Semantic Role Labeling using Contextualized word embeddings, the Biaffine Attention Mechanism and exploiting the Syntactic features

Andrea Bacciu
***@studenti.uniroma1.it

1 Introduction

The addressed task is the Semantic Role Labeling - SRL. The objective of the task is to recognize the predicate-argument structure of each predicate in a sentence. For clarity we summarize: a **predicate** defines an action/event; an **argument** is the participant to the action or feature of the event and the **semantic role** is the relationship between the arguments with respect to the predicate. Informally we can state the SRL aims to answer to the questions *Who did What? To whom? How? Where? When?* The SRL pipeline is divided into four steps as shown in figure 3. In this work, I have dealt with the last two steps of the pipeline. In figures 4 and 5 we can see the diagram of the final model and the evolution of the experiments.

2 Dataset

The dataset[1] is in English and already split into three portions train, dev and test, as shown in table 4. In table 3 we can see the label distribution and we can notice that the train is annotated with 34 labels, while not all the classes are present in dev and test which have 31 and 32 classes respectively. The task organizers pre-processed the dataset by tokenizing it and extracting different features: the Part of Speech - PoS, lemmas, predicate list, the dependency relations and the dependency heads, the latter are a list representation of a **syntactic** tree.

3 Pre-processing

For the pre-processing phase, I started doing data exploration. I noticed that the train, dev and test sets, respectively, contain 9.47%, 7.87% and 9.1% of sentences without any predicate, I decided to remove these samples. To predict a sequence of roles w.r.t. a single predicate, I applied the following preprocessing. Given an instance with more than one predicate, such as [_, REMAIN,_,PROVE,_], I duplicated the sample by the number of predicates, inserting one predicate at a time. So I got two distinct samples, one with [_,REMAIN,_,_,_] and one with [_,_,_, PROVE,_]. At the end of the preprocessing phases, the instances of the dataset are increased, as can be seen in table 4.

[1]The dataset used is not public. Was provided by the Sapienza NLP Group

4 Model Architecture

This section describes the components used in the experiments.

4.1 Bidirectional Long Short-Term Memory

The main component of all experiments is the Long Short Term Memory — LSTM in their bidirectional version. Since their advent, they have become the point of reference for many natural language processing problems. The use of a BiLSTM allows to consider the context from left to right and from the opposite side in a sequence.

4.2 BERT – Bidirectional Encoder Representations Transformers

Bert (Devlin et al., 2018) is a model developed by Google researchers, capable of obtaining the *State Of The Art* in various tasks. Bert is based on the encoder part of the transformer architecture and is trained to learn language representation.

Importance of Contextualized Word Embeddings: The (contextualized) word embeddings that Bert produces given a word will be different based on the context of the used sentence, instead, context-free word embeddings (GloVe, W2v) will produce the same vector representation regardless of the context and this limits their effectiveness.

Word pieces: Bert's vocabulary is made up of word pieces. This allow to reduce the size of the vocabulary and make it almost impossible to have

out of vocabulary words. This word segmentation is based purely on frequency, the least frequent words are divided into different word pieces.

4.3 Graph Convolutional Network

As highlighted in (Marcheggiani and Titov, 2017)'s paper, the syntactic representation is very close to the sematic one. In the paper was addresses the SRL task by exploiting the syntactic dependency tree using a particular network called Graph Convolutional Network — GCN, from (Kipf and Welling, 2017)'s research. GCNs are neural networks that operate on graphs and induce features of nodes based on the properties of their neighbours. For this reason, I decided, inspired by (Marcheggiani and Titov, 2017)'s work, to introduce syntactic information into the model, concatenating the encoding obtained by GCN to the word representation described in chapter 4.5.

Syntactic Graph Representation As a syntactic representation, I used the dependency heads tree to built a graph representing it as an adjacency matrix. The nodes in the graph represent the tokens in a sentence and the edges represent the dependencies between them. GCN requires the graph to be indirect and nodes must have self-loops. The self-loops are required to consider the node itself and furthermore, a normalization of the above matrix is required to avoid the vanish/exploding gradient problem.

Combine LSTM Encoder with GCN: (Marcheggiani and Titov, 2017) show how it is possible to overcome the problem that GCN is not able to capture long dependencies between distant nodes in the graph. They solve the problem by placing a BiLSTM encoder as input for GCN instead of a static embedding. The use of BiLSTMs allowed GCN to have input capable of considering the surrounding context, avoiding the increase of GCN-Convolutional layers to consider distant nodes.

4.4 Biaffine Attention Layer

One of the most recent innovations in the field of NLP is the use of the attention layers that made it possible to reach new states of the art in several fields. These layers of attention allow the network to have a better understanding of the encoding sentence by highlighting the most relevant parts. The final model uses an Biaffine attention layer as the last layer, taking inspiration from the works of (Di Fabio et al., 2019). The biaffine attention layer will return scores for each token in the sentence. Let $\mathbf{y}_1$ and $\mathbf{y}_2$ be the inputs of the attention layer, both are the hidden representation of two different 2xBiLSTM $\mathbf{B}_1$ and $\mathbf{B}_2$. Precisely $\mathbf{y}_1$ correspond to the output of $\mathbf{B}_1$ using as input the word representation (see section 4.5); whereas $\mathbf{y}_2$ is made up stacking a second 2xBiLSTM $\mathbf{B}_2$ using as input $\mathbf{y}_1$. Before feeding the LSTM hidden representation $\mathbf{y}_1$ and $\mathbf{y}_2$ into the attention layer, I found beneficial to placed two linear layer, one for each input. This let me to reduce the number of learned parameters in the attention layer and let the model to converge faster. This Biaffine attention layer consist in the use of a bilinear layer $\mathbf{U}$ and linear layer $\mathbf{W}$ with the bias. The symbol $\circ$ is the concatenation operator

$$\text{Biaffine}(y_1, y_2) = \underbrace{\mathbf{y}_1^T \mathbf{U} \mathbf{y}_2}_{\text{Bilinear}} + \underbrace{\mathbf{W}(\mathbf{y}_1 \circ \mathbf{y}_2) + \mathbf{b}}_{\text{Linear}}$$

4.5 Word Representation - WR

Given a word w in a sentence, its representation WR is composed of several concatenations of different trainable embeddings. **GloVe**: I used GloVe 6B with 300-dim (Pennington et al., 2014) as pre-trained word embeddings, it allowed the model to converge much faster with a slightly higher result. **PoS**: One of the most important feature for the model is the PoS embeddings because some pos-tag tokens are highly correlated with specific argument tags. For example the proper noun is strongly correlated with the *agent* class. **Lemmas**: I use the lemmas (embeddings) because they group the same word in the dictionary form, for instance, the lemma *walk* will group the same words expressed in different tenses such as *walking* and *walked*. **Dependency Relations:** Dependency relations are crucial for the SRL task, it turns out as for the PoS, that there is a strong correlation between DepRel tokens and arguments. **Predicate emb**: I performed the predicate embeddings after doing the preprocessing step, see section 3.

The following formula are the word representation used, expressed w.t.r. to the word w.

Base WR: $w^{\text{GloVe}} \circ w^{PoS} \circ w^{Lemma} \circ w^{Predicate}$

Core WR: Base WR $\circ$ $w^{DependencyRelations}$

Where the $\circ$ is the concatenation operator.

5 Experimental Setup

This section shows how I organized and dealt with various tests. The metric used in the task is Macro F1-Score. In all models I propose the Cross Entropy as a loss function and the Adam optimizer

although I have tried other optimizers such as SGD and variants of Adam using different learning rate, the results were not the best. To reduce the amount of overfitting I applied the dropout technique to the various embeddings, to the LSTMs and to the GCN layers. To make the model more robust to OOVs, I dropped all the entries from the vocabularies of words and lemmas with a frequency less than 2. The model used was trained with 13 epochs with batch size 32. Using 32 as the batch size yielded better results than using higher batch values such as 64 or 128. To produce a non-trainable contextualized word embeddings I used the *bert-model-base* with 768-dimension. The large version of Bert didn't help to gain significantly higher performance. As Bert pooling strategy to create the word vector I summed the last 4 hidden layers, which is of the most performing strategy to get more meaningful encoding. In order to calculate the Bert embedding for a given word divided into several subwords, I took the embeddings of his word pieces and used their average as embeddings. For more details on the hyperparameters used, see Table 1.

Tuning Criteria: As a metric to monitor the training I used the dev f1 instead of the dev loss, making use of the regularization technique called early stopping[2], which allows you to restore the model to the best state. As we can see from the loss and f1 plot respectively 1, 2, the local minimum of the loss does not correspond with the maximum of f1. I used the grid search techniques to find the best hyperparameters.

6 Experiments

The score is calculate using the avg. between dev and test f1 classification w.r.t. the previous model.

Baseline Base WR: I conducted my experiments starting from a very simple baseline which immediately gave a satisfactory result, using Base WR, two BiLSTMs and a linear layer. This baseline will be the main component of all subsequent experiments.

Baseline Core WR: Then I moved to the Core WR. (see 4.5 section), that uses the dependency relations which have a strong correlation with the target variable, obtaining an increase of 3%.

GCN: I wanted to exploit the syntactic information, highlighted by Marcheggiani to be very close to semantic information. I added to the Core WR the output of GCN with 2 GCN-Convolutional Layer, bringing an increase of 1%. Making use of a 2xBiLSTM encoder to overcome the fact that GCN is not able to capture long dependencies between distant nodes in the graph as described in section 4.3. However, I also tried adding more GCN-Convolutional layers which didn't bring an improvement.

Bert Embeddings: I have added a non-trainable contextualized word embeddings, to have a better representation of the words, as described in section 4.2. Bert as shown in the (Shi and Lin, 2019) paper is effective in SRL tasks. I applied the 2xBiLSTM encoding to the Bert emb. by concatenating it with the Core WR, the use of encoding improved the f1 by 0.2%. The use of Bert improved the f1 by 1%.

Biaffine: Inspired by the work done in (Di Fabio et al., 2019), I used the Biaffine Attention layer. Attention layers are able to highlight the most relevant parts of an encoding sentence. As a first approach, I tried a solution similar to that of (Di Fabio et al., 2019), using the encoding of the predicate embeddings as input for $\mathbf{y}_2$ (see section 4.4 for the formula used). However, I experimented a more generic approach which led to better performance. Said approach takes into account the whole Word Repr. by using a deeper encoding, by placing a second BiLSTM after **B1**, which should impose a distinction between the attention layer inputs. Supposedly leveraging less restrictions allows the attention mechanism to automatically learn what to focus on, rather than forcing it to the encoding of predicate embeddings only. This layer gave an increase in performance of 0.5%.

7 Results & Conclusion

Table 2 shows the results of the experiments and the ablation study for both subtasks. The combination of all the components made it possible to build the model with the highest score of dev 91.38% and f1 test 92.78%, the model diagram is shown in fig. 4. In fig. 5 we can visualize the evolution of the experiments. I can reconfirm the importance of syntactic features and contextualized word emb. for a SRL system. Furthermore, the use of the Biaffine Att. layer helped to achieve a higher result. Figures 6 and 7 show the confusion matrix of dev and test respectively, from which we can see that the model incorrectly classifies the less present classes in the dataset such as *Material* (see label distribution tab. 3). Despite the many experiments, it is still possible to do a lot: for example, implement the full pipeline using external knowledge such as Verbatlas.

[2]Regularization techniques are used to avoid overfitting

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. VerbAtlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 627–637, Hong Kong, China. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Figure 1: Loss vs Epochs
The blue line is the train classification loss and the orange one is the dev classification loss. As we can see the early stopping on the dev loss indicates 12 as the best epoch. However, the early stopping on f1 suggests training the network over 13 epochs allowing to reach a higher dev f1-score, as we can seen in fig 2.

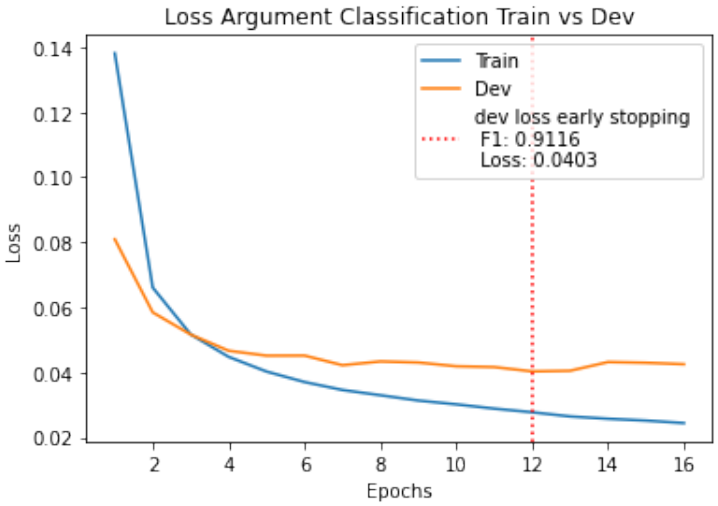

Figure 2: F1 score vs Epochs
The blue one is the f1 score of the dev set on identification parts and the orange line is the f1 score of the dev set on the argument classification.
As you can see from the red dotted line, epoch 13 allows you to reach the highest dev f1 score.

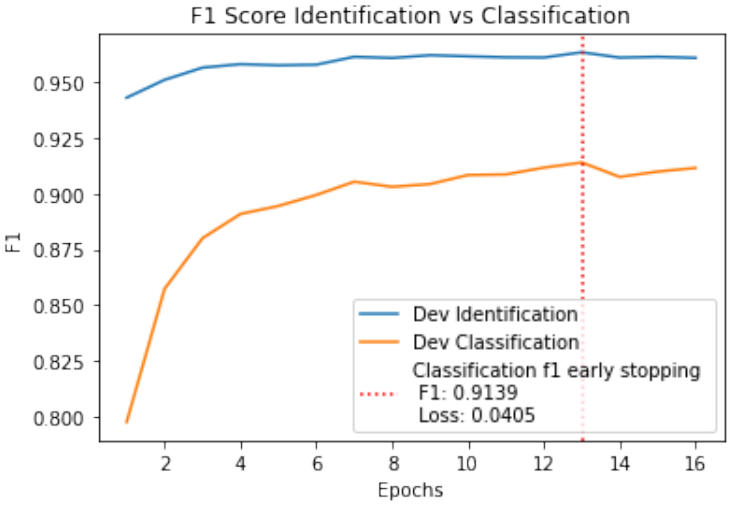

Table 1: Hyperparameter table.
This hyperparameters are found after a several fine-tuning experiments. To decide which set allows to reach the best performance I compare the behaviour of the model using the dev f1 score

HParams	**Value**	**Notes**
Epochs	13	
Batch Size	32	
Optimizer	Adam	
Learning Rate	0.001	
Loss Function	Cross Entropy	
Word Vocab Min frequency	2	
Lemma Vocab Min frequency	2	
Dropout Embeddings	30%	
Word Emb dim	300	GloVe 6B
Bert Emb dim	768	Bert Base uncased
Pos Emb dim	300	
Lemma Emb dim	300	
Predicate Emb dim	400	
Dependency Relations Emb dim	300	
Num GCN-Convolutional Layer	2	
GCN-Convolutional Layer Hidden size	250	first layer
GCN-Convolutional Layer Hidden size	35	second layer
GCN Dropout	50%	
Dropout BiLSTM	30%	
BiLSTM Bert	300 out dim	x2 layer
BiLSTM GCN	300 out dim	x2 layer
BiLSTM Biaffine B1	300 out dim	x2 layer
BiLSTM Biaffine B2	300 out dim	x2 layer
Biaffine inputs	35, 35	y1, y2
Biaffine outputs	35	length of label vocab
Linear layer y1	35	
Linear layer y2	35	

Table 2: Semantic Role Labeling results
The model with the * symbol is the final model.
In bold you can see the highest score obtained.
From this table we can see the variation of the f1 score on the validation & test set for each component.
The experiments below the dotted line is an ablation study. As can be seen from this study, even simple models such as GCN + Biaffine and the model with only Bert achieve satisfactory results. The Biaffine attention layer is a trusted friend, in fact in every test it has always improved performance.

SRL Results				
Experiments	F1-Dev		F1-Test	
	Identification	Classification	Identification	Classification
Baseline + Base WR	90.84%	85.84%	91.80%	87.38%
Baseline + Core WR	94.05%	88.96%	94.52%	90.30%
Baseline + GCN + Core WR	95.68%	89.35%	96.62%	91.25%
Baseline + Bert + GCN + Core WR	95.80%	91.07%	95.90%	92.05%
Baseline + Bert + GCN + Biaffine + Core WR*	**96.32%**	**91.38%**	**96.87%**	**92.78%**
Baseline + Bert + Core WR	95.14%	90.71%	95.72%	91.98%
Baseline + GCN + Biaffine + Core WR	95.78%	89.70%	96.72%	92.28%
Baseline + Bert + Biaffine + Core WR	95.09%	90.80%	95.90%	92.21%

Table 3: This table shows the distribution of the various classes over the 3 available datasets.
In bold I have highlighted all those classes that are not present in all datasets.
The _ is a reserved symbol to tag tokens that are not an argument.

Argument class	Train	Dev	Test
Agent	54.321	1.844	2.718
Asset	1.441	52	62
Attribute	17.405	624	894
Beneficiary	2.169	92	101
Cause	1755	67	82
Co-Agent	652	23	20
Co-Patient	248	7	15
Co-Theme	1.411	51	69
Connective	4.717	201	264
Destination	5.201	133	245
Experiencer	3.432	149	189
Extent	2.907	87	137
Goal	3.094	106	151
Idiom	**20**	/	/
Instrument	919	31	62
Location	6.310	205	340
Material	166	3	11
Modal	9.000	315	460
Modifier	**7**	/	/
Negation	3.165	104	190
Patient	15.388	543	754
Predicative	68	3	5
Product	1.518	73	75
Purpose	2.310	86	105
Recipient	2.811	109	102
Recursive	**14**	/	**1**
Result	4.117	136	174
Source	2.926	88	135
Stimulus	2.559	103	132
Theme	51.662	1.831	2.672
Time	14.789	536	849
Topic	4.933	172	217
Value	776	18	25
_	89.154	3.215	4.450

Figure 3: SRL Pipeline

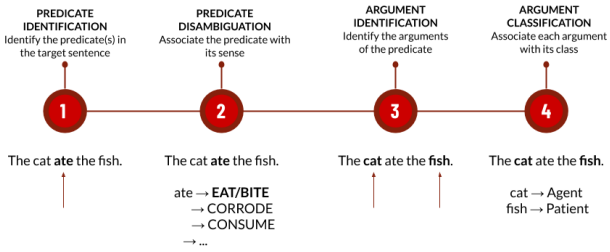

Table 4: Dataset train, dev & test split
In this table are reported the split of the dataset used.
The second column reports the number of instances present in the dataset.
The third column report the number of samples used in the experiments after the preprocessing phase of the section 3

Dataset	# of sample	# sample after preprocessing
Train	39.279	89.154
Dev	1.334	3.215
Test	2.000	4.450

Figure 4: Final model diagram

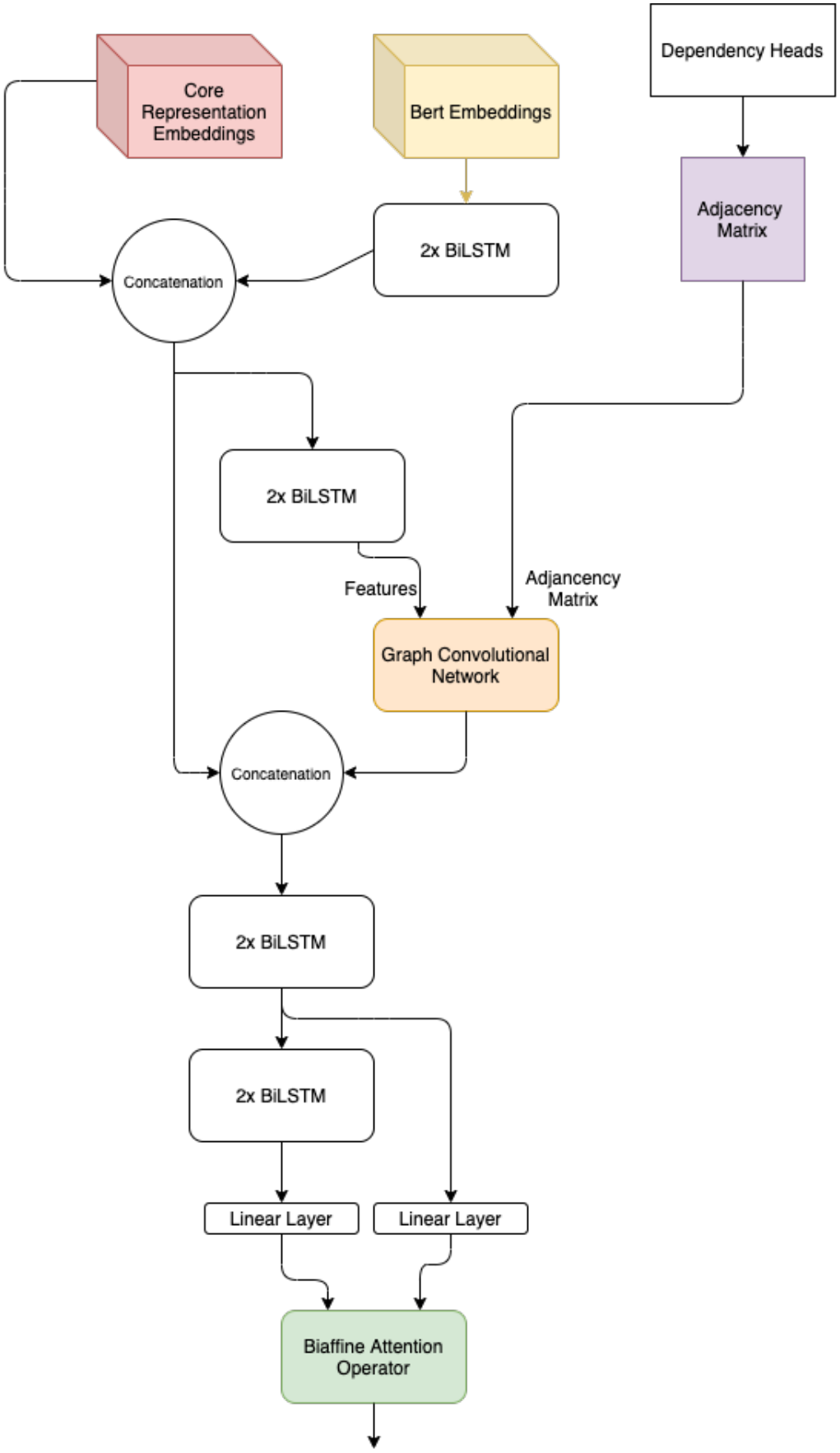

Figure 5: From this image you can visualize the experiments evolution.
The last step is placed in fig. 4

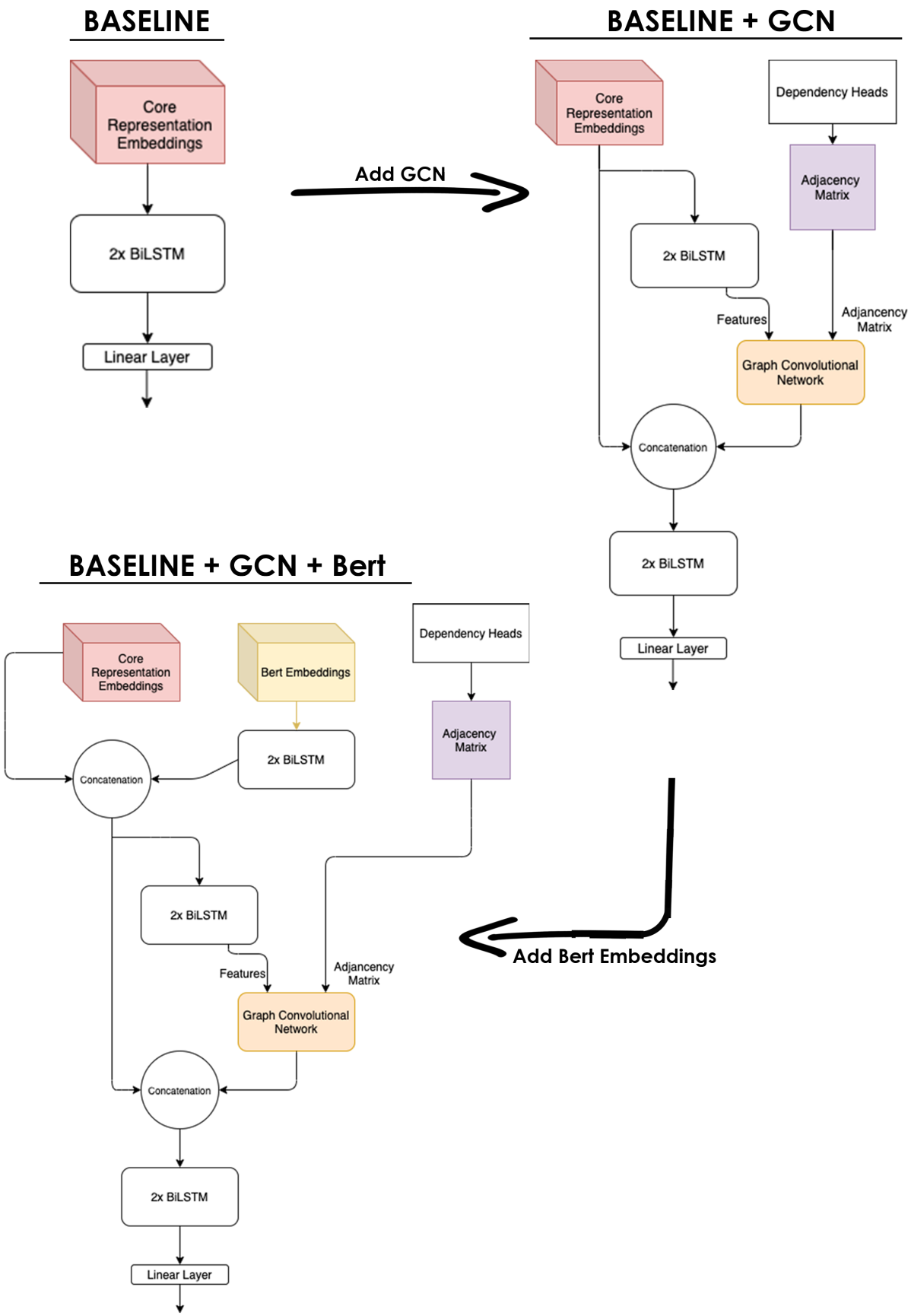

Figure 6: Normalized confusion matrix on Dev set
The true labels are on the y axis and predicted are on the x axis.

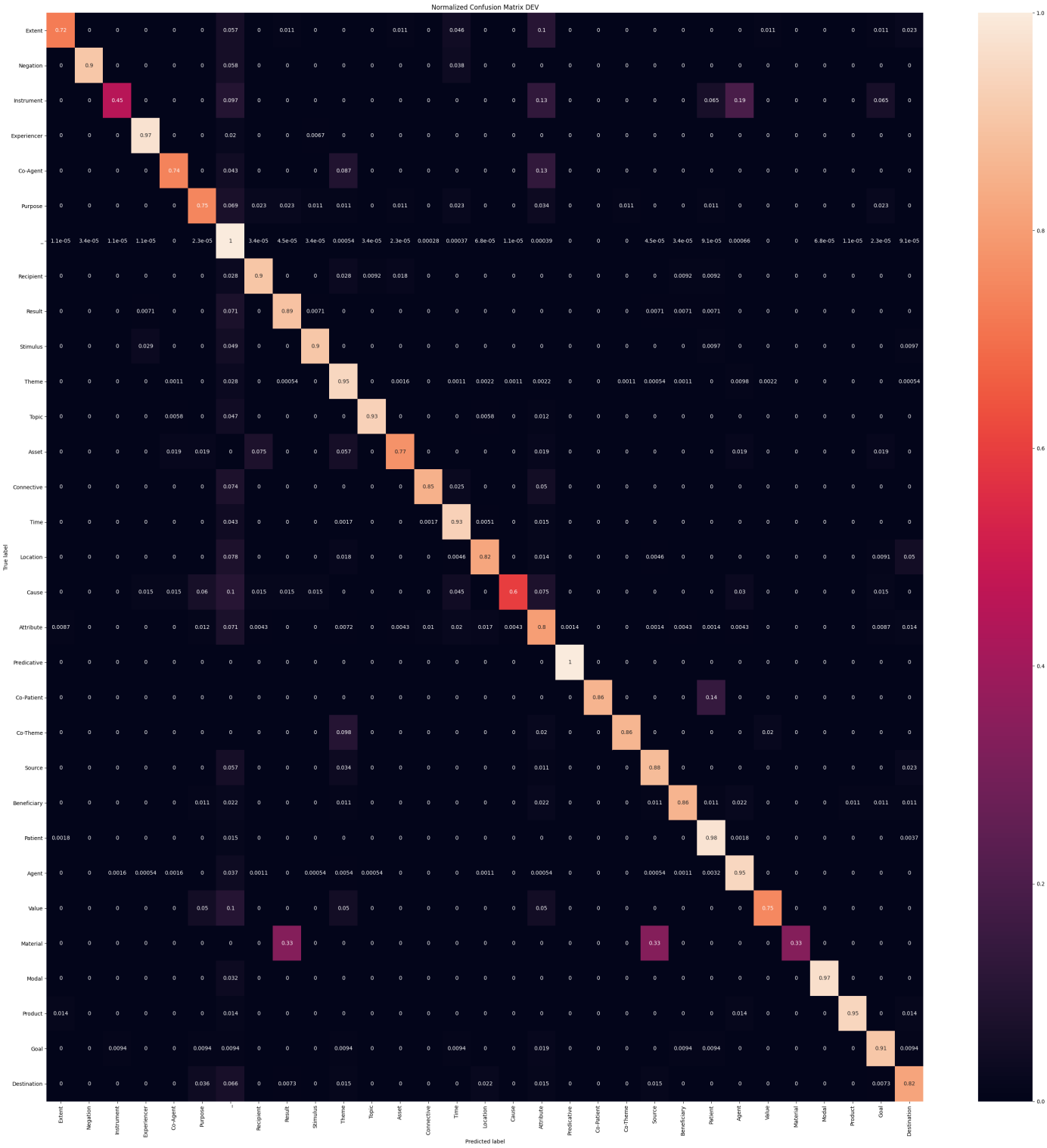

Figure 7: Normalized confusion matrix on Test set
The true labels are on the y axis and predicted are on the x axis.