERT	R-BERT
80.06	83.27	84.61	87.42	88.54	88.74

Table 2: Best performance obtained with the implemented methods

The results we report in the second table are the best we achieved in terms of hyperparameters, pre-trained weights, and input features to the model. We have already explained these configurations when explaining each method.

We can see how we have reproduced quite decently the different methods except for CNN. In the other three explained methods, we achieved very similar results. Besides the initial four, we have tried to improve the results of the other two that rely on distributional embeddings by swapping to contextual embeddings from BERT. By doing this simple and easy change, we improved the official score by 4% in each method. Also, and this is what we expected, implementing R-BERT was a very simple process that outperformed the others with the default parameters.

6.2 F1 scores of each class

Now that we have demonstrated that we were mostly successful in reproducing and improving the results of the papers, we can move to see how well our implementations perform in the different classes of the SemEval dataset. In Table 3, we present the F1 scores of each class for the different methods.

[2]`https://paperswithcode.com/sota/relation-extraction-on-semeval-2010-task-8`

	CNN	Attention Bi-LSTM	Entity-Aware Attention	Attention Bi-LSTM with BERT	Entity-Aware Attention with BERT	R-BERT
Cause-Effect	89.26	92.02	90.72	93.49	91.94	92.95
Component-Whole	75.12	79.00	81.05	83.65	85.25	85.25
Content-Container	81.91	82.50	85.92	88.37	89.62	89.49
Entity-Destination	85.90	88.47	90.32	92.99	92.72	93.54
Entity-Origin	82.20	86.21	84.29	88.51	89.15	88.76
Instrument-Agency	71.71	74.42	74.36	76.07	82.54	80.94
Member-Collection	80.33	81.78	86.13	87.19	87.03	88.61
Message-Topic	79.32	84.13	87.64	89.71	91.65	91.31
Product-Producer	74.84	80.93	81.09	86.81	86.97	87.79
Other	43.00	53.66	58.02	61.97	62.84	63.28

Table 3: F1 scores of the different classes for the different methods

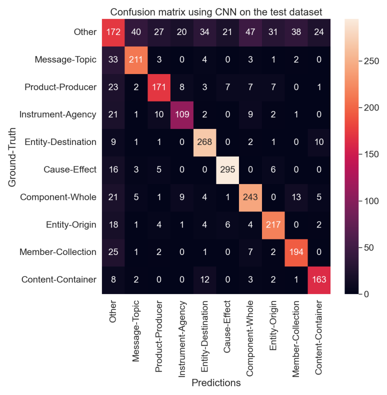
The first we can see is that those methods using contextual word embeddings from BERT reach higher F1 scores than those using distributional ones. We expected it, given the amount of data used to train language models like BERT. It is also interesting to see how all the methods have problems predicting the "Other" class. The official score does not consider it because it does not reflect a concrete relationship between the entities. We see how the different methods improve the score on that class as we move from left to right, but not even R-BERT can achieve a 65% F1 score. The authors of the dataset used it to assign all they could not fit into the other classes. And some of these classes are quite confusing, even for a human.

However, the most interesting thing is that different methods reach the highest performance in the different classes. Even though R-BERT gets the highest macro-average F1 score, the other two methods using contextual word embeddings perform better in some classes. In most cases, the difference is very small and not significant, but there are cases, like "Cause-Effect" and "Instrument-Agency", where the difference is higher.

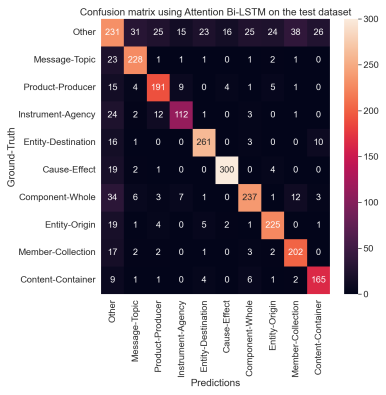
Indeed, certain classes seem to need entity information, like the one from R-BERT and Entity-Aware BERT. Others even need extra information, like the latent types in Entity-Aware BERT. And then, "Cause-Effect" seems to work better when only relying on the whole sentence summary rather than adding entity information. Later, we will look at some sentences from different classes to see why this seems to happen.

6.3 Confusion Matrices

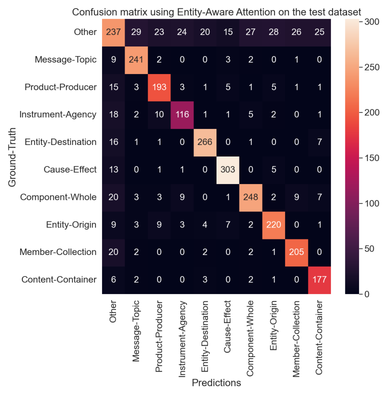
Before looking at examples of sentences, we will generate the confusion matrices to understand the type of errors most methods generate. In Figure 6, we show the confusion matrices of the methods using distributional embeddings. In Figure 7, we show the confusion matrices of those using contextual embeddings.



(a) CNN



(b) Attention Bi-LSTM



(c) Entity-Aware Attention

Figure 6: Confusion matrices of the methods using distributional embeddings

Let's start by looking at those methods using distributional embeddings. We can see how the number of true positives increases from left to right. We expected it considering the F1 scores of each class. If we look at particular classes:

- We can see that in all three methods, the biggest problem is with the "Other" class. We have a lot of false positives and false negatives. We have false negatives because the model finds some words used in a particular class and automatically labels them wrong. False positives happen because some cases, the ground truth is confusing even for a human. We have very few types of relations, they are generic, and even some pairs are conceptually similar.

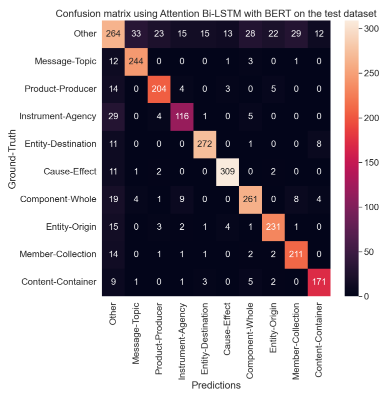
- "Content-Container" is the class we less mistake with "Other". We indeed have more cases of "Other" sentences that we classify as "Content-Container". But we have the smallest number of sentences belonging to some known relationship (i.e. any of the 9) that we predict as "Other". And this is maintained in the three methods.

- A second pair getting confused is when wrongly classifying a "Component-Whole" sentence as "Member-Collection". It is not surprising as these two classes are very similar. The main difference seems to be that the collection members do not have any function. We believe the model wrongly classifies as "Member-Collection" because it cannot infer any functionality from the entity representing "Component". The other way around also happens, but mostly on CNN. Therefore the Bi-LSTM and the attention mechanism help avoid these errors.

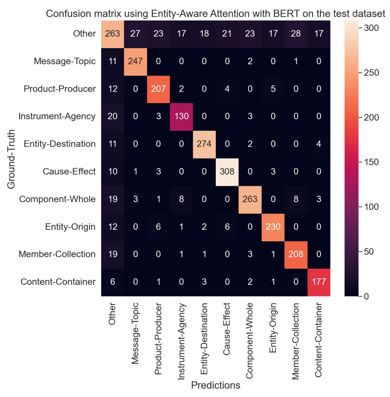
- A third pair is "Product-Producer" and "Instrument-Agency". The problem seems to be in wrongly predicting a "Instrument-Agency" sentence as "Product-Producer", as Entity-Aware Attention has very few errors when wrongly predicting it the other way around. The difference between these two classes is that one produces the product, and the other uses it. In most cases, we have some verb that gives away the particular type of action, like "build" for "Product-Producer", and "use" for "Intrument-Agency". However, there are cases where we only have the verb for one relation, but the sentence belongs to the other class. In other cases, we have verbs for both relations, but it first sees the one from the wrong class. And this seems to happen more for wrongly predicting as "Product-Producer".

- A fourth pair is "Content-Container" and "Entity-Destination". The dataset authors said that these two classes overlap a lot. We have a lot of "Entity-Destination" sentences talking about putting something into another thing. According to [4], the authors assigned these confusing sentences to one class or the other depending on the type of situation. If the sentence talks about some dynamic situation happening at the moment, it is assigned to "Entity-Destination". If the sentence talks about some static situation that already happened, it belongs to "Content-Container". Again, we have verbs that can give away the relation, like "full of" for "Content-Container" and "move into" for "Entity-Destination". The problem is when we encounter those confusing sentences. The model sees verbs like "put into", but the sentence talks about moving some entity and gets confused. We can see in the confusion matrices how we have more cases of wrongly predicting a "Entity-Destination" sentence as "Content-Container". However, CNN also has problems wrongly predicting it the other way around. Again, the Bi-LSTM and attention mechanism gather the information that helps predict those cases.

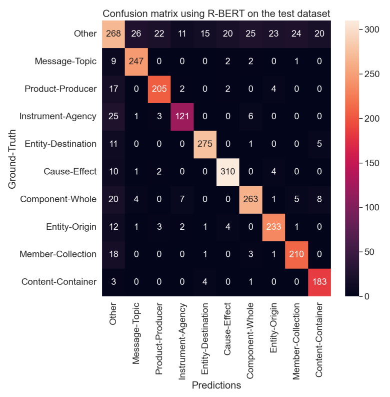
- The last pair that has problems in all three cases is "Instrument-Agency" and "Component-Whole", especially when wrongly predicting a "Component-Whole" sentence as "Instrument-Agency". In CNN, it also happens a lot the other way around. The two classes are confused because we can consider the instrument a functional part of the agent, especially if the agent refers to a job. "Component-Whole" has particular verbs that can give away the relation like "comprised of". The problem is that in most cases, we do not have that verb, but verbs like "use" that are more related to "Instrument-Agency". Therefore, even though the sentence belongs to "Component-Whole", in most cases, it talks about how the whole uses the component in some way, and the models predict it wrong.



(a) Attention Bi-LSTM BERT



(b) Entity-Aware Attention BERT



(c) R-BERT

Figure 7: Confusion matrices of the methods using contextual embeddings

Let's now look at the methods using contextual word embeddings. Looking at the numbers, we can see how we have more true positives and fewer false positives and false negatives. And this is because of the improved word embeddings that help disentangle some relations. However, we can also see that we still have some problems in some of the pairs we mentioned before, meaning that better embeddings alone do not solve the problem. In particular:

- "Other" still has bad results in all three methods, Attention Bi-LSTM BERT being the worst. If we compare the latter two, we see that overall, R-BERT has fewer false positives and false negatives. However, R-BERT seems to do it worse when wrongly predicting "Other" instead of "Instrument-Agency" and "Product-Producer". Considering the differences between both methods, the main one is the latent types Entity-Aware BERT considers when computing the sentence representation. Therefore, those classes seem to benefit from knowing the entities' type. When looking at particular sentences, we will see how it helps.

- If we look at the pair "Component-Whole" and "Member-Collection", we still have some errors when predicting "Member-Collection" instead of "Component-Whole". R-BERT has fewer mistakes, and the other two methods have reduced it thanks to contextual embeddings. The numbers of wrongly predicting "Component-Whole" instead of "Member-Collection" are maintained the same, meaning that we would need another type of information to fix them.

- If we look at "Product-Producer" and "Instrument-Agency", the contextual word embeddings have helped a lot in reducing the cases of wrongly predicting a "Instrument-Agency" sentence as "Product-Producer". They help even when we have a verb that should be more used to classify "Product-Producer", but the sentence belongs to "Instrument-Agency". We do not see significant differences between the three methods.

- If we look at the pair "Content-Container" and "Entity-Destination", we see again that the contextual embeddings seem to help in fixing the "Entity-Destination" sentences wrongly predicted as "Content-Container". Specially, we see fewer sentences mistaken as "Content-Container" in Entity-Aware BERT. If we compare the three methods, we can see how we still have differences between Attention Bi-LSTM BERT and Entity-Aware BERT, meaning that the entities' information seems to help. Between Entity-Aware BERT and R-BERT, there is no significant difference.

- If we look at the pair "Instrument-Agency" and "Component-Whole", we still have problems with this pair. The contextual embeddings do not seem to help that much when wrongly predicting "Instrument-Agency" instead of "Component-Whole". On the other way around, we seem to have some problems with Attention Bi-LSTM BERT and R-BERT. And more surprisingly, we have increased the number of errors in Attention Bi-LSTM BERT compared to the version with distributional embeddings. The reason why Entity-Aware BERT seems to do it better has to be the extra entity information. Therefore, we can distinguish cases that talk about a person using a tool rather than something being a functional part of another thing.

- Another pair is “Entity-Origin” and “Product-Producer”. We did not explain it when looking at the previous matrices because the other pairs had more errors. However, with the contextual embeddings, we have still not removed the errors in this last pair. All methods have errors when wrongly predicting a “Product-Producer” sentence as “Entity-Origin”. Entity-Aware BERT has more errors when doing it the other way around. The latent types seem to mislead the model. It is not strange to confuse this pair, as “Entity-Origin” refers to something that comes from another thing or place. We can think of the product coming from the producer in some way. Verbs like “obtained from”, “generated from”, or only “from” seem to be used in “Entity-Origin”. And it is not that strange encountering this preposition when talking about producing some product.

6.4 Examples of Sentences

In this subsection, we will present different sentences to demonstrate what we have seen in previous subsections about some classes doing a better job of predicting certain classes. We will mainly focus on comparing the methods using contextual word embeddings as they reach better results. Howver, we have seen how they still commit similar errors as those using distributional embeddings.

To perform these comparisons, we will leverage the attention weights, and the entity types in Entity-Aware Attention methods.

6.4.1 Other

We have seen how “Other” is the most difficult class to predict correctly. It is not that strange, as the authors of the dataset use it to assign everything that they could not fit into another. And some sentences are really difficult to predict, even for a human. Also, the official evaluation metric ignores that class, meaning that the authors do not care about it. In any case, we want to try to understand why all the models fail that much.

Let’s start by looking at false positives and see which methods predict them correctly or not.

- “A child is told a <e1>lie</e1> for several years by their <e2>parents</e2> before he/she realizes that a Santa Claus does not exist.”
  - Ground truth is “Product-Producer(e1, e2)”
  - All top-3 methods predicted it as “Other”

The ground truth of this sentence is “Product-Producer(e1, e2)”, meaning that “parents” produced a “lie”. All the methods predicted it as “Other” because most sentences having this class in the dataset refer to producing tangible objects. Entity-Aware BERT does not benefit from having the latent types because the entities do not refer to something that creates other things, like a person working in some industry or factory. Also, no verb could help the methods to predict it correctly. We have “by”, which with some other verb like “made”, could be of some use, but it does not help in this case.

- “Skype, a free software, allows a <e1>hookup</e1> of multiple computer <e2>users</e2> to join in an online conference call without incurring any telephone costs.”
  - Ground truth is “Member-Collection(e2, e1)”
  - Entity-Aware BERT predicted it correctly
  - R-BERT and Attention Bi-LSTM BERT predicted it as “Other”
  - Entity-Aware without BERT predicted it as “Other”

We can see how “users” represents non-functional members of “hookup”. Only Entity-Aware BERT predict it correctly, but not the other two. For simple curiosity, we wanted to check if Entity-Aware with distributional embeddings could predict it correctly, but it could not. Therefore, thanks to the contextual embeddings and the latent types, Entity-Aware BERT predicts it correctly. And this is because “users” and “hookup” can have multiple meanings, even appearing together in the same sentence, so we need to know the exact context and what they represent.

Let's now look at some false negatives. In most cases, the problem is that the models find some words always used to predict a particular class, and the models automatically wrongly label them as that class.

- "The <e1>kitchen</e1> was full of <e2>smoke</e2.>"
  - Ground truth is "Other"
  - All top-3 methods predicted it as "Content-Container(e2, e1)"

We can see how the ground truth is "Other" because even though the smoke is inside the kitchen (i.e. a delimited area of space), we cannot consider it to be physically stored, as this is not something someone put it in. Also, the smoke can easily escape through some opening in the door or the window. However, all the methods classify it as "Content-Container" because they see "full of" and do not care about the entities themselves. It is expected, as "full of" is mostly used in the dataset to denote that something is full of other things.

- "Eighteen <e1>studies</e1> were focused on older <e2>adults</e2> living at home."
  - Ground truth is "Other"
  - All top-3 methods predicted it as "Message-Topic(e1, e2)"

This sentence is quite confusing because we do not know if "studies" refers to some written communication act. We suppose it is not because otherwise, the ground truth should be "Message-Topic" as the topic is the older adults living at home. All methods predicted it as "Message-Topic" because they detected "focused on", which is used to denote the topic in other sentences of the dataset.

6.4.2 Component-Whole and Member-Collection

As we saw in the confusion matrices, all methods seem to have problems when trying to predict "Component-Whole" but end up predicting "Member-Collection". If we remember, we said that the main difference was that the member does not have any function in the collection, but the component has it as part of the whole.

- "My second question is regarding the sale of bones and <e1>hide</e1> of <e2>tigers</e2> going on in Delhi."
  - Ground truth is "Component-Whole(e1, e2)"
  - All top-3 methods predicted it as "Member-Collection(e2, e1)"

It is easy to see how "hide" refers to the skin of the tigers, but all methods predicted the sentence as "Member-Collection". And this is because they are not getting the correct meaning of "hide", which is not strange. The sentence could have used "skin", which allows all three methods to predict it correctly. However, the methods think the sentence talks about the hideout of a pack of tigers, even after using BERT embeddings. Maybe, that meaning is not that common in the dataset used to pre-train the language model, and this is why the methods do not seem to understand it, and they focus on "hide of" to predict it as "Member-Collection". Because we have other words like "bones", that should help in case the meaning of "hide" is understood.

- "Without wasting time, a medical <e1>team</e1> of the <e2>army</e2> rushed to the house of the militant and provided emergent medical treatment to her."
  - Ground truth is "Component-Whole(e1, e2)"
  - Attention Bi-LSTM BERT predicted it correctly
  - Attention Bi-LSTM without BERT predicted it as "Member-Collection(e1, e2)"
  - The others predicted it as "Member-Collection(e2, e1)"

The sentence belongs to “Component-Whole” as the army is divided into groups of people with different functions and tasks. Therefore, we cannot consider them members of a collection, as they have some function inside that “collection”. However, we can see how only Attention Bi-LSTM with BERT predicts it correctly. The same method without contextual embeddings fails to predict the class but at least matches the direction. The others are not even able to get the direction right. Those methods consider “army” as members of “team”. We believe they understand “army” as a group of people rather than the institution itself. Attention Bi-LSTM BERT predicts the sentence correctly because it does not get distracted by the extra entity information the other methods have. If we remember R-BERT, it is weighting equally the [CLS] embedding and the entities' embedding. Thus, it pays too much attention to the entities and not to other parts of the sentence. Focusing on the “the” in “the army” should help to know that the sentence talks about the institution.

6.4.3 Product-Producer and Instrument-Agency

We will spend less time in this pair, as the contextual embeddings solved most prediction errors, even though they cannot solve all of them if we consider how related these two classes are.

- “<e1>Carpenters</e1> build many things from <e2>wood</e2> and other materials, like buildings and boats.”
  - Ground truth is “Instrument-Agency(e2, e1)”
  - All three methods using distributional embeddings predicted it as “Product-Producer(e2, e1)”
  - All three methods using contextual embeddings predicted it correctly

The problem with this sentence is that some models get confused because they think that “Carpenters” created “wood” instead of using “wood” to build other things. Even though we have the words “build from” or “materials”, which should help, they still get confused. However, when switching to contextual word embeddings, the same methods are able to predict it correctly, meaning that the improved embeddings help to realize the relation talks about using wood to build things.

6.4.4 Content-Container and Entity-Destination

As we explained in the confusion matrices, these two classes are very easy to confuse, especially when considering what we have explained about the assignment depending on if the situation is dynamic or static. If we remember, CNN had a lot of mistakes in both directions, but the other two using distributional embeddings fixed the cases of wrongly predicting “Content-Container” as “Entity-Destination”. However, the errors when wrongly predicting “Entity-Destination” as “Content-Container” were still high. When switching to contextual embeddings, we reduced the latter errors, reaching the lowest numbers in Entity-Aware BERT and R-BERT. Therefore, the entities' information seems to help.

- “Patty the Milkmaid was going to market carrying her <e1>milk</e1> in a <e2>pail</e2> on her head.”
  - Ground truth is “Content-Container(e1,e2)”
  - CNN predicted it as “Entity-Destination(e1, e2)”
  - All other five methods predicted it correctly

It is easy to know that the sentence belongs to “Content-Container” by looking at the words “carrying in”, explaining why most methods predict it correctly. However, CNN predicts it as “Entity-Destination”. We cannot check it, but maybe the problem is that the model mainly looks at “going to” and cannot relate it with the entities because it cannot properly deal with dependencies outside the context defined by the kernel size. Even though we should be able to merge those contexts with the max-pooling and the non-linear layer, this method is not that effective at carrying information forward as an RNN.

- “Douglas obeyed the king, and the <e1>heart</e1> was enclosed in a silver <e2>casket</e2>.”
  - Ground truth is “Entity-Destination(e1,e2)”

  - Attention Bi-LSTM BERT predicted it as “Content-Container(e1, e2)”
  - The other top-2 methods predicted it correctly

This sentence is a little confusing because the ground truth is “Entity-Destination”, but the sentence talks about enclosing something. And this should relate to “Content-Container”. Thanks to the explanation about the dynamic and static situation we presented before, we understood why it belongs to “Entity-Destination”. For some reason, Attention Bi-LSTM BERT cannot predict it correctly, but the other two can. However, we believe this is due to some coincidence rather than a pattern. We have other sentences presenting the same confusion, and all top-3 methods predict it as “Content-Container”. The models need to relate the verb tense with the appearance of words like “already” to disambiguate the classes in these confusing cases, but they cannot do it.

6.4.5 Instrument-Agency and Component-Whole

If we remember, the difference between these two classes is that the first one implies the agent using the instrument, while the component is a part of the whole rather than using it. We saw that we had problems in both directions, especially in R-BERT and Attention Bi-LSTM BERT. However, Entity-Aware BERT was able to fix some of the “Instrument-Agency” wrongly classified as “Component-Whole”. And for some reason, Attention Bi-LSTM BERT generated more errors of this type than without BERT.

- “One design shows a <e1>farmer</e1> with a <e2>hoe</e2> and reads "Genuine Dirt Farmer Descendant".”
  - Ground truth is “Instrument-Agency(e2,e1)”
  - Entity-Aware BERT predicted it correctly
  - Entity-Aware without BERT predicted it correctly
  - Attention Bi-LSTM BERT and R-BERT predicted it as “Component-Whole(e2, e1)”

This sentence belongs to “Instrument-Agency” because a hoe is a tool a farmer usually uses. Entity-Aware BERT predicts it correctly because, thanks to the latent types, it knows that “farmer” is a job, and “hoe” is a tool. Also, we do not have some verb like “use” that could give away the relation. It is also interesting that the same method without BERT predicted it correctly. However, the other top-2 methods predicted it as “Component-Whole”. In some sense, we could consider the hoe a part of a farmer, as the farmer needs tools to do his job. For this reason, the other methods without the latent types cannot predict it correctly because of this lack of meaning in the sentence about using something.

- “Flanking or backing <e1>rudders</e1> are used by <e2>towboats</e2> and other vessels that require a high degree of manoeuvrability.”
  - Ground truth is “Component-Whole(e1,e2)”
  - Top-3 methods predicted it as “Instrument-Agency(e1, e2)”

This sentence belongs to “Component-Whole” because rubber is a component with an important functionality in a boat. However, all the methods failed and predicted it as “Instrument-Agency”. The problem here is the verb “used by”, which is more related to the mistaken class, and the models focus on it. Although they should understand the relation between the two entities, especially those using more entity-related information to predict the class, they are fooled.

6.4.6 Entity-Origin and Product-Producer

“Entity-Origin” talks about the origin of some product, including the place it comes from, while “Product-Producer” talks about who produced it. Thus, the difference can be subtle in some cases. When looking at the confusion matrices, we see that we have more problems when wrongly predicting “Product-Producer” as “Entity-Origin”. However, Entity-Aware BERT seems to have some more errors the other way around.

- “In recent years, most <e1>floppies</e1> have shipped pre-formatted from the <e2>factory</e2> as DOS FAT12 floppies.”
  - Ground truth is “Product-Producer(e1,e2)”
  - Top-3 methods predicted it as “Entity-Origin(e1, e2)”

All top-3 methods predicted it as “Entity-Origin” because they encounter “shipped from”, and understand the “floppies” came from the “factory”. Also, in most “Product-Producer” cases, the producer appears as some person rather than a generic concept like “factory”.

- “The full <e1>press release</e1> from the <e2>studio</e2> is pasted at the bottom of this post.”
  - Ground truth is “Entity-Origin(e1,e2)”
  - Entity-Aware with and without BERT predicted it as “Product-Producer(e1, e2)”
  - R-BERT and Attention Bi-LSTM BERT predicted it correctly

This sentence refers again to the place where the product comes from. We have the preposition “from” that can appear when predicting both classes. However, we do not have a verb to distinguish them. R-BERT and Attention Bi-LSTM BERT predicted it correctly, but both Entity-Aware methods did not. The problem here is the latent type of “studio”, which makes the model think it is the producer rather than the place it was produced.

6.4.7 Another case where Entity-Aware BERT fails

- “Researchers put <e1>bio-energetics</e1> into <e2>bio-magnification</e2>.”
  - Ground truth is “Entity-Destination(e1,e2)”
  - Entity-Aware BERT predicted it as “Other”
  - R-BERT and Attention Bi-LSTM BERT predicted it correctly

This sentence belongs to “Enitty-Destination”, even though the first entity is not a physical object moved to some destination, which is usually the case in this class. We have the word “put into”, which helps predict this class. R-BERT and Attention Bi-LSTM BERT predicted it correctly, but Entity-Aware BERT did not. The problem is the latent types, which make the model think that the first entity is not something usually moved to the second. And this is because the entities are not an object and a place but rather abstract concepts.

6.4.8 Cause-Effect with Attention Bi-LSTM BERT

From the F1 scores of each class, Attention Bi-LSTM BERT achieved the best performance in “Cause-Effect”, followed by R-BERT. For this reason, we want to present some cases where the former does it better than the latter. Looking at the confusion matrices, we can see how R-BERT commits more false positives when the ground truth is “Other”.

- “<e1>Asteroid</e1> threatened Earth with <e2>disaster</e2>.”
  - Ground truth is “Other”
  - Attention Bi-LSTM BERT predicted it correctly
  - R-BERT predicted it as “Cause-Effect(e1, e2)”

The main difference between these two methods is that R-BERT uses the embeddings of the entities to predict the class besides the [CLS] token embedding. On the other hand, Attention Bi-LSTM BERT only uses the sentence representation from the attention mechanism. We know that the embedding of the [CLS] token should be better, but here R-BERT seems to fail while the other predicts it correctly. The sentence belongs to “Other” because even though it talks about some asteroid and has the word “disaster”, the most important word is “threatened”, meaning that an effect did not happen. However, as R-BERT has the same weight in the [CLS] embedding and the entities' embedding, the embeddings of "Asteroid" and "disaster", which usually refers to a bad effect, belittles the information of "threatened" contained in the [CLS] embedding, causing a wrong prediction.

- "The <e1>guard</e1> pushes the passports back with a <e2>chuckle</e2> and ignores them."
  - Ground truth is "Other"
  - Attention Bi-LSTM BERT predicted it correctly
  - R-BERT predicted it as "Cause-Effect(e1, e2)"

We present another sentence having the same problem. The ground truth is "Other" because no other relation can fit. However, R-BERT predicts it as "Cause-Effect". We believe R-BERT puts too much attention on the entities and not on other words. The model thinks that "guard" is making someone else "chuckle", but it is not. Attention Bi-LSTM BERT predicts it correctly because it focuses on the other words and not forcing more weight on the entities.

7 Conclusions

In this project, we have been able to implement and compare different methods to perform RE in the SemEval dataset. We have reached, in most cases, a similar performance to that reported by the authors and even improved it in some cases. When trying different distributional embeddings, we could improve the results in the CNN method. However, we could not in the others, as the authors already used the best distributional embeddings we could find. Of course, the best improvement came when switching to contextual embeddings, following what we would expect from what is changing the state of the art of most deep learning problems.

However, and this relates to the goals' accomplishment, we could not finish one of the methods we decided to implement at the beginning of the project, so we discarded it. It was a method based on SDP [16], but we had problems with the dependency parser. We could not find one without errors, so we had to discard the sentences the parser could not process. It also gave bad classification results in the SemEval dataset, so we gave up.

Another goal we could not achieve was trying more features in the methods using distributional embeddings. As we explained in the project goals, we wanted to try different handcrafted features and modify the model architecture to improve performance. We tried with CNNs but did not have enough time to try it on the other methods. If we had done it, we would not have had enough time to perform the comparisons, which was the main point of the whole project.

After looking at the performance of the different models, we have seen what we were expecting about BERT-based models beating the others. However, we have also seen how in some cases having better embeddings is not enough. Sometimes, we need certain extra features to disambiguate certain classes. This dataset has very few classes, and they are very generic. Even for humans, we encountered some sentences that we did not properly understand why it was labelled as it was.

In future work, we would like to try more modern architectures now that we have focused on older ones. We have seen how the top performers use BERT, so it would be interesting to try others. Also, we would like to see if we can improve the performance of some of the models by trying those features that we had not enough time to try.

References

[1] Amir D. N. Cohen, Shachar Rosenman, and Yoav Goldberg. "Relation Extraction as Two-way Span-Prediction". In: *CoRR* abs/2010.04829 (2020). arXiv: `2010.04829`. URL: `https://arxiv.org/abs/2010.04829`.

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: `1810.04805`. URL: `http://arxiv.org/abs/1810.04805`.

[3] Claire Gardent et al. "Creating Training Corpora for NLG Micro-Planners". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 179–188. DOI: 10.18653/v1/P17-1017. URL: https://aclanthology.org/P17-1017.

[4] Iris Hendrickx et al. "SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 33–38. URL: https://aclanthology.org/S10-1006.

[5] Joohong Lee, Sangwoo Seo, and Yong Suk Choi. "Semantic Relation Classification via Bidirectional LSTM Networks with Entity-aware Attention using Latent Entity Typing". In: *CoRR* abs/1901.08163 (2019). arXiv: 1901.08163. URL: http://arxiv.org/abs/1901.08163.

[6] Cheng Li and Ye Tian. "Downstream Model Design of Pre-trained Language Model for Relation Extraction Task". In: *CoRR* abs/2004.03786 (2020). arXiv: 2004.03786. URL: https://arxiv.org/abs/2004.03786.

[7] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[9] Sebastian Riedel, Limin Yao, and Andrew McCallum. "Modeling Relations and Their Mentions without Labeled Text". In: *Proceedings of the 2010th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*. ECMLPKDD'10. Barcelona, Spain: Springer-Verlag, 2010, pp. 148–163. ISBN: 3642159389. DOI: 10.1007/978-3-642-15939-8_10. URL: https://doi.org/10.1007/978-3-642-15939-8_10.

[10] Bryan Rink and Sanda Harabagiu. "UTD: Classifying Semantic Relations by Combining Lexical and Semantic Resources". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 256–259. URL: https://aclanthology.org/S10-1057.

[11] Yatian Shen and Xuanjing Huang. "Attention-Based Convolutional Neural Network for Semantic Relation Extraction". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 2526–2536. URL: https://aclanthology.org/C16-1238.

[12] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. "Word Representations: A Simple and General Method for Semi-Supervised Learning". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 384–394. URL: https://aclanthology.org/P10-1040.

[13] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[14] Shanchan Wu and Yifan He. "Enriching Pre-trained Language Model with Entity Information for Relation Classification". In: *CoRR* abs/1905.08284 (2019). arXiv: 1905.08284. URL: http://arxiv.org/abs/1905.08284.

[15] Kun Xu et al. "Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling". In: *CoRR* abs/1506.07650 (2015). arXiv: 1506.07650. URL: http://arxiv.org/abs/1506.07650.

[16] Yan Xu et al. "Improved Relation Classification by Deep Recurrent Neural Networks with Data Augmentation". In: *CoRR* abs/1601.03651 (2016). arXiv: 1601.03651. URL: http://arxiv.org/abs/1601.03651.

[17] Daojian Zeng et al. “Relation Classification via Convolutional Deep Neural Network”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.* Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 2335–2344. URL: `https://aclanthology.org/C14-1220`.

[18] Peng Zhou et al. “Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 207–212. DOI: `10.18653/v1/P16-2034`. URL: `https://aclanthology.org/P16-2034`.