Semantic Role Labeling

The aim of this homework was to tackle the problem of semantic role labeling (SRL) which consists in assigning labels and roles to words in a sentence according to their semantic relation with the predicate of the sentence. As it is informally described, it is the task of describing *who* did *what* to *whom*, *when*, *where*, *how*, *why...*

We want to be able to automatically (1) discover the arguments of a predicate and (2) to assign roles to the arguments. For instance given the sentence "*he climbed a hill*" and the predicate *climbed*, a SRL system would discover the arguments of the predicate, which in this case could be: *he* and *hill*, and would assign roles to these arguments, which in this case could be: *he* $\rightarrow$ agent and *hill* $\rightarrow$ patient.

In order to treat this problem I decided to use the typical approach which is based on supervised machine learning. Specifically, I was inspired by the solution proposed by Diego Marcheggiani, Anton Frolov and Ivan Titov in the suggest publication about a SRL model. I made several experiments and modifications to that model and I will discuss about it in this report.

1 Neural Models for Semantic Role Labeling

The neural models are based on the supervised learning paradigm. A neural network is fed of training examples from the training set, by means of which a model learns how to classify unseen instances from the development or test set.

The dataset on which I trained, evaluated and tested my neural models is the *CoNLL 2009 - English.* Sentences in the dataset are lists of words which keep informations like: lemmas, POS tags, the semantic dependencies, if they are predicates, if they are arguments and the relative semantic roles. So given a training example, a sequence of words, my neural model was trained to predict if a given word is an argument and what is its role.

The models were evaluated in the following way, a predicted role is considered a:

- *true positive* (`TP`) if it is equal to the ground truth and it is not null (null = input word is not an argument). Correctly predicted the role of an argument. E.g. *prediction* $\rightarrow$ `A0` : *truth* $\rightarrow$ `A0`.
- *false positive* (`FP`) if it is different from the ground truth and it is not null. Wrongly predicted the role of an argument or predicted a role for a non-argument. E.g. *prediction* $\rightarrow$ `A0` : *truth* $\rightarrow$ `A1`; *prediction* $\rightarrow$ `A0` : *truth* $\rightarrow$ `null`.
- *false negative* (`FN`) if it is different from the ground truth and it is null. Predicted that an argument is a non-argument. E.g. *prediction* $\rightarrow$ `null` : *truth* $\rightarrow$ `A1`.

Having such information allowed me to compute useful metrics like the precision: $p = \texttt{TP}/(\texttt{TP} + \texttt{FP})$, the recall: $r = \texttt{TP}/(\texttt{TP} + \texttt{FN})$ and the F1 measure: $2 \cdot p \cdot r/(p + r)$. The goodness of one model was chosen according to the highest value of the F1 measure during a development evaluation.

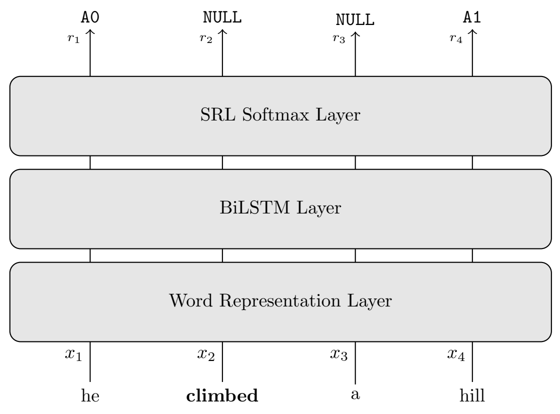
1.1 Model 1: Basic neural model

Model 1 is composed of three layers, as shown in figure 1a. The layers are the following:

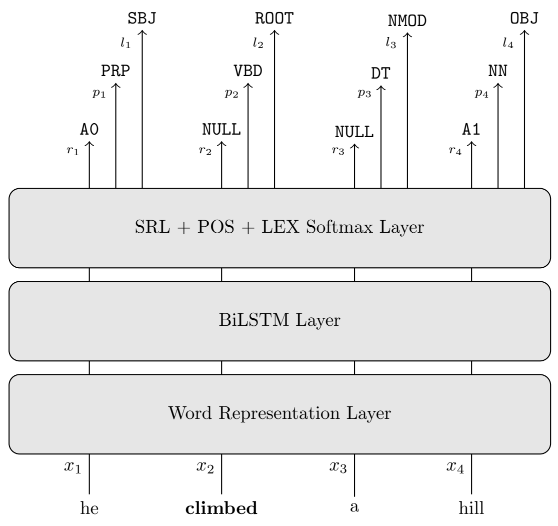
Layer 1 - Word Representation Layer It is responsible for creating a proper word representation for each word in the sentence. Given an input sentence, for each word x, a word representation generator $g(\cdot)$ produces a word representation $g(x)$ which identifies x. The goal for such a generator is to produce representations that are expressive enough to map words that have similar semantic roles to a similar representation.

Building a good $g(\cdot)$ function is crucial for performance. So I did several attempts, (1) I started with a very simple $g(\cdot)$ function which given a word as input returns the pre-trained embedding of the lemma x_{lemb}. This simple representation lacks of syntactic and morphological information. With lemma embeddings only, we are not able to distinguish the word "*dress*" in the following sentence "*Did you see her dress?*", it may either be a verb or a noun. To be more expressive (2) I added the one hot encoding of the part of speech tag of the word x_{pos}. Now ambiguous words like the one seen above, will have different representations. (3) I added the one hot encoding of the semantic dependency relation x_{lex}.

In addition, I also added a bit $p \in \{0, 1\}$ to the representation indicating if the word is the predicate of the sentence or not. Eventually the best $g(\cdot)$ turned out to be the one producing the following word representation: $wr_x = g(x) = x_{lemb} \circ x_{lex} \circ x_{pos} \circ p$.



(a) Model 1, Basic Model



(b) Model 2, Multitask Learning Model

Figure 1: The structure of the first two neural models. Three stacked layers produce words representations, words latent representations affected by theirs context, predictions of semantic roles.

Layer 2 - BiLSTM Layer It is responsible for modeling sequences of words. For each word representation wr coming from layer 1, it returns an encoding of the word that's affected by its context. In order to produce such an encoding, two layers (or more) of LSTM networks are stacked to make a forward pass and a backward pass over the sequence, the concatenation of the outputs coming from the two networks, gives the actual latent representation h_x of the word.

Furthermore this layer is responsible for concatenating to h_x a bit $p \in \{0, 1\}$ indicating if the word is the predicate of the sentence or not; it also concatenates the hidden representation h_y of the predicate of the sentence, together with its p indicator. So the output of this layer is the concatenation $cr_x = h_x \circ p \circ h_y \circ 1$.

Layer 3 - SRL Softmax Layer It is responsible for predicting semantic roles. By labeling each word in the input sentence with its role, including the `NULL` role for the words that are not arguments of the predicate, we can train a multi layer perceptron to predict the correct semantic role. The input of the network is cr_x coming from the underlying layer. The network computes the probability that this concatenation has the semantic role r for all r in the set of roles, $p(r|cr_x = h_x \circ p \circ h_y \circ 1)$. A softmax layer produces this probability distribution that is then used to predict the actual role of the word.

1.2 Model 2: Multitask learning neural model

Model 2 is composed of three layers, as shown in figure 1b. The layers are the following:

Layer 1 - Word Representation Layer It is responsible for creating a proper word representation for each word in the sentence. For each word it outputs the representation $wr_x = g(x) = x_{lemb} \circ x_{lex} \circ x_{pos} \circ p$. It is analogue to layer 1 of model 1 described earlier, for detailed information look above.

Layer 2 - BiLSTM Layer It is responsible for modeling sequences of words. From a sequence of word representations wr_x it outputs the hidden representation of such words $cr_x = h_x \circ p \circ h_y \circ 1$. It is analogue to layer 2 of model 1 described earlier, for detailed information look above.

Layer 3 - SRL + POS + LEX Softmax Layer It is responsible for predicting semantic roles. It is a multitask learning neural network, we try to improve the performance of our learning algorithm by leveraging training signals contained in related tasks so to exploit their commonalities and differences. What we do here is to train an architecture using multiple loss functions and one shared representation, I added one output layer for each additional task keeping the innermost hidden layer common across all tasks. I thought it could be a good idea to work in parallel on the following three tasks since they are all very much related to each other: semantic role labeling (SRL), part of speech tagging (POS) and lexical semantic tagging (LEX).

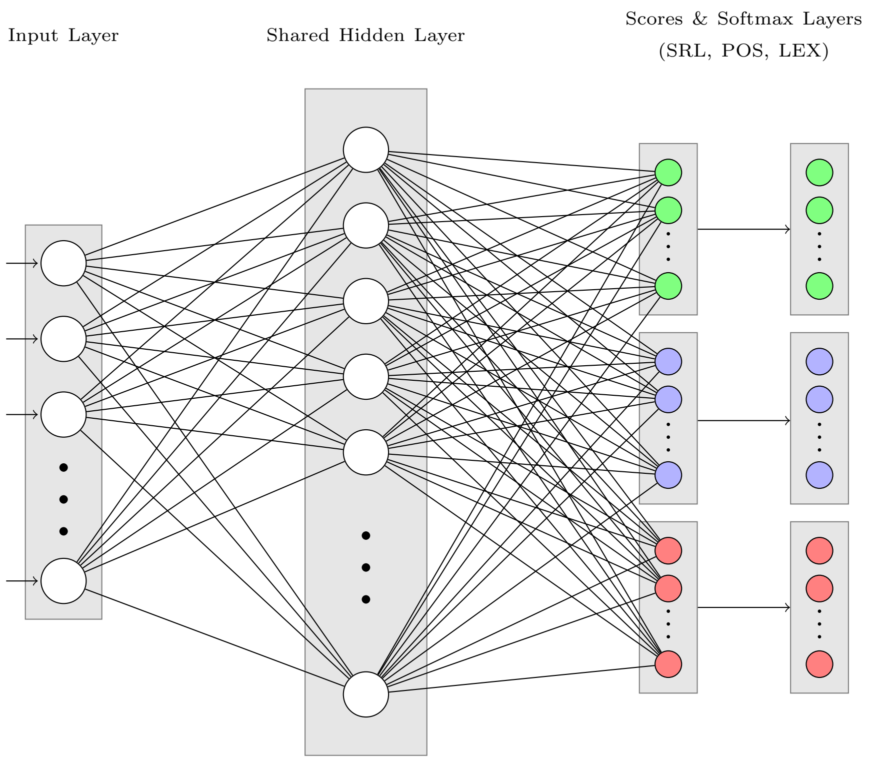


Figure 2: Layer 3 of model 2: SRL + POS + LEX Softmax It is responsible for predicting semantic roles. Given batches of input sentences, for each word it produces the probability distributions for semantic roles, for part of speech tags and for lexical dependencies. The goodness of the predictions influences updates either in the hidden layer shared among tasks or on the local hidden layers of each task.

As soon as a batch of sequences of hidden representations cr_x arrives to layer 3, together with its labels (semantic role, pos tag, lexical tag), the model makes its guesses. The guesses are provided by means of three softmax layers which return the most likely semantic role (green in the figure above), post tag (violet in the figure above) and lexical tag (red in the figure above). For each sequence in the batch, for each hidden representation, depending on the goodness of the three guesses the model gets updated accordingly. Specifically the loss function of the network $loss_{net}$ is the sum of the three loss functions $loss_{net} = loss_{srl} + loss_{pos} + loss_{lex}$ which is back-propagated in the network to update the weights in the shared hidden layer and in the isolated hidden layers.

1.3 Model 3: Basic neural model with semantic integration

This model is a extension of model 1. In addition to model 1, here we have another subsystem, the disambiguation system, which is plugged in the word representation layer. So model 3 is still composed of three layers, as shown in figure 1a. The layers are the following:

Layer 1 - Word Representation Layer It is responsible for creating a proper word representation for each word in the sentence. In addition to the word representation expressed in the previous two models, here we have another component concatenated to wr_x, which is x_{sen}. This component is the sense embedding of the word. In order to retrieve it, the whole dataset gets disambiguated (using the system built for homework 2). After the disambiguation, ambiguous words will have their BabelnetSynset associated, through which we can access the sense embedding x_{sen}.

Layer 2 - BiLSTM Layer It is responsible for modeling sequences of words. From a sequence of word representations wr_x it outputs the hidden representation of such words $cr_x = h_x \circ p \circ h_y \circ 1$. It is analogue to layer 2 of model 1 described earlier, for detailed information look above.

Layer 3 - SRL + POS + LEX Softmax Layer It is responsible for predicting semantic roles. By labeling each word in the input sentence with its role, with a multilayer perceptron. The input of the classifier is the context representation cr_x coming from layer 2.

An alternative attempt was to concatenate the semantic embedding x_{sen} of a word to its context representation: $cr_x = h_x \circ p \circ h_y \circ 1 \circ x_{sen}$, but not to the word representation, and then to pass this concatenation to the classifier. This version showed to perform best with respect to the original attempt.

2 Experimental analysis and results

I made several executions of the three models and I evaluated the goodness of the three systems on the development set provided. The *CoNLL 2009 - English* development split: has 1334 sentences, 33368 total words, 6390 predicates. Considering that each sentence is fed in the neural architecture np_i times during one epoch, where np_i is the number of predicates within sentence i, the number of instances to classify is 195029 of which 181164 are not arguments while the remaining 13865 are arguments. So in order to achieve 100% F1 it would be needed to correctly predict that those 13865 are indeed arguments and their role.

For each model, a detailed list of tables relative to the evaluation is proposed in section 4. Tables starting with letter:

- **A** are the hyperparameter optimization grids, which list several tests and the relative outcomes. A grid search approach was made by searching through the possible values that hyperparameters may assume, values which led to an increase in F1 were chosen for the subsequent tests.
- **B** show the variation in precision, recall and F1 with the passing of the epochs, in order to see how the learning process changes through time. Generally during an execution, the best model was chosen among those resulting in the highest precision at the end of the epoch.
- **C** are the confusion matrices which list the number of `TP`, `TN`, `FP`, `FN` for each class (the roles). They should be read by keeping the truth on left and the prediction above. On the green diagonal there are the `TP`, on the violet diagonal there are the `TN`. The sum of the elements on the rows, excluded the `TP` quantity of the row, is the number of `FP` of a role. The sum of the elements of the columns, excluded the `TP` quantity of the column, is the number of `FN` of the role.
- **D** plot the precision, recall and F1 for each of the classes, in order to evaluate how good a model predicts a certain role.

After executing and testing the models the final outcome was pretty satisfying. The best scores were met with model 1, which indeed is the most simple implementation. In particular I obtained the results: **84,8%** of F1, **86,3%** of Recall, **83,4%** and Precision (see `Test 1.14` in Table A1). I found this model good enough to annotate the test set with.

3 Conclusion

This work was a good exercise to experiment how effective it is to perform semantic role labeling using properly tuned recurrent neural networks and classifiers. In this report we could see several implementations of neural models for semantic role labeling.

Model 1 showed how important it is to build a proper word representation, in order to obtain an effective context representation that may lead to accurate classifications. Namely, by concatenating to the lemma embedding, the pos tag and the lex tag encoding, we obtained up to +8% in F1. Also stacking multiple LSTM layers (more than 2) led to better results.

Model 2 was the most particular model structurally, the multitask learning classifier was focused on three tasks: predicting semantic roles, pos tags, and lex tags. Surprisingly the performance of this model were just as competitive as those of model 1, despite the higher complexity of the classifier. A deeper search through the possible hyperparameters values (e.g. the optimizer and the shared hidden layer size) may reveal the true power or weakness of this classifier.

Model 3 was the result of two big subsystems, the world sense disambiguation system (WSD) built for the homework 2, and the semantic role labeling system (SRL). Semantic integration to the SRL system should have produced better scores, but in my tests this wasn't so. Mainly due to the fact that the WSD system of homework 2 wasn't a top quality disambiguator, the noise produced led to results that are as good as those of the previous models.

One could decide to expand all this, for example by: adding more than just 4 LSTM layers to produce an even more significant hidden representations; adding more significant features to the word representation, for example an accurate sense embedding, by exploiting a valuable word sense disambiguator; accurately tuning hyperparameters.

4 Tables of the experiments

4.1 Tables of Model 1

	Pre-trained embeddings	Batch size	LSTM Hidden size	Drop-out bit	Epochs	LSTM Layers	x_{lemb}	x_{pos}	x_{lex}	Initial avg loss	Final avg loss	Initial Precision (%)	Initial Recall (%)	Initial F1 score (%)	Final `TP` (#)	Final `FP` (#)	Final `FN` (#)	Final Precision (%)	Final Recall (%)	**Final F1 score (%)**
`Test 1.1`	`GOOGLE`	5	100	0.0	5	2	●	○	○	0.28	0.11	60	39	48	8593	3269	4085	72	68	70
`Test 1.2`	`GLOVE`	-	-	-	-	-	●	○	○	0.37	0.14	49	28	36	7672	3284	4575	70	62	66
`Test 1.3`	`FTEXT` ♠	-	-	-	-	-	●	○	○	0.32	0.11	57	43	49	8953	2993	3460	75	72	74
`Test 1.4`	-	50	-	-	-	-	●	○	○	0.21	0.15	61	42	50	7348	3606	4778	67	60	64
`Test 1.5`	-	100	-	-	-	-	●	○	○	0.21	0.16	56	47	50	5654	2928	6445	66	46	55
`Test 1.6`	-	5 ♠	200	-	-	-	●	○	○	0.75	0.12	53	46	49	9165	4177	3046	69	75	72
`Test 1.7`	-	-	512	-	-	-	●	○	○	0.76	0.16	50	14	21	6820	3747	5134	65	57	61
`Test 1.8`	-	-	50 ♠	-	-	-	●	○	○	0.25	0.09	60	51	55	9370	2837	3211	77	74	76
`Test 1.9`	-	-	-	-	-	-	●	●	○	0.22	0.08	69	63	66	10133	2818	2399	78	81	80
`Test 1.10`	-	-	-	-	-	-	●	○	●	0.20	0.08	70	64	67	10833	2871	2999	83	82	82
`Test 1.11`	-	-	-	-	-	-	●	●	●	0.19	0.06	72	64	67	10599	2307	1929	82	85	84
`Test 1.12`	-	-	-	0.5	-	-	●	●	●	0.22	0.09	70	49	57	9629	2579	2633	78	80	79
`Test 1.13`	-	-	-	0.0 ♠	15 ♠	-	●	●	●	0.19	0.04	73	62	67	10740	2184	1896	83	85	84
`Test 1.14`	-	-	-	-	-	4 ♠	●	●	●	0.20	0.4	77	77	77	10959	2276	1711	83	86	85

Table A1: Execution of neural Model 1. Hyperparameter tuning via grid search approach. Each test has been performed with a *Gradient Descend* optimizer with learning rate set to 2. Increasings in the batch size and in the size of the hidden representation of the LSTM layers, wasn't a good choice because of the decrease in F1 they produced during testing. Pre-trained word embeddings families showed to be a significant hyperparameter, choosing FastText against Glove embeddings produced a +8% of F1 score (`T1.2` - `T1.3`). Another good choice was to build different word representations. Just by concatenating x_{lex}, the lexical information, to x_{lemb} an increase of +6% in F1 was obtained (`T1.8` - `T1.10`). Concatenating even the POS tag encoding x_{pos} produced an increase of +2% in F1 with respect to the previous test (`T1.10` - `T1.11`). Using four stacked LSTM layers resulted in the best model (`T1.14`).

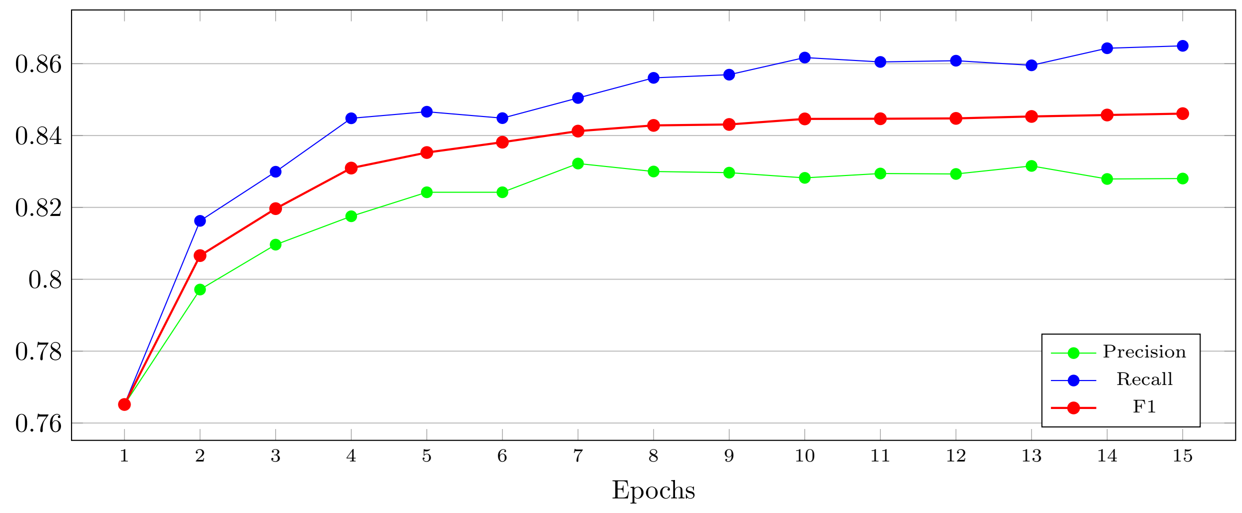


Table B1: Execution of neural Model 1. The variation in precision, recall and F1 metrics with the passing of the epochs. The metrics look interesting in some epochs. From epoch 8 onward, the recall tends to increase while the precision slowly drops. Models with high recall and low precision, are very good at avoiding labeling actual arguments as *non-argument* (few false negatives) but indeed they are not precise in labeling the role of the argument (many false positives). So what model to chose depends mainly on the task that one wants to solve.

T |– P	A0	A1	A2	A3	A4	A5	AA	AM-ADV	AM-CAU	AM-DIR	AM-DIS	AM-EXT	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-PNC	AM-PRD	AM-TMP	C-A1	C-A2	C-AM-CAU	C-AM-DIR	C-AM-MNR	R-A0	R-A1	R-A2	R-AM-CAU	R-AM-EXT	R-AM-LOC	R-AM-MNR	R-AM-TMP	NULL
A0	2897	98	38	7	1	0	0	0	0	0	0	0	5	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	397
A1	114	4378	105	23	2	0	0	1	0	1	0	2	32	6	0	0	5	0	8	4	0	0	0	0	0	2	0	0	0	0	0	0	539
A2	55	117	1101	16	3	0	0	7	0	9	1	1	40	6	0	0	4	0	7	2	0	0	0	0	0	1	0	0	0	0	0	0	225
A3	15	20	26	168	0	0	0	3	1	1	0	0	1	5	0	0	2	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	57
A4	1	3	12	2	45	0	0	2	0	1	0	1	3	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
A5	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AA	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-ADV	0	3	5	2	0	0	0	177	1	0	13	1	0	12	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	60
AM-CAU	0	1	1	0	0	0	0	4	29	0	0	0	0	0	0	0	4	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	5
AM-DIR	0	2	4	0	1	0	0	0	0	20	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3
AM-DIS	0	0	1	0	0	0	0	14	0	0	146	0	1	1	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	34
AM-EXT	0	0	3	0	0	0	0	1	0	0	0	30	0	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	8
AM-LOC	10	13	19	0	0	0	0	5	0	0	0	0	217	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	75
AM-MNR	2	20	27	6	1	0	0	21	0	1	3	5	11	247	0	0	2	0	4	3	0	0	0	0	0	0	0	0	1	0	0	0	75
AM-MOD	0	0	1	0	0	0	0	0	0	0	0	0	0	0	302	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
AM-NEG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	111	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	9
AM-PNC	0	3	7	3	0	0	0	7	2	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
AM-PRD	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-TMP	1	7	2	2	0	0	0	7	0	0	3	0	3	4	0	1	0	0	707	0	0	0	0	0	0	0	0	0	0	0	0	0	141
C-A1	1	9	8	0	0	0	0	2	0	0	0	0	2	0	0	0	1	0	0	105	0	0	0	0	0	0	0	0	0	0	0	0	11
C-A2	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-CAU	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-DIR	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-MNR	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-A0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	131	4	0	0	0	0	0	0	7
R-A1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	0	0	0	7
R-A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	0	0	0	0	0	1
R-AM-CAU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2
R-AM-EXT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R-AM-LOC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2	0	0	6
R-AM-MNR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	2
R-AM-TMP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	15
NULL	252	287	147	28	2	0	0	56	4	2	22	4	70	63	12	3	5	0	97	4	0	0	0	0	7	7	0	1	0	0	0	8	180083

Table C1: Confusion Matrix of Model 1.

	A0	A1	A2	A3	A4	A5	AA	AM-ADV	AM-CAU	AM-DIR	AM-DIS	AM-EXT	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-PNC	AM-PRD	AM-TMP	C-A1	C-A2	C-AM-CAU	C-AM-DIR	C-AM-MNR	R-A0	R-A1	R-A2	R-AM-CAU	R-AM-EXT	R-AM-LOC	R-AM-MNR	R-AM-TMP	NULL
TP	2897	4378	1101	168	45	0	0	177	29	20	146	30	217	247	302	111	50	0	707	105	0	0	0	0	131	73	2	1	0	2	4	16	180083
FP	454	586	408	89	10	0	0	136	9	16	42	14	171	104	13	4	26	0	145	15	0	0	0	0	8	15	1	1	0	0	1	8	1711
FN	549	844	494	134	39	3	1	110	17	14	56	17	124	182	13	12	30	3	171	34	3	1	1	2	13	9	3	2	1	7	2	15	1081
Precision(%)	86	88	73	65	82	0	0	57	76	56	78	68	56	70	96	97	66	0	83	88	0	0	0	0	94	83	67	50	0	100	80	67	99
Recall(%)	84	84	69	56	54	0	0	62	63	59	72	64	64	58	96	90	62	0	81	76	0	0	0	0	91	89	40	33	0	22	67	52	99
F1(%)	85	86	71	60	65	0	0	59	69	57	75	66	60	63	96	93	64	0	82	82	0	0	0	0	92	86	50	40	0	36	73	59	99

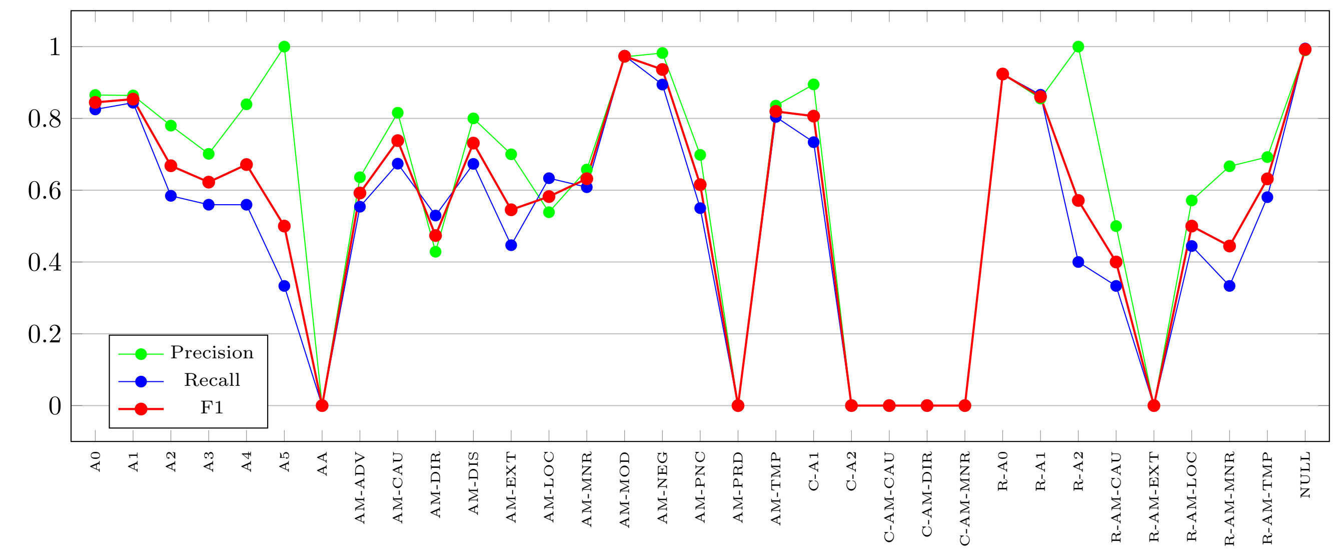


Table D1: Confusion Matrix of Model 1. The variation in precision, recall and F1 metrics for each role. Notice how some predictions have high scores having many instances (A0, A1, AM-MOD, AM-TMP ...). From role A0 to A5 the scores tend to decrease, since they progressively appear less frequently and furthermore the way in which they get overloaded may influence.

4.2 Tables of Model 2

	Pre-trained embeddings	Batch size	LSTM hidden size	Shared hidden layer size	Drop-out bit	Epochs	LSTM Layers	x_{lemb}	x_{pos}	x_{lex}	Initial Precision (%)	Initial Recall (%)	Initial F1 score (%)	Final TP (#)	Final FP (#)	Final FN (#)	Final Precision (%)	Final Recall (%)	**Final F1 score (%)**
`Test 2.1`	`FTEXT`	5	100	300	0.0	15	2	●	●	●	77	76	77	10777	2379	1780	82	86	84
`Test 2.2`	-	-	-	900	-	-	-	●	●	●	76	72	74	10742	2288	1855	82	85	84

Table A2: Execution of neural Model 2. Hyperparameter tuning via grid search approach. Each test has been performed with a *Gradient Descend* optimizer with learning rate set to 0.2. The setting of the hyperparameters of the tests `T2.1` and `T2.2` was the same as the best execution of model 1, `T1.14`. These two tests were focused on tuning the hyperparameter introduced in the model, which is the size of the shared hidden layer. Increasing the size from 300 to 900 didn't lead to any better result. And since we want to keep the system simple and efficient, I preferred the model of test `T2.1`.

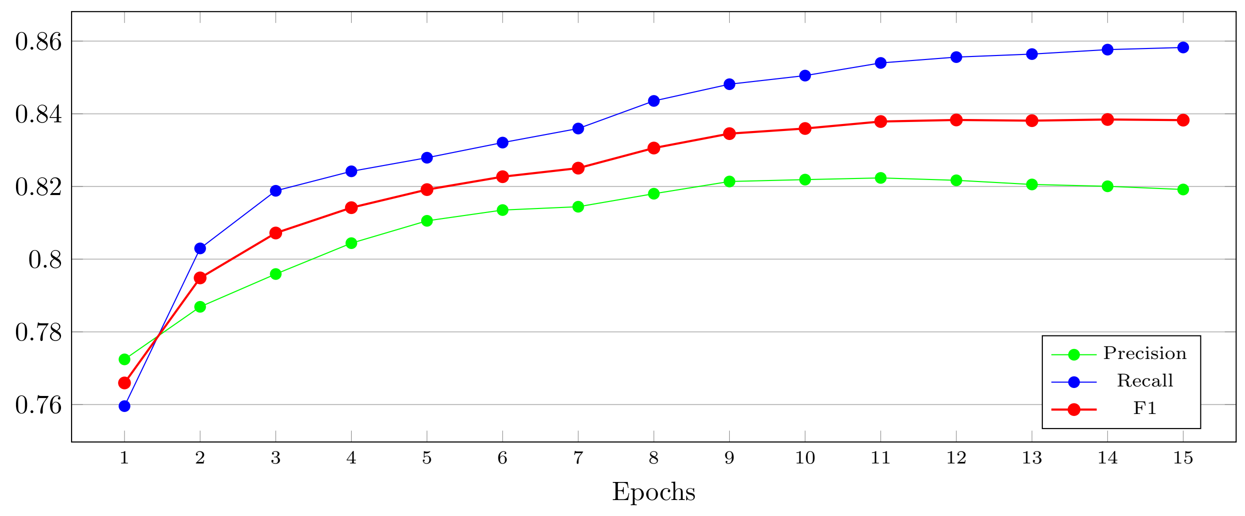


Table B2: Execution of neural Model 2. The variation in precision, recall and F1 metrics with the passing of the epochs. Here we can appreciate a progressive growth, no ups and downs, the model seems to learn pretty smoothly until convergence. The more the training goes on, the more recall and precision diverge, the first increases the latter decreases.

T |– P	A0	A1	A2	A3	A4	A5	AA	AM-ADV	AM-CAU	AM-DIR	AM-DIS	AM-EXT	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-PNC
A0	2870	113	39	9	0	0	0	0	1	0	0	0	5	1	0	0	0
A1	143	4310	113	17	3	0	0	4	0	2	0	0	31	9	0	0	2
A2	60	156	1052	14	8	0	0	3	0	5	1	6	28	14	0	0	2
A3	15	22	41	156	0	0	0	2	0	2	0	0	5	5	0	0	4
A4	0	6	10	2	47	0	0	0	0	2	0	0	2	0	0	0	4
A5	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
AA	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-ADV	0	1	5	1	0	0	0	158	2	0	16	0	0	17	0	1	0
AM-CAU	0	0	2	0	0	0	0	4	29	0	0	0	0	0	0	0	2
AM-DIR	0	2	6	0	0	0	0	0	0	15	0	0	1	0	0	0	0
AM-DIS	0	0	1	0	0	0	0	6	0	0	148	0	1	1	0	0	0
AM-EXT	0	2	2	0	0	0	0	2	0	0	0	23	1	6	0	0	0
AM-LOC	14	10	21	3	1	0	0	3	0	0	0	0	198	0	0	0	0
AM-MNR	3	20	30	6	3	0	0	16	0	0	4	6	5	255	0	1	2
AM-MOD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	308	0	0
AM-NEG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	114	0
AM-PNC	0	8	0	4	1	0	0	4	3	0	0	0	0	0	0	0	45
AM-PRD	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
AM-TMP	3	10	2	1	0	0	0	4	0	0	5	0	3	4	0	1	0
C-A1	0	9	6	1	0	0	0	1	1	0	0	0	2	0	0	0	0
C-A2	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
C-AM-CAU	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
C-AM-DIR	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C-AM-MNR	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
R-A0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-A1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-A2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-AM-CAU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-AM-EXT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-AM-LOC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-AM-MNR	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-AM-TMP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-AM-PNC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-A4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NULL	251	295	149	31	3	0	0	38	6	4	24	3	47	55	12	6	6

T |– P	AM-PRD	AM-TMP	C-A1	C-A2	C-AM-CAU	C-AM-DIR	C-AM-MNR	R-A0	R-A1	R-A2	R-AM-CAU	R-AM-EXT	R-AM-LOC	R-AM-MNR	R-AM-TMP	R-AM-PNC*	R-A4*	NULL
A0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	407
A1	0	5	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	578
A2	0	7	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	235
A3	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	47
A4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
A5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-ADV	0	12	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72
AM-CAU	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
AM-DIR	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
AM-DIS	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39
AM-EXT	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
AM-LOC	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88
AM-MNR	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	73
AM-MOD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
AM-NEG	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
AM-PNC	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
AM-PRD	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-TMP	0	719	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	126
C-A1	0	0	105	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14
C-A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
C-AM-CAU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-DIR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-MNR	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-A0	0	0	0	0	0	0	0	133	3	1	0	0	0	0	0	0	0	4
R-A1	0	0	0	0	0	0	0	2	70	1	0	0	0	0	0	0	0	5
R-A2	0	0	0	0	0	0	0	1	1	2	0	0	0	0	0	0	0	0
R-AM-CAU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
R-AM-EXT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R-AM-LOC	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	7
R-AM-MNR	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4
R-AM-TMP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	15
R-AM-PNC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-A4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NULL	0	120	1	0	0	0	0	8	6	0	0	0	0	0	4	1	1	180093

Table C2: Confusion Matrix of Model 2 Notice that roles R-AM-PNC and R-A4 were not part of the development set, but they were predicted by the model since they were met during training.

	A0	A1	A2	A3	A4	A5	AA	AM-ADV	AM-CAU	AM-DIR	AM-DIS	AM-EXT	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-PNC	AM-PRD
TP	2870	4310	1052	156	47	0	0	158	29	15	148	23	198	255	308	114	45	1
FP	493	661	429	89	19	0	0	90	14	17	50	15	131	112	12	9	23	0
FN	576	912	543	146	37	3	1	129	17	19	54	24	143	174	7	9	35	2
Precision (%)	85	87	71	64	71	0	0	64	67	47	75	61	60	69	96	93	66	100
Recall (%)	83	83	66	52	56	0	0	55	63	44	73	49	58	59	98	93	56	33
F1 (%)	84	85	68	57	63	0	0	59	65	45	74	54	59	64	97	93	61	50

	AM-TMP	C-A1	C-A2	C-AM-CAU	C-AM-DIR	C-AM-MNR	R-A0	R-A1	R-A2	R-AM-CAU	R-AM-EXT	R-AM-LOC	R-AM-MNR	R-AM-TMP	R-AM-PNC*	R-A4*	NULL
TP	719	105	0	0	0	0	133	70	2	0	0	2	1	16	0	0	180093
FP	166	14	1	0	0	0	11	12	2	0	0	0	0	4	4	1	1780
FN	159	34	3	1	1	2	11	12	3	3	1	7	5	15	0	0	1071
Precision (%)	81	88	0	0	0	0	92	85	50	0	0	100	100	80	0	0	99
Recall (%)	82	76	0	0	0	0	92	85	40	0	0	22	17	52	0	0	99
F1 (%)	81	82	0	0	0	0	92	85	44	0	0	36	29	63	0	0	99

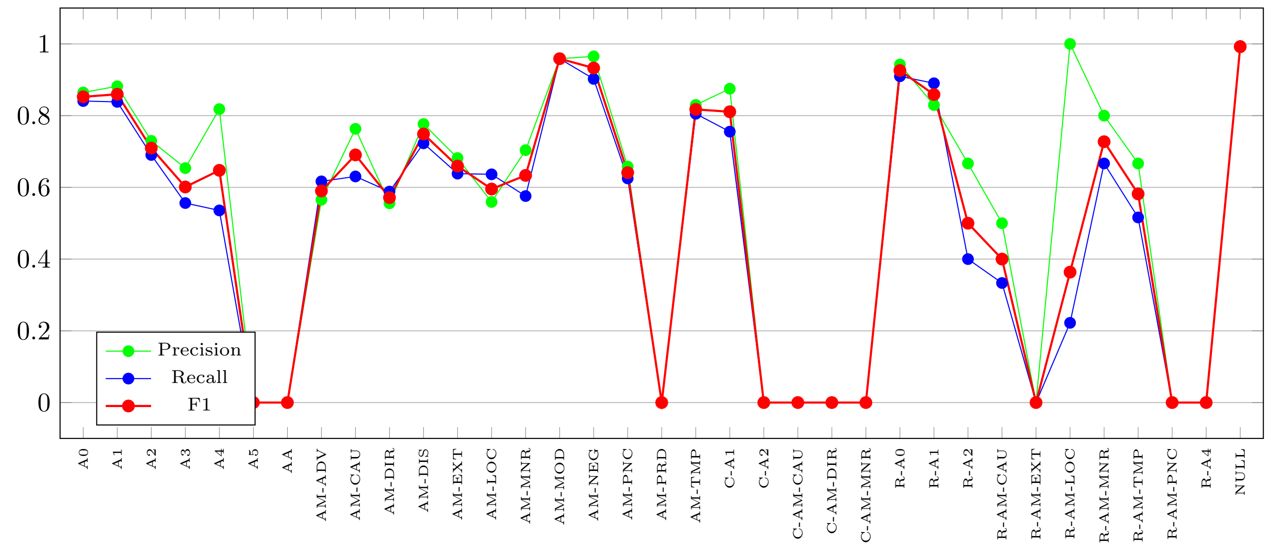


Table D2: Confusion Matrix of Model 2. The variation in precision, recall and F1 metrics for each role.

4.3 Tables of Model 3

	Pre-trained embeddings	Batch size	LSTM hidden size	Drop-out bit	Epochs	LSTM Layers	x_{lemb}	x_{pos}	x_{lex}	x_{sem}	Initial Precision (%)	Initial Recall (%)	Initial F1 score (%)	Final TP (#)	Final FP (#)	Final FN (#)	Final Precision (%)	Final Recall (%)	**Final F1 score (%)**
`Test 3.1`	`FTEXT`	5	100	0.0	15	2	●	●	●	wr	72	72	72	10052	2627	2190	79	82	81
`Test 3.2`	-	-	-	-	-	-	●	●	●	cr	72	70	71	10680	2341	1819	82	85	84

Table A3: Execution of neural Model 3. Hyperparameter tuning via grid search approach. Each test has been performed with a *Gradient Descend* optimizer with learning rate set to 0.2. The setting of the hyperparameters of the tests `T3.1` and `T3.2` was the same as the best execution of model 1, `T1.14`. These two tests were focused on two different approaches in concatenating the sense embedding of a word. In the first case, `T3.1`, the sense embedding is concatenated to the word representation, in the second case `T3.2`, the sense embedding is concatenated to the hidden representation generated by the LSTM layers. The second test shows the best results, and they are as competitive as those of the previous models. The reason why this model doesn't completely outperform the others could be linked to the fact that the disambiguator behind it is not a top quality one.

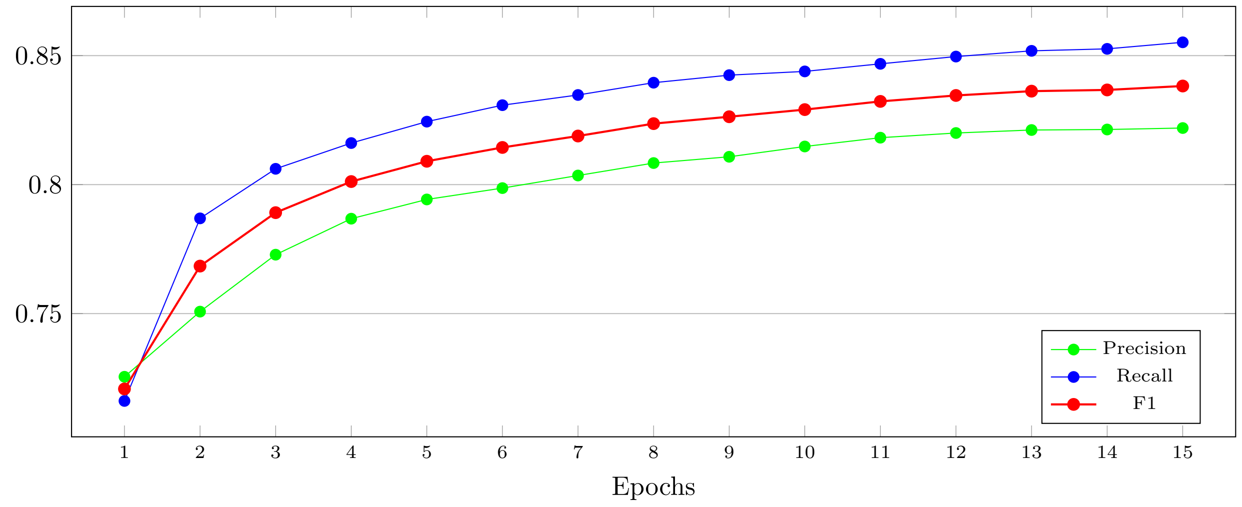


Table B3: Execution of neural Model 3.

T |− P	A0	A1	A2	A3	A4	A5	AA	AM-ADV	AM-CAU	AM-DIR	AM-DIS	AM-EXT	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-PNC	AM-PRD	AM-TMP	C-A1	C-A2	C-AM-CAU	C-AM-DIR	C-AM-MNR	R-A0	R-A1	R-A2	R-AM-CAU	R-AM-EXT	R-AM-LOC	R-AM-MNR	R-AM-TMP	NULL
A0	2849	133	39	3	0	0	0	0	0	0	0	0	5	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	415
A1	127	4428	89	13	1	0	0	2	0	1	0	0	22	7	0	0	1	0	6	1	0	0	0	0	0	2	0	0	0	0	0	0	522
A2	58	175	1018	13	2	0	0	3	0	9	1	0	29	12	0	0	1	0	6	1	0	0	0	0	0	1	0	0	0	0	0	0	266
A3	18	34	25	141	2	0	0	0	0	1	0	0	2	8	0	0	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	67
A4	1	8	8	2	46	0	0	0	0	2	0	0	4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
A5	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AA	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-ADV	0	1	2	2	0	0	0	171	2	0	12	1	0	16	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	65
AM-CAU	0	1	3	1	0	0	0	4	28	0	1	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	4
AM-DIR	0	5	6	0	0	0	0	0	0	15	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6
AM-DIS	0	0	1	0	0	0	0	15	0	0	141	0	1	1	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	37
AM-EXT	0	1	4	0	0	0	0	3	0	0	0	17	1	10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	10
AM-LOC	7	25	20	1	1	0	0	3	0	0	0	0	193	1	0	0	0	0	3	1	0	0	0	0	0	0	0	0	0	1	0	0	85
AM-MNR	4	23	31	5	0	0	0	18	0	1	1	5	7	231	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	94
AM-MOD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	304	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
AM-NEG	0	1	0	0	0	0	0	58	0	0	4	0	0	1	0	19	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	20
AM-PNC	0	3	8	3	0	0	0	4	0	0	0	0	0	0	0	0	48	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	12
AM-PRD	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM-TMP	3	12	4	1	0	0	0	5	0	0	0	0	1	4	0	0	0	0	707	0	0	0	0	0	0	0	0	0	0	0	0	0	141
C-A1	1	9	6	0	0	0	0	1	0	0	0	1	2	1	0	0	1	0	2	104	0	0	0	0	0	0	0	0	0	0	0	0	11
C-A2	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-CAU	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-DIR	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-AM-MNR	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R-A0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	125	8	1	0	0	0	0	0	7
R-A1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	68	0	0	0	0	0	0	9
R-A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	2
R-AM-CAU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
R-AM-EXT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R-AM-LOC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	5
R-AM-MNR	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	3
R-AM-TMP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	11
NULL	247	306	110	16	5	0	0	39	1	7	14	5	37	43	7	1	3	0	103	0	0	0	0	0	9	9	0	0	0	2	1	10	180189

Table C3: Confusion Matrix of Model 3

	A0	A1	A2	A3	A4	A5	AA	AM-ADV	AM-CAU	AM-DIR	AM-DIS	AM-EXT	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-PNC	AM-PRD	AM-TMP	C-A1	C-A2	C-AM-CAU	C-AM-DIR	C-AM-MNR	R-A0	R-A1	R-A2	R-AM-CAU	R-AM-EXT	R-AM-LOC	R-AM-MNR	R-AM-TMP	NULL
TP	2849	4428	1018	141	46	0	0	171	28	15	141	17	193	231	304	19	48	0	707	104	0	0	0	0	125	68	2	0	0	4	1	20	180189
FP	471	743	359	61	12	0	0	158	4	22	33	12	112	106	7	1	13	0	177	4	0	0	0	0	10	21	1	0	0	3	1	10	1819
FN	597	794	577	161	38	3	1	116	18	19	61	30	148	198	11	104	32	3	171	35	3	1	1	2	19	14	3	3	1	5	5	11	975
Precision (%)	86	86	74	70	79	0	0	52	88	41	81	59	63	69	98	95	79	0	80	96	0	0	0	0	93	76	67	0	0	57	50	67	99
Recall (%)	83	85	64	47	55	0	0	60	61	44	70	36	57	54	97	15	60	0	81	75	0	0	0	0	87	83	40	0	0	44	17	65	99
F1 (%)	84	85	69	56	65	0	0	56	72	42	75	45	60	61	97	26	68	0	80	84	0	0	0	0	90	79	50	0	0	50	25	66	99

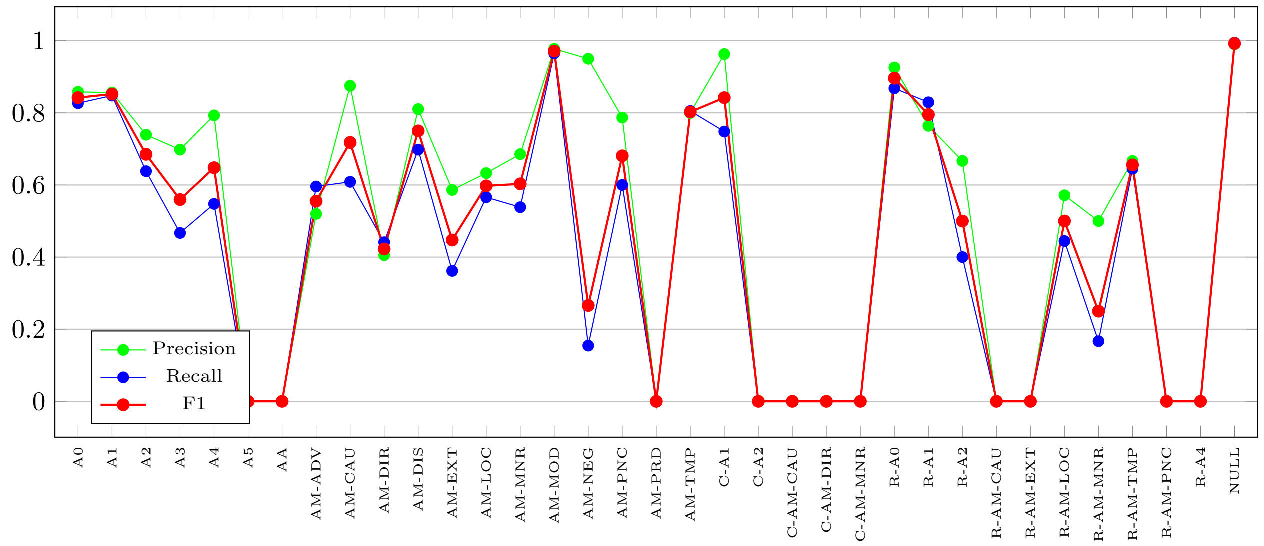


Table D3: Confusion Matrix of Model 3. The variation in precision, recall and F1 metrics for each role.