Assignment4 – Report

Dorin Keshales – 313298424

In this assignment, I chose to take a learning-based approach by using the Transfer Learning technique.

Link to a colab notebook showing the model's training:
https://colab.research.google.com/drive/15QFn0N0y5g_Gxno5TgyE47_WLuJVkzlJ?usp=sharing

Problem background

In this assignment I chose to identify the Work_For relation.

Relation Extraction aims at extracting semantic relations between entities from a given text. In this assignment, we focus on extracting binary relations, i.e. relations between exactly two entities, from sentences. Given a sentence, the task is to decide whether the sentence contains a mention of the Work_For relation, and if yes - to extract it and the two corresponding entities.

This problem is best framed as a binary classification, which predicts whether a sentence with two target entities will be in one of two classes – { Work_For relation – 1, not Work_For relation – 0}. That is, the model will predict whether the relation between two target entities in the sentence is a Work_For relation or not. The approach taken is supervised learning – each dataset was provided with an annotations file containing the ground truth of the data.

The training set consists of 354 sentence examples of which 109 are Work_For relations on which the model can train and the dev set consists of 340 sentence examples of which 106 are Work_For relations that the model can use for evaluation.

I applied the labels {Work_For relation – 1, not Work_For relation – 0} to each pair of PER and ORG entities found in the sentence, for each of the sentences. It is clear that not all PER and ORG entity pairs have a Work_For relation between them. So by using the train annotations in training I can distinguish most of the time (It does not work 100% of the time) between positive and negative examples of Work_For relation in between PER and ORG entity pairs, where positive example is an example in which in a sentence there is a pair of PER and ORG entities and the relation between them is a Work_For relation, while a negative example is an example where there is a pair of PER and ORG entities, as in the positive example, but the relation between these entities is not a Work_For relation (there may be no relation between these entities at all). By labeling these examples using this additional information, the model can learn the differences between a pair of PER and ORG entities who's the relation between

them is a Work_For relation and a pair of PER and ORG entities who's the relation between them is not a Work_For relation.

Challenges:

- Small dataset

  Trying to build a model that can generalize beyond the training set from a small amount of examples while maintaining a close supervision that the model does not overfit on the training set.

- Mismatches between extracted entities by NER taggers to the entities appeared in the annotations file

  Some of the entities appearing in the annotations file are 'different' from those extracted by spaCy or by other NER taggers I've tried to use during this work, where 'different' means:

  1. Different chunking:

     - spaCy: 'the Federal Home Loan Bank'
       Stanza: 'the Federal Home Loan Bank of San Francisco'
       Flair: 'Federal Home Loan Bank of San Francisco'
       annotations: 'Federal Home Loan Bank'

     - spaCy: 'U.S. Circuit Court of Appeals'
       Stanza (CoreNLP): '2nd type: ORDINAL
       'U.S. Circuit Court of Appeals' type: ORG
       annotations: '2nd U.S. Circuit Court of Appeals'

     - spaCy: 'the European Space Agency'
       Stanza: 'the European Space Agency'
       annotations: 'European Space Agency'

     - spaCy: 'Higgins'
       Stanza: 'Higgins'
       Flair: 'Higgins'
       annotations: 'Mrs. Higgins'

  2. Different tagging:

     - spaCy: 'Winter , 53' type: DATE
       annotations: 'Winter' type: PERSON

     - spaCy: 'Epps' type: ORG'
       annotations: 'Epps' type: PERSON

     - Stanza: 'Farley' type: PERSON
       Flair: 'Farley' type: PER

annotations: ‘Farley’ type: ORG

3. Missing entities:

- spaCy: ‘Rollei’ type: O
  annotations: ‘Rollei’ type: ORG
- spaCy: ‘Higgins’ type: O
  annotations: ‘Higgins’ type: PERSON
- Flair: ‘Health and Human Services’ type: O
  annotations: ‘Health and Human Services’ type: ORG

This entities mismatch is the main problem of this assignment and it has a direct influence on the final results. If you do not find a good enough solution to this problem, the final score will most likely be quite low.

Model description

NER Tagger:

I will detail briefly (I will try) about the NER taggers I tried through the work on this assignment - what were the problems with each NER tagger and how were they solved as I moved from one NER tagger to the other:

spaCy

I started with spaCy's NER tagger, but I had a lot of mismatches, caused by spaCy, between the entities extracted by spaCy and the entities in the annotations file, like:

- Incorrect tagging of many relevant entities (PERSON / ORG) as 'O'.
- Identify a person's name as the name of an organization.
- Severe problem of NE chunking for ORG entities.
- When spaCy's tokenizer tokenize a sentence, it separates the dot at the end of the entity from the entity itself, which I think makes sense, but in the case of this assignment the dot should have remained part of the entity itself after the tokenization.

I really tried a lot to address all the issues I mentioned, including trying to use person and organization gazetteers and trying to manually chunk the NE myself with spaCy's NER tagger, meaning not to use the spaCy 'sent.ents' option. But all my attempts to solve these problems came to naught, as there were too many problems to solve.

Stanza (CoreNLP)

So, I decided to try another NER tagger by Stanza, which includes a Python interface to the CoreNLP Java package. With Stanza I was able to solve the issue of the tokenized dot in spaCy, since the input to Stanza's tokenizer can be pre-tokenized, i.e. the input to Stanza can

be the list of the words in the sentence obtained from the split function (the split function allows to keep the dots attached to the words). So, with pre-tokenized input to Stanza we can overcome this issue of spaCy. Another spaCy's issue solved by Stanza is the mis-tagging of a person's name as an organization name, meaning that Stanza is better than spaCy at identifying people's names.

However, Stanza's NER tagger also has problems that cause a mismatch between the entities it has extracted and the entities in the annotations file:

- Stanza, as well as spaCy, has a serious chunking problem of ORG entities, and it also incorrectly tags some words of an ORG entity as O or other tags. That is, Stanza is not particularly good at identifying the names of the organizations.
- Extracted ORG entities by Stanza do not include the determiner ‘the’ that appears just before the organization name in the sentence (which to my opinion is fine), which causes about 22 potential 'Work_For' ratios to 'fall' when evaluating the model. In other words, I lose recall over it.
- Some of the PERSON entities extracted by Stanza contain " ‘s " at the end of them, which makes me lose recall here, as well.

So, in an attempt to solve the last two problems I mentioned, I constructed some basic rules: one for removing " 's ", if exists, at the end of a PERSON entity and second rule in which I add the determiner 'the' to ORG entities if the determiner appeared before the organization name in the sentence (Much later after replacing the Stanza’s NER tagger with my final NER tagger, we were instructed in the piazza to ignore Stanza’s NER tagger when evaluating the model's score). The rules actually solve these problems and I started to see an improvement in the F1 score - 64%.

Flair

Although using Stanza's NER tagger made me get a better F1 score, I was still looking for a better NER tagger with which I would not even need the manual rules of which I used to solve some of Stanza's problems. And indeed, I found such a NER tagger – flair's NER tagger. Unlike the first two NER taggers that support 18 different entity types, the flair's NER tagger only supports 4 entity types (PER, ORG, LOC, MISC), which I think contributes to its success over the previous tags. Flair's NER tagger, like Stanza, can get a pre-tokenized input, it has amazing NE chunking capability that allows him to overcome spaCy's and Stanza's main problem, it is able to extract the ORG entities together with the determiner 'the' and to extract the PERSON entities without the suffix of " ‘s ". And all this without any need for manual rules for that.

Overall, flair solves most of the problems found in spaCy and Stanza but is still not perfect due to a slight mis-tagging problem. Even so, it's still the best NER tagger I have been able to find that has such a high compliance rate to the entities listed in the annotations file. Therefore, I chose to use the flair's NER tagger for this work.

The Pre-processing phase:

Input - because I used the flair's NER tagger, the input to my program is the txt file.

Relation – Work_For

Relevant entity pair for the relation – (PER, ORG)

Features – only the NE, since the model (shown below) doesn't take or should take any features.

As first step, I read the input file and saved in a dictionary each sentence id and the sentence associated with it. I also read the annotations file and each annotation in it is saved as a tuple of (sentence id, subject, relation, object, sentence) to a list.

For processing the raw input data (from the first step), I use flair's NER tagger on the sentence to extract the entities from the sentence. For each input sentence, I keep a dictionary with the following information:

- 'id' - the sentence identifier.
- 'sent' - the sentence associated with the sentence identifier.
- 'entities' - the entities identified in the sentence by flair.
- 'words' - the list of words in the sentence obtained from the 'split' function (by space).

Then, in order to prepare the dataset for the training phase i.e. the input samples for the model, which consisting of a sentence and two target entities, I loop over the processed sentences (the sentences dictionaries created in the last step). for each sentence, I extract from the entities list all the PER and ORG entities identified in the sentence, along with their locations in the sentence. Flair gives the position of an entity as a token position (i.e. from token number 5 to token number 7 for example) and not the character position (i.e. from character number 27 to character number 39 for example). So, for each PER entity and ORG entity I convert its token position into character position.

Next, for each sentence I use the itertools.product operation to get all the permutations of pairs of PER and ORG entities. I save this permutations list in a tuple together with the sentence dictionary, i.e. (sentence dict, permutations), from which I can extract later, in the training phase, the examples to the model, since each entity pair from the permutations list together with the sentence itself is a sample to the model (actually, the input to the model is the sentence with specially marking of the entities in it - will be explained in more detail later).

The model:

Introduction

Over the past few years, Transfer Learning has led to a new wave of state-of-the-art results in Natural Language Processing (NLP). Transfer Learning's effectiveness comes from pre-training a model on abundantly-available unlabeled text data with a self-supervised task, such as language modeling or filling in missing words. After that, the model can be fine-tuned on smaller labeled datasets, often resulting in (far) better performance than training on the

labeled data alone. The recent success of Transfer Learning was ignited in 2018 by GPT, ULMFiT, ELMo, and BERT.

BERT, or Bidirectional Encoder Representations from Transformers, is one of the most popular transformer-based models, as for it obtains state-of-the-art results on a wide array of NLP tasks, and is considered a major breakthrough in NLP. BERT is a deeply bidirectional, unsupervised language representation, pre-trained using a combination of Masked Language Modeling objective and Next Sentence Prediction task on a large corpus comprising the Toronto Book Corpus and Wikipedia. It consists of deep Transformer layers, enabling it to capture dependencies across long sequences as well as sentence-level understanding. Its bidirectional representation is conditioned on both left and right context at the same time in all layers. Those features make it an excellent model to produce contextual representations for texts. The pre-trained version of BERT (that is, the weights obtained from the Masked Language Modeling and Next Sentence Prediction training routines outlined above) are used as starting weights for a supervised learning phase. During a fine-tuning phase, the previously learned representations are used as a baseline for a problem-specific training. Fine-tuning of BERT is always associated with a particular practical task such as for example Question Answering.

So, as mentioned earlier, in this assignment I chose to use the Transfer Learning technique by using BERT model. That is, I picked up pre-trained BERT language model and fine-tuned it for few epochs on the given Relation Extraction dataset.

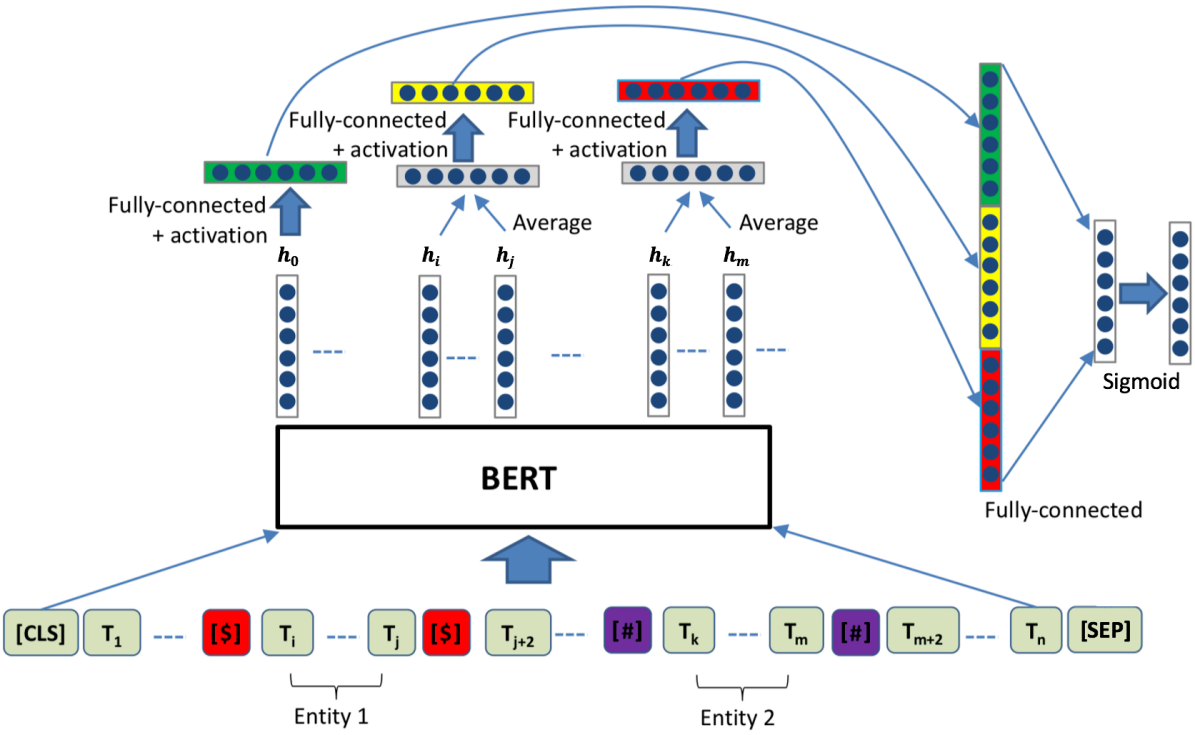
The task of Fine-Tuning a network is to tweak the parameters of an already trained network so that it adapts to the new task at hand. There are different fine-tuning techniques. In this project, I chose to examine and compare two main fine-tuning techniques on BERT-Base-Uncased model, released by Google research in 2018, with the Relation classification task using the provided dataset. The first technique is to train the entire architecture. That is, to further train the entire pre-trained model together with the additional task-specific layers on the dataset and feed the final output to a sigmoid layer. In this technique, the error is back-propagated through the entire architecture and the pre-trained weights of the model are updated based on the new dataset. The second technique is to train some layers while freezing others. That is, to train it partially – for example to keep the weights of the initial layers of the model frozen while retraining only the higher layers. This technique became popular, among other reasons, following observations that have been made in NLP literature regarding pre-trained language models. For example, Clark et al. (2019) analyzed BERT's attention and observed that the bottom layers attend broadly, while the top layers capture linguistic syntax. Kovaleva et al. (2019) found that the last few layers of BERT change the most after task-specific fine-tuning.

To get to the bottom of this assignment, I tried about 8 different settings of the second fine-tuning technique in addition to experimenting with the first fine-tuning technique, in order to find the best fine-tuning settings of BERT for this task.

General approach

For this work, I got inspired by the paper 'Enriching Pre-trained Language Model with Entity Information for Relation Classification' written by Shanchan Wu and Yifan He, and I decided

to try and replicate the RBERT model introduced in this paper, while making the necessary changes in order to fit it to my binary classification task and the amount of available data I have got for it, hoping to achieve good performance as reported in the paper.
(The figure below was taken from the paper but was edited by me, so that it will represent the architecture I actually implemented. I also changed some of the original notations for my convenience - I used these new notations in the code as well).



The model architecture:

For a sentences identified with two target entities e1 (PER entity) and e2 (ORG entity), in order to make BERT capture the location information of the two entities, at both the beginning and end of the first entity, I insert a special token '[$]', and at both the beginning and end of the second entity, I insert a special token '[#]'. Additionally, the special '[CLS]' token is added at the beginning of each sentence and the special '[SEP]' token at the end.

Given a sentence S with entity e1 and e2, suppose its final hidden state output from BERT is h. Let vectors h_i to h_j represent the final hidden state vectors from BERT for entity e1 and let h_k to h_m represent the final hidden state vectors from BERT for entity e2. For each of the two target entities I perform the average operation to get a vector representation. To each of the result vectors I apply dropout and then an activation operation (i.e. tanh). Then, I add a fully connected layer to each of the two vectors, and the output for e1 and e2 is H_1 and H_2 respectively. This process can be mathematically formalized as in the below Equations:

$$H_1 = W_1\left[tanh\left(\frac{1}{j-i+1}\sum_{t=i}^{j} h_t\right)\right] + b_1$$

$$H_2 = W_2 \left[tanh\left(\frac{1}{m-k+1} \sum_{t=k}^{m} h_t \right) \right] + b_2$$

W_1 and W_2, b_1 and b_2 share the same parameters. In other words, we set $W_1 = W_2$, $b_1 = b_2$. For the final hidden state vector of the first token (i.e. '[CLS]'), I also apply dropout, an activation operation and then a fully connected layer, which is formally expressed as:

$$H_0 = W_0 \left[tanh(h_0) \right] + b_0$$

Matrices W_0 , W_1, W_2 have the same dimensions, i.e. $W_0 \epsilon R^{d \times d}$, $W_1 \epsilon R^{d \times d}$, $W_2 \epsilon R^{d \times d}$, where d is the hidden state size from BERT. W_0 and b_0 doesn't share the same parameters as W_1, W_2 and b_1, b_2.

Once the final vector representations for both e1 and e2 entities and the '[CLS]' token are generated, we concatenate H_0 , H_1 , H_2 (let the final concatenated vector be H) and then add a fully connected layer and a sigmoid layer - for final classification of whether the relation between the two target entities (e1, e2) is 'Work_For' relation or not.
This can be formally expressed as:

$$H = W_3 \left[concat(H_0, H_1, H_2) \right] + b_3$$

$$p = sigmoid(H)$$

where $W_3 \epsilon R^{2 \times 3d}$ (In binary classification we have only 2 classes), and p is the probability output.

*The dropout before each fully connected layer is applied only during training.

<u>The changes made from the original RBERT:</u>

- In order to make BERT capture the location information of the two entities, at both the beginning and end of the first entity (that is, the PER entity), I insert a special predefined token '[$]', instead of the already existing token '$', which they used in the original paper. For the second entity (that is, the ORG entity), as well, I insert a special predefined token '[#]' at both the beginning and end of it, instead of the already existing token '#', which they used in the original paper. I defined these new tokens as special tokens to the tokenizer, so It won't tokenize them as part of the tokenization process. In addition, because these tokens are manually configured to the tokenizer, I had to resize BERT's token embeddings table so that the model can learn representation for them in the training process. The model needs to learn a good representation of these tokens in order to first identify the target entities in the sentence so that it can correctly classify the relation between these entities later. This is a crucial part in order for the model to be able to learn and achieve good performance on this task.

- In my model I kept the special token '[SEP]' at the end of the tokenized sentence and didn't omit it as they did in the paper.

- When applying the average operation to the sum of the hidden output vectors of each of the two target entities to obtain a vector representation for each, I omitted the hidden output vectors of the special tokens '[$]', '[#]' from the calculation. That is, unlike the paper's writers who included the hidden output vectors of the special tokens '[$]', '[#]' in the calculation, I didn't consider the special tokens as part of the entities themselves.

- In this assignment, the task in need is binary classification - 1 if the 'Work_For' relation was identified in the sentence and 0 otherwise. So, I used the Sigmoid function instead of the Softmax function at the end of the neural network in order to make the classification. As a side note, the Sigmoid is applied on the model output in the nn.BCEWithLogitsLoss function –Binary Cross Entropy Loss function – which is the loss function I used for this task.

- As mentioned above, I changed some of the original notations from the paper, the changes are as follows:

  - Instead of the notation of $H_0,\ H_i - H_j,\ H_k - H_m$ for the hidden output vector of the '[CLS]' token and the hidden output vectors of the tokens composing each of the target entities, respectively, I used the following notations $h_0,\ h_i - h_j,\ h_k - h_m$.

  - And instead of the notation of $H'_0,\ H'_1,\ H'_2$ for the processed vectors and h'' for the concatenation result, I used $H_0,\ H_1,\ H_2$ and H notations, respectively.

Experimental details:

I used HuggingFace's PyTorch implementation of BERT. The pre-trained BERT model used is BERT-base-uncased, which contains around 110M parameters. The BERT fine-tuning configuration selected for this task is the first fine-tuning technique presented above. That is, all BERT parameters (in all layers) are not frozen in training but fine-tuned together with the additional neural network layers on top of BERT. The model is trained with AdamW optimizer included in HuggingFace's implementation, which is an optimizer with weight decay fixed that can be used to fine-tuned models (I didn't use weight decay in this work). The seed value is set to 42 and the max sequence length is set to 512 as Bert-Base's default setting. Since the datasets are quite small I didn't use batches but passed one sample to the model at a time, i.e. batch size of 1. The conducted experiments included tuning of the following hyperparameters: learning rate, number of training epochs and dropout rate. The dropout rate used in the paper is 0.1, but since the dataset provided for this task is quite small, I used dropout rate of 0.2 to reduce overfitting on the training set and consequently help the model better generalize on the dev set and hopefully also on the test set.
All experiments in tuning the hyperparameters were conducted in about 8 different fine-tuning settings that I tried, in order to find the best fine-tuning setting for this task based on the particular dataset provided. In total, I conducted more than 100 experiments during this assignment. I ran each experiment on the CPU of Google Colab platform, which allowed me to run several experiments in parallel.

Parameter Settings:

Pretrained BERT:	bert-base-uncased
Number of epochs:	8
Batch size:	1 (as the dataset is very small)
Activation:	Tanh
Optimizer:	AdamW
Learning rate:	2e-5
Dropout rate:	0.2
Loss Function:	BCELoss (binary Cross Entropy loss)

The training phase:

In the training phase, I go through the training set that was built in the pre-processing phase so that each element in the dataset appears as a tuple of (sentence_dict, permutations) representing the current sentence and all the permutations of PER and ORG entities which might have between them a relation of Work_For. Each pair of PER and ORG entities is considered as an example to the model – they art the target entities in which between we check whether the relation is a relation of Work_For or not.
The input for BERT should be a tokenized sentence with the target entities marked up in it, using the special tokens I defined for this matter – '[$]' at the beginning and at the end of the PER entity and '[#]' at the beginning and at the end of the ORG entity.
Before marking the pair of PER and ORG entities found in the sentence by flair, I use the annotations from the train annotations file in order to fix possible chunking mistakes of flair's NER tagger.

The way I do it is as follows:

For each sentence and two target entities (PER, ORG) extracted by Flair, I look in the train annotations under the same sentence id, if the two current target entities have a high matching rate for a pair of entities (PER, ORG) from the annotations file, when a 'high match rate' is set to a 75% match rate - I have found that this rate is good enough to catch the relevant matches from the annotations while filtering out most of the discrepancies. The match rate is calculated using python's SequenceMatcher.ratio(), which gives a score between [0,1] for the similarity between two strings. In my approach, , I require that each of the target entities in the pair (PER, ORG) have a 'high matching rate' with the corresponding entity in the pair of entities from the annotations. This can be formally expressed as:

```
# For each annotation in the train annotations
for (sent_id, subject, rel, object, _) in train_annotations:

    # Compute the match rate between e1 to subject and e2 to object.
    person_ent_match_rate = SequenceMatcher(None, subject, e1).ratio()
    organization_ent_match_rate = SequenceMatcher(None, object, e2).ratio()
```

```
# If a match was found
if sent_id == sentence['id'] and person_ent_match_rate >= 0.75 and
                                organization_ent_match_rate >= 0.75:
```

where `e1` and `e2` are the PER and ORG entities identified by flair, respectively.
If we do find that the current two target entities have a high match rate for a pair of entities (PER, ORG) from the annotations file, then I use the matching entities from the annotations as the new pair of target entities – `(e1, e2)`. This is done in order for the model to train on the right and full-chunked entities, but this approach also helps me extract whether the relationship between the target entities is 'Work_For' or not. That is, if I found a high match rate between `(e1, e2)` and a pair of entities `(subject, object)` from the annotations file under the same sentence id, then I also know what is the relation between the entities by the relation between the entity pair from the annotations found as a match, so if the relation is 'Work_For' then the label for this training example will be 1 and for any other relation between the entities the label will be 0. If there is no match for `(e1, e2)` entity pair in the annotations, then the relation between these entities is set to 'NIL' and therefore the label for this example will be 0.

Of course, this approach may not work in some specific cases, but I still found it helpful in improving the model performance.

I just want to emphasize that this approach is done for training examples only, for the purpose of effective training of the model.

After that thing, all is left to do to make the train sample as input to the model is marking the target entities (the original `e1` and `e2` entity pair if no match was found, or the entity pair `(subject, object)` from the annotations if a match was found between them and the original entity pair `(e1, e2)`) in the sentence using the special predefined tokens.
When the input to the model is finally ready, I pass it through the HuggingFace's tokenizer for the tokenization process and forward the tokenized input to the model together with indexes representing the position of each of the target entity.

Next, I compute the loss (using BCEWithLogitsLoss which is binary Cross Entropy loss) of the model's output comparing to the label of the example (which was set as described above), I calculate the gradients and update the parameters using the optimizer. In order to reflect how well the model learns and improves from one epoch to another, I compute at the end of a full pass over the training examples the precision, the recall and F1 score from the model predictions during this training epoch. Before moving on to the next epoch, I evaluate the model on the dev set to see whether the model succeed in generalizing what it learned on the dev set.

This process is done in every epoch for a total of 8 training epochs.

Error analysis

Recall errors:

#	Error type	Gold	Prediction	Analysis
1	Mis-tagging of flair	1. sent1232 Stephen Greiner Work_For Farley 2. sent1265 Hernandez Galicia Work_For Mexican Oil Workers Union 3. sent1394 Robert Thompson Work_For Eaglebrook 4. sent1561 Al Comproni. Work_For Massachusetts Department of Public Health 5. sent2282 Roger Mauro Work_For Ancora State Psychiatric Hospital 6. sent2647 Jeffrey Laitman Work_For Mount Sinai School of Medicine.	No prediction	1. flair tagged 'Farley' as PER instead of 'ORG'. 2. flair tagged 'Hernandez Galicia' as ORG instead of 'PER'. 3. flair tagged 'Eaglebrook' as PER instead of 'ORG'. 4. flair tagged 'Al Comproni.' as ORG instead of PER. 5. flair tagged 'Ancora State Psychiatric Hospital' as LOC instead as 'ORG', probably because of the entity's start 'Ancora State' which falls under the entity type of LOC. 6. flair tagged 'Mount Sinai School of Medicine' as LOC instead as 'ORG', probably because of the entity's start 'Mount Sinai' which I assume is a location.
2	Flair didn't identify one of the entities	1. sent1948 Richard P. Kusserow Work_For Health and Human Services 2. sent2906 Nikolai Kuznetsov Work_For Navy 3. sent2990 Juppe Work_For French Government	No prediction	1. Flair didn't identify the 'Health and Human Services' entity. 2. Flair didn't identify the 'Navy' entity. 3. Flair didn't identify the entity 'French Government' – only identified the entity French and as a MISC.
3	Flair's NE chunking	1. sent1480 Dennis DeConcini Work_For Senate Appropriations treasury and postal subcommittee 2. sent1489 Edward Roybal Work_For House Appropriations treasury and postal subcommittee 3. sent1517 Mrs. Higgins Work_For Marine Corps 4. sent1517 Mrs. Higgins Work_For Pentagon 5. sent1585 Allen Harker Work_For Coast Guard 's 8th District headquarters 6. sent2021 Fran Sepler Work_For state Crime Victim and Witness Advisory Council. 7. sent1506 Vaughan Work_For Air Force	1. sent1480 Dennis DeConcini Work_For Senate Appropriations 2. sent1489 Edward Roybal Work_For House Appropriations 3. sent1517 Higgins Work_For Marine Corps 4. sent1517 Higgins Work_For Pentagon 5. sent1585 Allen Harker Work_For Coast Guard 6. sent2021 Fran Sepler Work_For Crime Victim and Witness Advisory Council. 7. sent1506 Vaughan Work_For Army Air Force	1. Flair identified part of the entity: Senate Appropriations type: ORG 2. Flair identified part of the entity: House Appropriations type: ORG 3. Flair identified part of the entity: Higgins type: PER 4. Flair identified part of the entity: Higgins type: PER 5. Flair identified part of the entity: Coast Guard type: ORG 6. Flair identified part of the entity: Crime Victim and Witness Advisory Council. Type: ORG 7. The entity Flair identified includes more words in it – Army Air Force – type: ORG.
4	An understandable tagging of flair	1. sent1411 Will Carlson. Work_For Michigan State 2. sent1566 Linda Schmitt Work_For Becton Dickinson	No prediction	1. flair tagged 'Michigan State' as LOC, which is a correct tagging. Most of the times the intention is the state as a location and not as a working place.

	conflicts with the annotations tagging			2. flair tagged 'Becton Dickinson' as PER instead as 'ORG', this seems to happen because both Bacton and Dickinson are family names (of the organization co-founders).
5	No reasonable explanation	1. sent1514 Higgins Work_For United Nations 2. sent1550 Donald W. Riegle Jr. Work_For Senate Banking Committee 3. sent1993 Abebe Worke Work_For High Court 4. sent2197 Louis Sullivan Work_For HHS 5. sent2536 Harrington Work_For Socialist International	No prediction	Correct Flair's tagging and chunking + not complicated sentence structure.
6	Other	1. sent1246 Sanders Work_For CBOT 2. sent1246 Daniel Dewey Work_For CBOT 3. sent1375 Larry Jinks Work_For Knight-Ridder 4. sent1375 William A. Ott Work_For Knight-Ridder 5. sent2055 Booth Gardner Work_For National Governors ' Association 6. sent2055 Branstad Work_For National Governors ' Association 7. sent2203 Charles McC. Work_For Congress 8. sent2208 Anthony M. Kennedy Work_For Supreme Court 9. sent2217 Strom Thurmond Work_For Congress 10. sent2097 John Johnson Work_For XBT Telecom	No prediction	1. complicated sentence structure. 2. complicated sentence structure. 3. In my opinion, the relation of Larry Jinks to Knight-Ridder org' is not clearly a Work_For relation. 4. In my opinion, the relation of William A. Ott to Knight-Ridder org' is not clearly a Work_For relation. 5. Unclear wording of the sentence - "…Democrat Booth Gardner of Washington was selected to serve as vice to follow Branstad as chairman in a year." – is it for this year? For next year? Not clear. Therefore the Work_For relation is in question here. 6. There are two mentions of the person in the expected gold relation in the sentence, once in the beginning of the sentence where it appears as 'Terry Branstad' from which it can be associated to working in National Governors ' Association, but the second mention of him 'Branstad'(the one expected in the relation) cannot be associated with National Governors ' Association. 7. Charles McC worked in the past in the Congress, but not in the present time of the sentence, therefore my model is correct for not predicting it. 8. complicated sentence structure. 9.. It's hard to connect between Strom Thurmond to the Congress. 10. original sentence: "Of 40 million party line calls logged by New England Telephone over a 2-LCB--year period , at least 10 percent were

				made by those who dial services like XBT Telecom 's ` ` Talkabout ' ' teen line , a group conversation designed for people under age 16 , said John Johnson , a spokesman for the telephone company. " As I understand it, the telephone company is a reference to the New England Telephone. So, my model was right for not predicting this relation.

Precision errors:

#	Error type	Gold	Prediction	Analysis
1	Training procedure	-	1. sent1313 Dietmar Kanzer Work_For AP 2. sent1332 Peter Butler Work_For 5th U.S. Circuit Court of Appeals 3. sent1857 Robert Bernero Work_For DOE 4. sent2013 Bernero Work_For Energy Department 5. sent2056 Dan Rather Work_For National Governors ' Association 6. sent2195 Mason Work_For O and Drug Administration 7. sent2649 Dean Falk Work_For Science News	1. The person is not working in this organization; he works for the other organization in the sentence (Rollei) and the relation between them was already predicted. 2. The person is not working in this organization; he works for the other organization in the sentence (A. Copeland Enterprises Inc.) and the relation between them was already predicted. 3. The person is not working in this organization; he works for the other organization in the sentence (NRC) and the relation between them was already predicted. 4. The person is not working in this organization – it's not clear if true or not. 5. The person is not working in this organization; he works for the other organization in the sentence (CBS) and the relation between them was already predicted. 6. The person is not working in this organization; he works for the other organization in the sentence (Public Health Service) and the relation between them was already predicted. 7. The person is not working in this organization; he works for the other organization in the sentence (State University of New York) and the relation between them was already predicted.
2	Missed relation in the annotations file	-	1. sent1702 Leonard Lee Work_For Chinese Times 2. sent2013 Sam Rousso Work_For Energy Department 3. sent2336 Bob Dole Work_For Senate 4. sent3073 Mickey Kantor Work_For USTR	1. Leonard Lee indeed works for Chinese Times *original sentence: "…said Leonard Lee , editor of the Chinese Times" so my model was right. 2. Sam Rousso works in the Energy Department of NRC. *original sentence: "…in a letter to Sam Rousso , acting of civilian radioactive waste management for the Energy Department " 3. As I understand it, Bob Dole is working at the U.S. Senate (not quite sure about it) *original sentence: "Senate Bob Dole of Kansas on Tuesday threatened…" 4. Mickey Kantor works at the USTR (Office of the United States Trade Representative) *original sentence: "…U.S. (USTR) Mickey Kantor"
3	Duplicate prediction	-	1. sent1859 Bernero Work_For NRC	1. it's because the NRC org appears twice in the sentence, so the relation was identified once with the mention of it at the beginning

				of the sentence and in the second time with the mention of it at the end of the sentence. In my opinion both are correct.
4	Mis-tagging and bad chunking of flair	-	1. sent2403 Tom Muri. Work_For Whitefish	1. flair identified the entity – Whitefish as an organization while it's actually a city, so the entity type should be LOC. In addition, there is a live_in relation between Tom Muri to 'Whitefish City', which also indicates on bad chunking of flair.
5	Mistake in the annotations file	1. John Johnson Work_For XBT Telecom	1. sent2097 John Johnson Work_For New England Telephone	1. original sentence: "Of 40 million party line calls logged by New England Telephone over a 2-LCB--year period , at least 10 percent were made by those who dial services like XBT Telecom 's `` Talkabout ' ' teen line , a group conversation designed for people under age 16 , said John Johnson , a spokesman for the telephone company. As I understand it, the telephone company is a reference to the New England Telephone.

Evaluation results

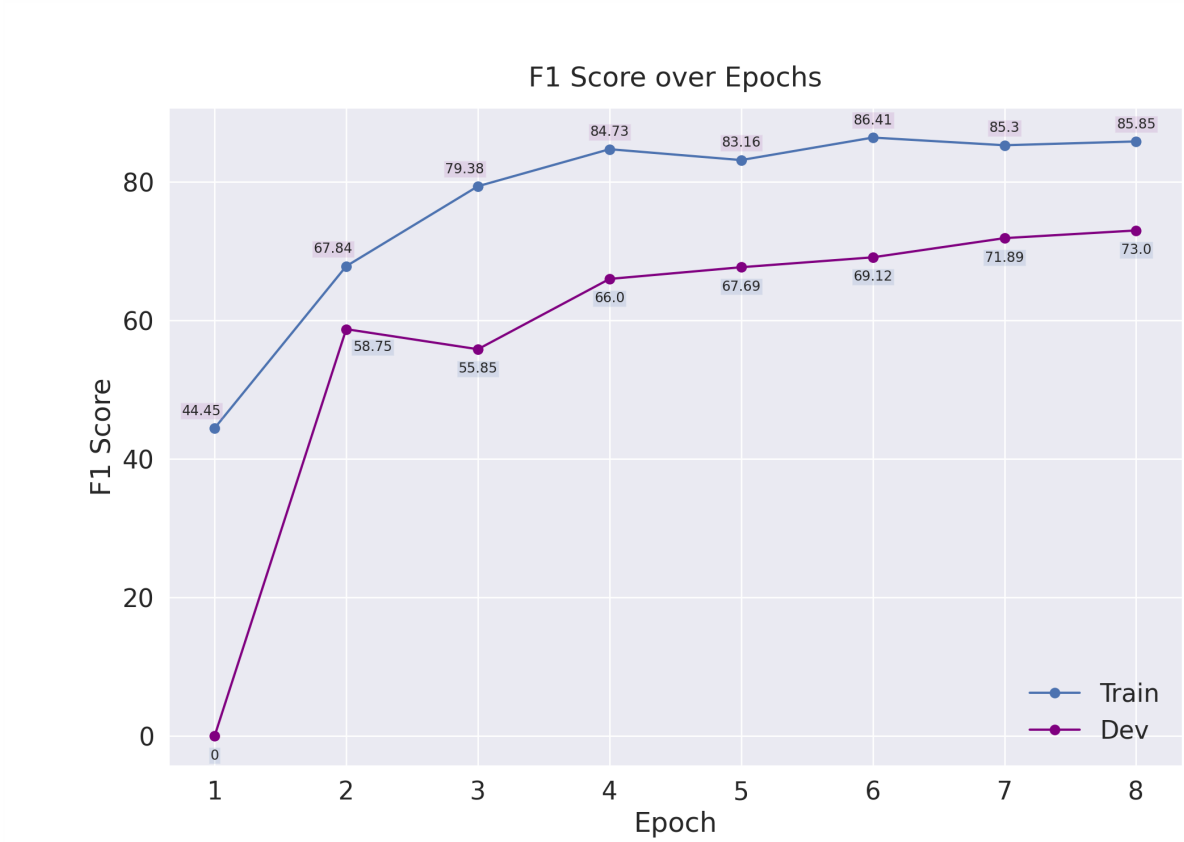
Original Scores

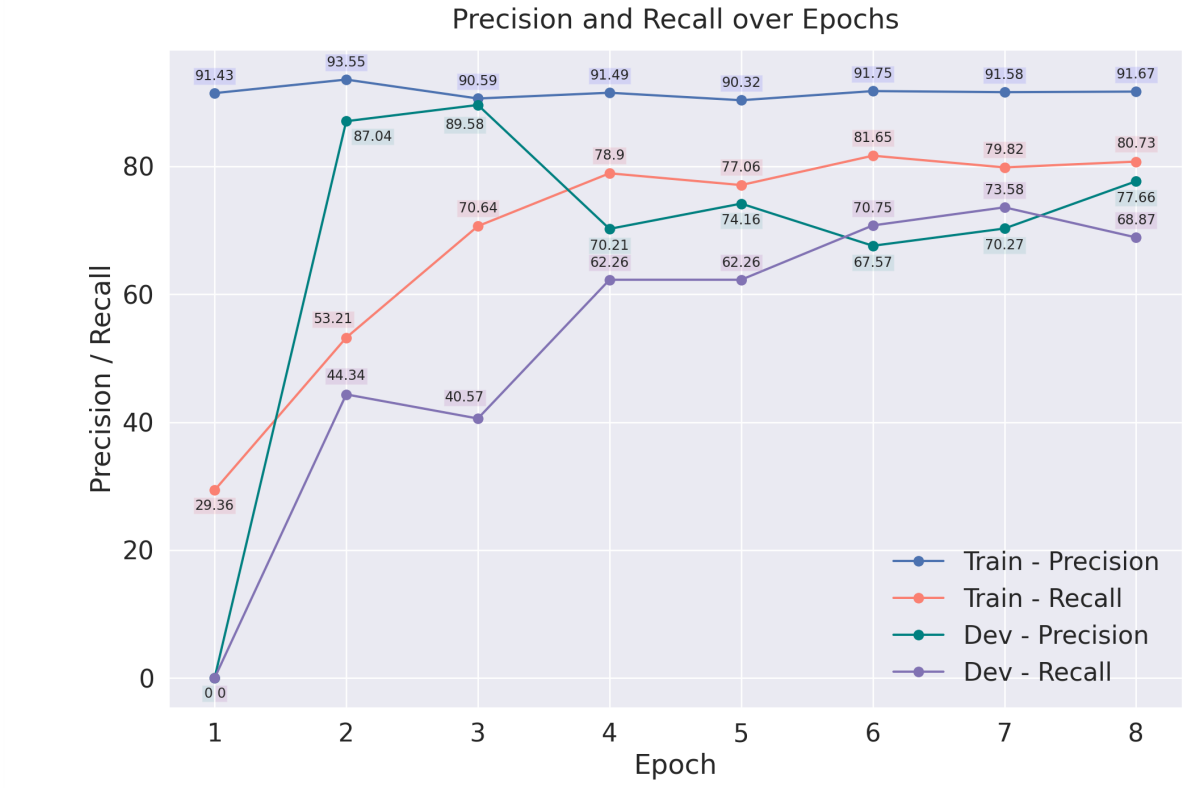
Relation	Train Recall	Train Prec	Train F1	Dev Recall	Dev Prec	Dev F1
Work_For	80.73%	91.67%	85.85%	68.87%	77.66%	73.0%

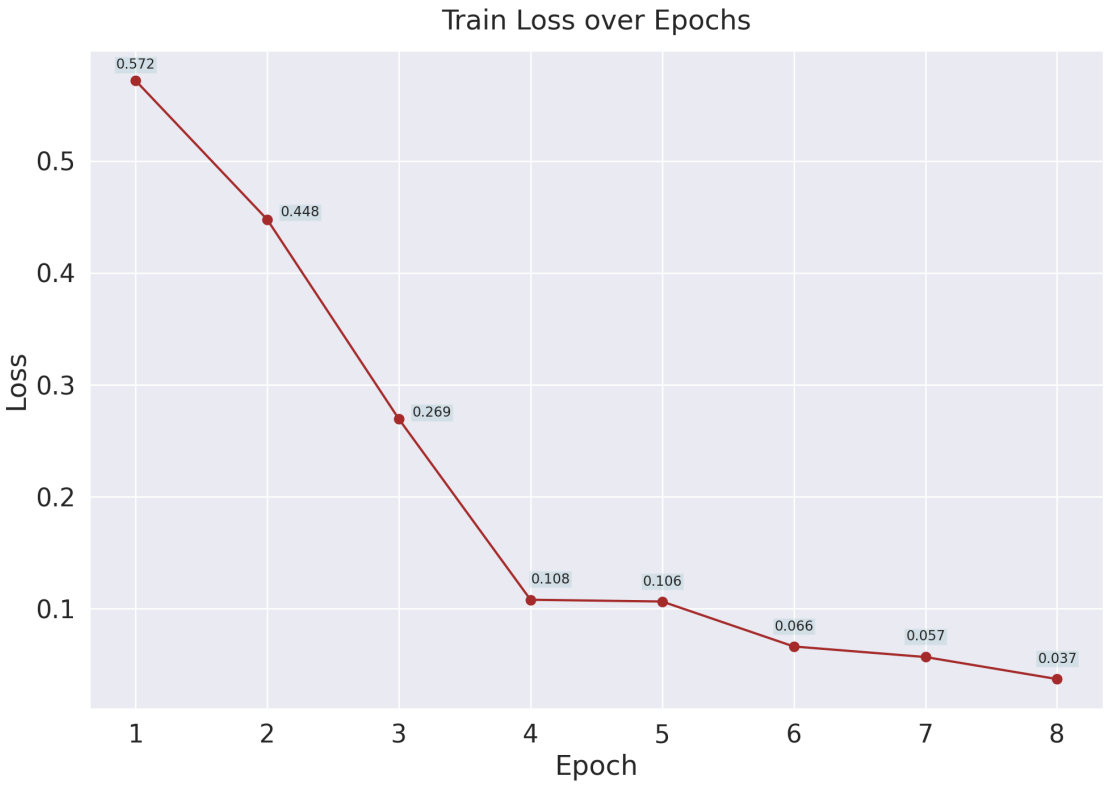
Generalized Scores

Relation	Train Recall	Train Prec	Train F1	Dev Recall	Dev Prec	Dev F1
Work_For	83.49%	94.79%	88.78%	75.47%	85.11%	80.0%

Graphs


F1 Score over Epochs
F1 Score
Epoch
Train
Dev
44.45
67.84
79.38
84.73
83.16
86.41
85.3
85.85
0
58.75
55.85
66.0
67.69
69.12
71.89
73.0


Precision and Recall over Epochs
Precision / Recall
Epoch
Train - Precision
Train - Recall
Dev - Precision
Dev - Recall
91.43
93.55
90.59
91.49
90.32
91.75
91.58
91.67
87.04
89.58
70.21
74.16
67.57
70.27
77.66
68.87
29.36
53.21
70.64
78.9
77.06
81.65
79.82
80.73
0 0
44.34
40.57
62.26
62.26
70.75
73.58


Train Loss over Epochs
Loss
Epoch
0.572
0.448
0.269
0.108
0.106
0.066
0.057
0.037
0.5
0.4
0.3
0.2
0.1
1
2
3
4
5
6
7
8