Natural Language Processing Implement and train a system capable of performing both fine-grained and coarse-grained WSD

Vannoli Marco matr.1860363[*]
Master's degree in artificial intelligence and robotics Department of computer engineering, automatic e management "Antonio Ruberti" University of Sapienza Roma
(Dated: February 14, 2020)

1. INTRODUCTION

The goal of homework was to create a system able to perform both fine-grained and coarse-grained WSD. For the fine-grained, the system has to consider the wordnet synset and so the bebelnet synset obtained through the mapping provided at the beginning. On the other hand, the coarse-grained has to be performed with domains and lexnames with which we able to collect senses into more general categories thereby reducing the sparsity of the data. To create a system able to perform both we had to regard a multitask model whose loss values of results the sum between all loss value among tasks. Moreover for evaluation was used a MSF (backoff strategy) to avoid lost the evaluation of inputs which the system has never seen during training [1].

2. PREPROCESSING

The Preprocessing phase predicted a first creation of the corpus for the input target and each label. at the beginning, all lengths of sentences were considered and then padded in the end but doing this not all WSD's lemmas were taken during preprocessing, especially considering that the length of sentences go beyond 60 reaching also 80. To overcall this problem all sentences were broken on a specific number (i.e default 20) and the model was able to operate with all WSD words. In the corpus for domain task, each babelnet synset whose value is not on vocabulary it was replaced with the domain "factotum" to label each single WSD word avoiding the data loss for domain task. Once each corpus was created the Preprocessing phase is passed on the building of vocabulary for both target and labels. For the creation of each label's vocabularies, all mappings from wordnet to babelnet, from babelnet to domains and from babelnet to lexnames provided were used. As regards the vocabulary for input, word embeddings pre-trained was used instead of using a representation based on one-hot encoding and so improve the efficiency of our network. The word embeddings were trained on Semcor Corpora to avoid the problem of lost lemmas. For each id of WSD on XML file Semcor of Raganato [1], only the first sense key was considered to simplify the process, considering that for many couples or plus of sense key of same WSD id have the same wordnet synset. For example, if we take the id d000.s046.t005 from file XML Semcor which represents the lemma "public" with pos "ADJ" we have two synset key:

$$public\%5:00:00:common:02public\%3:00:00::$$

For both synset key the same synset wordnet is obtained: wn:00493297s. Once the vocabularies were created the training sets and testing sets were obtained in such way each WSD/no-WSD words tagged on Semcor XML were replaced with our keys on vocabulary. To avoid dirty data and focus attention on the WSD was decided to eliminate all punctuation on the corpus. Finally, we have one input and three output corresponding to the number of tasks, one for fine-grained and two for coarse-grained. From the sentence of corpus Semcor: Are you underwriting expensive team trips?

Input:
Are you underwrite expensive team trip
from vocabulary: [1380, 20130, 18928, 6571, 17922, 18623, 0,..0]
Output Babelnet:
Are you bn:00094560v bn:00102576a bn:00076303n bn:00055344n
from vocabulary: [1, 1, 94554, 102570, 76298, 55341, 0,..0]
Output Domains:
Are you factotum factotum animals/biology factotum
from vocabulary: [1, 1, 772, 772, 146, 772, 0,.0]
Output Lexnames:
Are you verb.communication adj.all noun.group noun.act
from vocabulary: [1, 1, 35, 2, 17, 7, 0,..0]

For each output, the value 1 is set when the entity is not in the own vocabulary (UNK) and the value 0 is due

[*] vannolimarco956@gmail.com

to padding. The padding value chosen is 20 as default considering that the integer value chosen with which the sentences are broken is 20 (as default). The shape of each vocabulary is shown in the following table:

Vocabulary Input/Output	
Vocabulary	Shape
Words Embeddings voc.	20176
Babelnet vocabulary	117654
Domains vocabulary	1454
Lexnames vocabulary	46

TABLE I. Size of each vocabulary for multi-task

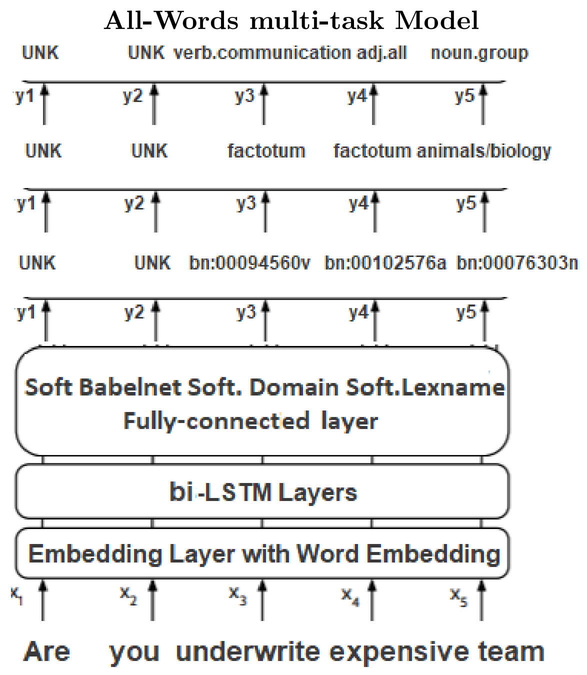
Moreover in the mapping among babelnet and domains are present hierarchical domains of the type:`buildings tourism town_planning`. So decided to consider hierarchically each domain to enclose their hierarchy in the vocabulary. For preprocessing has been decided to implement a class in python to call it whenever is need to be used it. (the name is Preprocessing.py)

3. WORD EMBEDDINGS

To improve the ability of the model to learn from text data it was decided to incorporate the word embeddings on the system. The word embeddings used were trained on corpora Semcor to not have a problem with some lemma missed. It can be useful to see figure 1 on page 4 that shows the plot of all word vectors at once form Corpora Semcor. It don't have been used the word embeddings of homework 2 for two reasons: one is that the goal of homework 2 provided to train words embedding considering the notation lemma_wnsynset and the second reason is based on idea to not have problem during the training of model which might detect a lemma that it never seen. The next phase was the creation of the matrix for the model; it was created using the vocabulary created from word embeddings.The hyperparameters used for trained words embeddings are the same used on homework 2 with model word2vec of gensim: size = 100, window = 25, min_count = 2,workers = 10,sg = 0,hs = 1 ,alpha = 0.025 ,cbow_mean = 1, sample = 1e-5, iter = 5, ns_exponent = 0.7.

4. MODEL

The model chosen is a traditional bi-LSTM based model [2, 3].The model is performed to follow a multi-task structure which provides three tasks: one for fine-grained (babelnet) and two for coarse-grained (domains and lexnames). The choice for the multitasking model was adopted to best perform the fine-ground task with coarse-grained task. The structure of the model is based on one embedding layer in which is set the weights of words embedding trained. This latter weights of word embeddings are set uniformly. The embedding size is set following the size of the vector of each word, i.e. 100. The embedding layer is consequently joint to the bi-LSTM layer in which is applied a dropout regularization algorithm. Finally, the three dense time distributed dense layers are defined through own vocabulary based on task and each of them is joint with the bi-LSTM layer. The structure of the model is the following:



All-Words multi-task Model
UNK
UNK verb.communication adj.all noun.group
y1 y2 y3 y4 y5
UNK
UNK
factotum
factotum animals/biology
y1 y2 y3 y4 y5
UNK
UNK bn:00094560v bn:00102576a bn:00076303n
y1 y2 y3 y4 y5
Soft Babelnet Soft. Domain Soft.Lexname
Fully-connected layer
bi-LSTM Layers
Embedding Layer with Word Embedding
x1 x2 x3 x4 x5
Are you underwrite expensive team

As shown from the schema of the model above, each word not WSD is mapped as output with notation "UNK" because the synset key was provided only for WSD. The best parameters of model tested are based on the idea to obtain the best F1 results which will be analyzed later: The optimizer used for the best model is Adam with

The best Parameters of Model	
Name Parameters	Value Parameters.
max lenght	120
size vocabulary	20176 + (1)
embedding size	100
hidden size	300

TABLE II. The best parameters of Model

learning rate equal to 0,001 because, as will be shown, it able to update network weights iterative based on training data, obtained so good results about metric F1. The best model was trained with 35 epochs with speed about 13/12 min per epoch with GPU of Google Colab obtaining value of accuracy over 90%; besides its loss functions of training sets present a good trend in way decreasing exponential rather than the constant trend obtained by test sets (see figure 3 on page 5).(model was developed with Keras). Twelve models were tested to find the best results about F1.The table 3 on page 4 shows all parameters of the twelve models.

5. EVALUATION

Once the three probabilistic tensor, i.e. one for fine-grained and two for coarse-grained, are computed, the evaluation of F1 for each task is started. The evaluation is based on the idea to take the candidate synsets for a given lemma e pos from data evaluation for each WSD words and then take the most probable among them from tensor through the mapping provided. So, as regards to fine-grained (babelnet), once the candidate wn synsets are obtained, the relative babelnet synsets are extract from mapping and the most likely is get from the likely babelnet synset's tensor. As regards to coarse-grained,once located the candidates of babelnet synset from lemma and pos, the domains and lexicographer's id are get from the mapping provided and the most probable among the candidates is extrapolated from the corresponding tensor to the task,(the code was written to use the same method in order to gets the candidates for the task required). The stage about taking the candidates and then computing the most likely is characterized by the MFS backoff strategy. The MFS strategy was incorporated whenever lemma never seen by the system. So it was taken the most frequent synsets and subsequently, the prediction is computed with the same modality described before. As for domains, whenever the single candidate babelnet synset is not in the mapping about domains then it is taken the "factotum" label.This last method was used only for domains as they are the only ones that are not completely mapped with babelnet. This last chose affects F1 results for domains as there will be as many factotum as absent babelnet synset in the mapping provided (p.s the script Scorer.java was uses for evaluation).

6. DISCUSSION ABOUT FINAL RESULTS

As has been said before, twelve models were tested to find the best results about F1. All results are reported in table 4 on page 5. From this latter table emerges that the switch from sgd optimizer to Adam optimizer brings the best results because it combines the advantages of two sgd extensions (RMSProp and AdaGrad) and it update network weights during training of model. The optimizer Adam As testing set, the semval2007 and semval2013 have been used but only semval2007 bring good results on evaluation. Another key point about the result is the overfitting due to the big difference in size among the training set (Semcor) and test set (Semval2007 and Semval2013). This last issue justifies the presence of the dropout regulation technique in the bilstm level. I preferred to don't use Semcor as a test set because wanted to have a training set and test set coming from different corpora. The initial value of 0.5 for both dropout and recurrent dropout still maintained the problem of overfitting and for this reason I decided to bring them to 0.8, thus increasing the performance of the model as is shown in the figure 2 on page 4. Its use has raised the score results from an average of interval among 50,4-52,8% (average between results F1 amoung fine-grained and coarse-grained) to 64,1-80,1%. From the results, it emerges that as far as the fine-grained is concerned, the prediction about the babelnet synset starts from a 61.8% (semval2007) up to a maximum of 76.6% (senval3) while the prediction about the domains and lexnames starts from a 81,5% (semval2007) to 89,3% (senval2) for domains and 75,6% (semval2007) to 86,7% (semval2013) for lexicographer's id. From the results obtained by the models 11 and 12 can be note that increasing the epochs has been obtained an overall improvement of the coarse-grained results while a worsening of the fine-grained slightly inflected by sparsity and overspecification. The latter mentioned problems justify the inferiority of the results of the fine-grained compared to those of the coarse-grained.

7. CONCLUSION

The adoption of an integrated system with the MFS backoff strategy and bi-LSTM with regulation system has led to good results both from the prediction of the fine-grained and from the coarse-grained. Furthermore, the adoption of a multi-task model has allowed us to integrate both fine-grained and coarse-grained WSD in a performance manner, leading to better results rather than the only training of synsets and their subsequent mapping to obtain their corresponding domains and lexicographer's IDs. As future work, I plan to extend the evaluation to larger sense-annotated corpora (Semcor + OMSTI, T-O-M,...) and integrate the model with an encoder-decoder architecture for sequence-to-sequence WSD, with 2 bidirectional LSTM layers and an attention layer [2].

8. REFERENCE

[1] Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli *Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison In Proceedings of European Chapter of the Association for Computational Linguistics (EACL).* Valencia, Spain, April 3-7, 2017.

[2] Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli *Neural Sequence Learning Models for Word Sense Disambiguation* Department of Computer Science Sapienza University of Rome, 2017.

[3] Mikael Kageb ack and Hans Salomonsson. *Word Sense Disambiguation using a Bidirectional LSTM.* 2016

list of each Model's parameters												
Models	1	2	3	4	5	6	7	8	9	10	11	12
max lenght	120	(- 120)	-	-	-	-	-	-	-	-	-	-
size vocabulary	20176+(1)	(- 20177)	-	-	-	-	-	-	-	-	-	-
embedding size	100	(- 100)	-	-	-	-	-	-	-	-	-	-
hidden size	200	300	300	500	500	500	250	650	650	650	300	300
optimizer	SGD	SGD	SGD	SGD	SGD	ADM	ADM	ADM	ADM	ADM	ADM	ADM
epoch	5	8	10	(- 10)	10	10	10	20	20	30	20	35
batch	64	128	64	32	(- 32)	-	-	-	-	-	-	-
dropout	0,5	(- 0,5)	-	-	-	-	-	-	-	-	0,8	0,8
Semval test	2007	(- 2007)	-	-	2013	-	-	-	-	-	-	-
The best model	no	no	no	no	no	no	no	no	no	no	yes	yes

TABLE III. Parameters adobted for the twelve models

FIG. 1. **TSNE from Semcor training corpora**

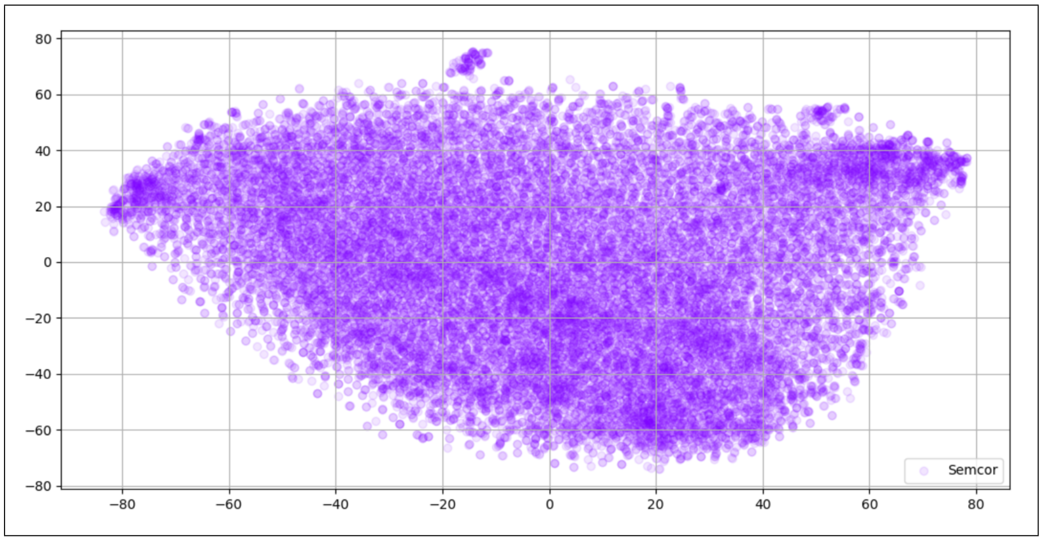
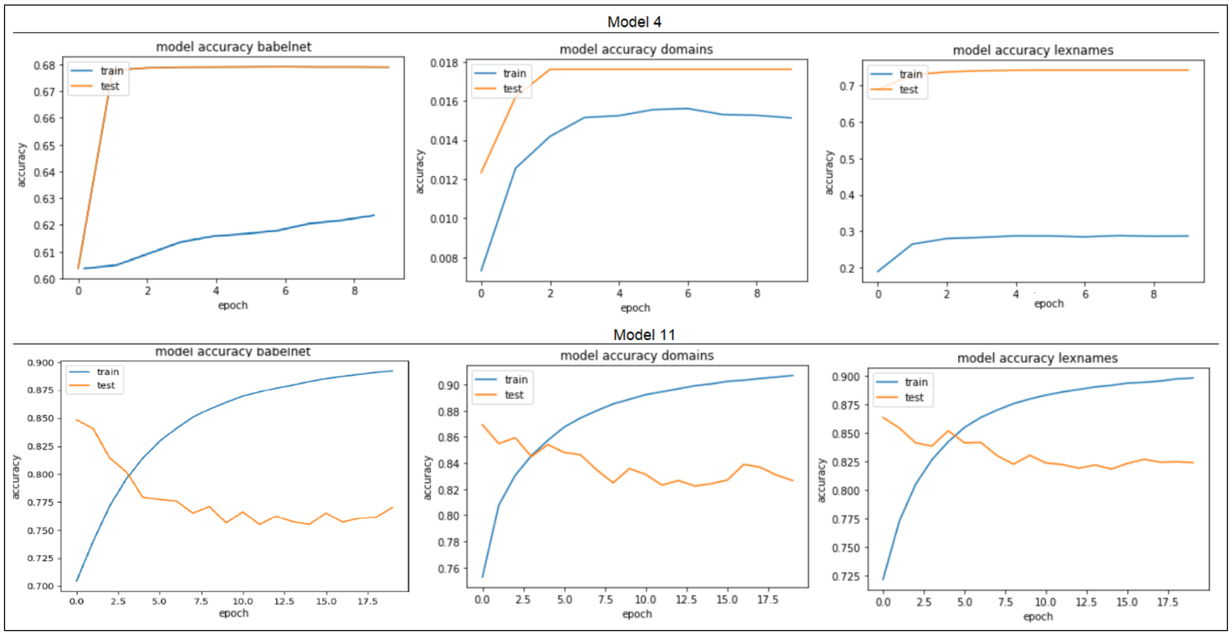


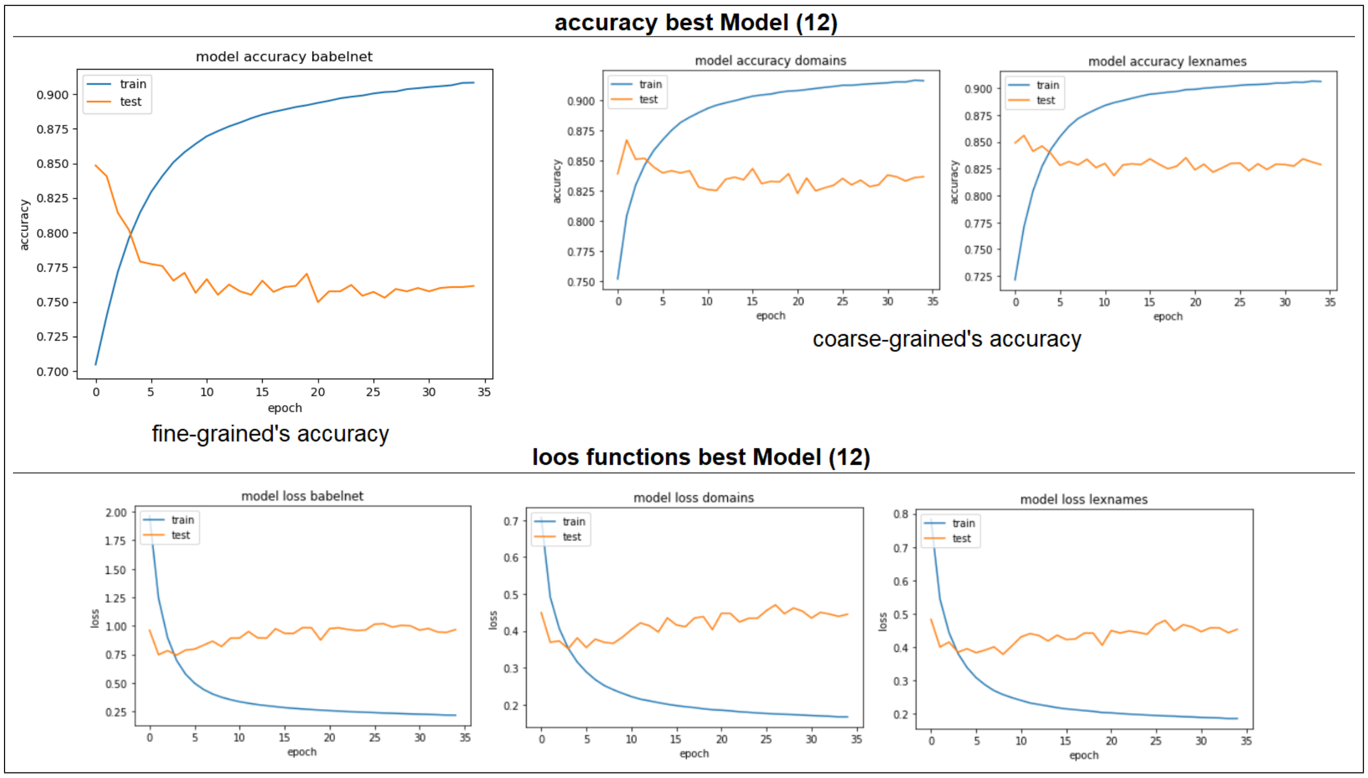
FIG. 2. **plot of accuracy for multi task: on left model 4 and on right model 11**



F1 results %													
Data ev.	Task	Mod.1	Mod.2	Mod.3	Mod.4	Mod.5	Mod.6	Mod.7	Mod.8	Mod.9	Mod.10	Mod.11	Mod.12
Semval2007	babelnet	24,0%	22,0%	26,4%	22,2%	22,9%	54,5%	52,7%	55,2%	55,2%	54,5%	60,9%	61,8%
	domains	58,9%	57,6%	66,4%	63,3%	62,4%	66,2%	65,9%	65,9%	65,9%	65,9%	**80,9%**	**81,5%**
	lexnames	47,3%	44,2%	43,5%	41,8%	44,4%	39,6%	39,6%	39,1%	39,8%	39,1%	**75,8%**	**75,6%**
Semval2013	babelnet	37,3%	39,2%	38,2%	38,0%	41,1%	**71,4%**	69,3%	**72,3%**	**70%**	**72,2%**	**74,1%**	**73,9%**
	domains	55,0%	52,8%	59,2%	55,9%	55,6%	59,5%	58,5%	58,5%	58,5%	58,5%	**82,1%**	**83,7%**
	lexnames	55,3%	49,7%	53,8%	56,2%	50,7%	60,0%	58,0%	57,8%	58,5%	57,7%	**86,3%**	**86,7%**
Semval2015	babelnet	31,2%	39,8%	34,2%	39,4%	34,9%	68,8%	67,4%	**70,8%**	69,4%	69,5%	**73,5%**	**73,8%**
	domains	68,0%	67,1%	**72,7%**	**70,6%**	67,9%	**73,7%**	**73,6%**	**73,7%**	**73,6%**	**73,7%**	**84,4%**	**83,4%**
	lexnames	55,9%	57,6%	57,2%	58,5%	52,9%	61,6%	60,6%	60,9%	60,8%	61,4%	**84,0%**	**84,7%**
Senval2	babelnet	36,1%	37,6%	38,0%	39,4%	36,9%	67,0%	65,2%	68,5%	66,5%	68,6%	**74,0%**	**71,9%**
	domains	**72,5%**	**70,8%**	**76,4%**	**75,6%**	**71,5%**	**76,8%**	**76,8%**	**76,6%**	**76,8%**	**76,7%**	**89,2%**	**89,3%**
	lexnames	61,9%	63,7%	62,2%	61,0%	60,4%	64,8%	64,4%	64,4%	64,6%	64,5%	**85,5%**	**85,8%**
Senval3	babelnet	32,9%	35,9%	34,2%	34,7%	33,3%	69,1%	68,8%	69,2%	69,1%	69,5%	**75,7%**	**76,6%**
	domains	66,0%	63,6%	**71,7%**	69,9%	69,1%	**72,3%**	**72,8%**	**72,8%**	**72,7%**	**72,8%**	**87,6%**	**87,4%**
	lexnames	54,1%	54,9%	55,2%	54,0%	54,2%	57,0%	58,3%	57,9%	58,1%	57,7%	**85,1%**	**85,4%**
Average All	babelnet	32,3%	34,9%	34,2%	34,7%	33,8%	66,2%	64,7%	67,2%	66,04%	66,9%	**71,7%**	**71,6%**
	domains	64,08%	62,38%	69,8%	67,6%	65,3%	69,7%	69,52%	69,5%	68,9%	69,5%	**84,84%**	**85,1%**
	lexnames	54,9%	54,02%	54,38%	54,3%	52,5%	56,5%	56,2%	57,9%	56,4%	56,1%	**83,3%**	**83,6%**
	Average Tot.	50,4%	50,4%	52,8%	52,2%	50,4%	64,1%	63,5%	64,9%	63,78%	64,1%	**79,9%**	**80,1%**

TABLE IV. F1 results of each models tested for each task

FIG. 3. **accuracy and loss functions of the best model**


accuracy best Model (12)
model accuracy babelnet
model accuracy domains
model accuracy lexnames
train
test
accuracy
epoch
fine-grained's accuracy
coarse-grained's accuracy
loos functions best Model (12)
model loss babelnet
model loss domains
model loss lexnames
loss