NLP-2022 Homework 2: Semantic Role Labeling

Dennis Rotondi
1834864
Sapienza, University of Rome
rotondi.1834864@studenti.uniroma1.it

1 Introduction

Whereas last time we had to face NER, today we are engaged in another of the most important and fascinating NLP tasks: Semantic Role Labeling (SRL). SRL aims to automatically extract the predicate-argument structures within a given sentence. Already between the 7th and 4th centuries BCE in a famous treatise of Sanskrit grammar, commended as "one of the greatest monuments of human intelligence", the importance of these structures were being investigated (Jurafsky and Martin, 2022). Moreover, several modern downstream tasks like question answering can be linked to SRL (Xu et al., 2021), therefore different strategies have been developed to solve this problem; in what follow I'll try to attack it by a divide and conquer approach completing: task-1 *predicate identification*; task-2 *predicate disambiguation*; task-3&4 *argument identification and classification*. Predicate-argument relations are not confined to a single language, actually developing a robust cross language system can even improve machine translation (Liu and Gildea, 2010). For this reason I'll pay attention to the language aspect for spanish (ES) and french (FR), but always giving priority to the english (EN) one since it is our main evaluation criterion.

2 Method

As anticipated I've built different models to assemble a pipeline for SRL. My idea has been to train each of them independently so to have a robust solution that could be later fine-tuned for other languages, exploiting at each sub problem only some information that we could take by processing previous levels. This system has in theory more generalization power, but speaking of the final result it is not optimal since we are limiting ourselves not going "all in" for example by collecting all the syntactic and semantic information (pos-tag, wsd, etc) at each level. This latter approach is maybe the best if you want to achieve SOTA but, due to my hardware constraints that do not even allow to think about it, I've taken advantage of this homework to just explore different concepts we studied during the lectures avoiding cumbersome networks.

2.1 Dataset and Preprocessing

First of all, when dealing with SRL data it is important to have a linguistic inventory that defines what the labels mean: in our case it is VerbAtlas (Di Fabio et al., 2019), which provides human-readable and informative role labels shared across different frames. There is nothing much to say about preprocessing steps here, the provided datasets (EN -biggest-, ES and FR) have all to start our job and although it is given with I guess the idea of train a pos-tag model (we do not have pos-tags at evaluation time), I decided to use the pre-trained tagger spaCy (Montani et al., 2022) to have ready-made POS for task-1. I've also involved AMuSE-WSD (Orlando et al., 2021) prediction data, which achieve SOTA results in multilingual WSD, to improve task-2, but since it returns BabelNet synsets (Navigli and Ponzetto, 2010) I've converted them in VerbAtlas frames using the available tables.

2.2 BERT the Transformer embedder

In all my networks, Transformers (Vaswani et al., 2017) have a central role. This relatively new kind of architecture has revolutionized numerous NLP areas attaining SOTA in most of them. The attention mechanism, their real introduction, can be described as mapping a query and a set of key-value pairs to an output (the key/value/query concept is analogous to retrieval systems and they are produced from the input vectors). Transformers are huge models which one of the biggest benefits comes from how they can be parallelized, but their great size to be exploited at its best need lot

of resources that unluckily I haven't. Hence, I've relegated them to the role of words embedder, and in particular I've loaded $BERT_{base}$ (Devlin et al., 2018) which uses bidirectional self-attention and it's pre-trained on 'Masked LM' and 'Next Sentence Prediction' unsupervised tasks. It comes associated with a word-piece tokenizer with a 30000 token vocab. so even for OOV words it's possible to, as I've done, average their splitting tokens.

2.3 Tasks 3 and 4

I've decided to solve the mandatory problems with a single model to best fit the TAs request. We need to predict the roles in the sentence associated to each predicate, so it's crucial to use different embeddings at inference time in function of the desired predicate. It's here that classical word embeddings fail, because they cannot capture such aspect, and BERT comes to the aid: on the wake of (Shi and Lin, 2019) I've used for each predicate in the sentence a context-question pair (sentence, (predicate, predicate-sense[1])), then the Transformer will produce different embeddings for each word in the sentence and its output will be first processed by a LSTM (Hochreiter and Schmidhuber, 1997), that at this point we know works very well as sequence encoder, and later by a single linear layer to make the final prediction. In addition, in order to smooth the training only for task-3 results, there is another linear head on top of LSTM which mission is to identify the arguments. In Figure 1a a summary.

2.4 Task 2

To disambiguate the predicates we do not really need a context-question pair, but I kept BERT anyway because I like the word-piece embedding more than using an OOV token, and in general the quality is better. What we concatenate to the transformer's output is the prediction of AMuSE-WSD, since it's clear that having reliable disambiguation from all the sentence can help the specific case. However, AMuSE disambiguate in all the VerbAtlas's frame (432) while in our english dataset we have only a portion of them (303), therefore I've built a fixed conversion layer on top of it that maps the WSD predictions outside our range of action in the most probable that is inside the reference dataset we have. The result of the concatenation is simply analyzed by a three layer MLP classifier. In Figure 1b a summary of this solution.

[1]Note that this information comes from task-2

2.5 Task 1

Predicate identification is both the simplest and most important task, everything relies on the fact that we know our predicates, indeed the final objective is to extract their arguments, hence we need to recognize as much of them as possible. How one can imagine, even if not all the predicates are verbs there is a strong correlation between these two syntactic structures and that is confirmed by an analysis on the english dataset in Figure 5. Therefore I've used the small version of spaCy to produce pos-tags information at inference time, each tag is concatenated to the relative omnipresent BERT word embedding and a sequence of a GRU and a one layer MLP will perform the identification. GRU informally is a simplified LSTM, so it has the same role: the sequence encoder is needed because the network has to understand that when there aren't verbs presumably the predicate hide in a nominal sentence. In Figure 1c a summary.

3 Training Setup

Training these architectures is basically solving binary-classification (task-1 and task-3) and multi-class classification (task-2 and task-4) for which, respectively, Binary-Crossentropy and Crossentropy losses are known to perform sublimely. For these training I've reused most of the tools that worked well in hw1, starting from the adaptive optimizer I've also applied 'EarlyStopping' and 'ReduceLROnPlateau' techniques to minimize overfitting and oscillations while monitoring the f1-scores. An exhaustive collection of HP is found in Table 1. In Figure 2 good trend of the losses for tasks-3&4.

4 Experiments

These networks are the results of many experiments and in this section I'm going to illustrate the most interesting reasonings that did allow to achieve my highest scores (for english if not specified).

4.1 Words or Lemmas?

Our datasets contain both the words and lemmas that compose a sentence, at a first glance I've thought it was redundant since lemmas in general comes after processing the words, but then I've realized that for some tasks the information provided by one are more useful than the other, and not always lemmas are better than words: for example if we want to identify a predicate such as "worked" (in word form) $\rightarrow$ "work" (lemmatized), that can be

confused with the noun, is counter-productive, as it's possible to see in Figure 3, a direct consequence of the spaCy's performances in the two cases Table 2. On the contrary, I've found that lemmas are better than words for tasks-3&4 Figure 4.

4.2 BERT layers

Transformers are cardinal, but are the whole of them important at the same level? Each of the 12 layers of $BERT_{base}$ capture disparate features from the words that can help in a task or not. Due to their millions of parameters they are difficult to load in memory, several studies have been published (Sun et al., 2020) to understand where focusing the training resources. Based on these studies and taking into account that I'm using them as encoder, I've decided to unfreeze different layers and check whether the performances follows the research results. Running this comparison on task-4 I've confirmed that fine tuning the words embedder is better, and doing it on the final blocks guarantee higher scores than in central ones Figure 6.

4.3 AMuSE and multiple languages

So far I've concentrated my analysis on english but there are some aspects that take care of the multi-language problem in the pipeline too. With this purpose the appropriate spaCy is loaded at inference time and the advised AMuSE (by means of API middleware) has been chosen for it's fantastic results on handling multiple languages. As mentioned I've refined its predictions and the improvements are collected in Table 3, here it's possible to note that if for EN we have a respectful score of .754, for ES and FR it decreases abruptly (.541 and .235). I've also checked that this is not due to bad parameters, and indeed if we predict spanish sentences with EN we got a poor .03, it simply really struggles with these datasets. This is a problem because even if my intention was to use it as "zero shot" model, just to make the right disambiguation extracting the predicates' sense, this is clearly not possible. I've decided to benefit from this extra for understanding how bad predictions impact on a simple model like the one proposed for task-2 that effortless tries to correct the one behind. The effect must be nothing enjoyable since the model presented for ES and FR are the result of transfer learning from EN and if for this latter the quality of AMuSE's disambiguation is much higher than the others we are not really transferring anything.

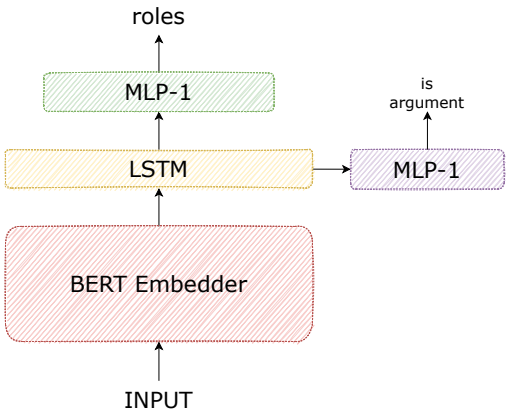
5 Results

Table 4 reports all my efforts to complete this project: here are evaluated in terms of precision (P), recall (R) and F1 (F) all the combinations of tasks and languages, calling for example '34-FR' the model that solve tasks 3 and 4 together for the french language. Having used BERT that is pre-trained on english corpus it's clear that most of the best scores are for that language, still it's impressive how for predicate identification Spain got the gold medal, this is a typical case of a model which learns to extract features intrinsic in the sentences' sense more than their syntax. For EN going deeper in the pipeline, so solving more and more tasks, the F1 drops only of .06, synonym of a very robust system taking into account that networks are independent. Unluckily we have to add another $\sim$5% error for ES and $\sim$12% for FR which suffer most the weakness in task-2 and the cascading failure. In Figure 7 we can observe a normalized confusion matrix for the output roles of model 34-EN: it measures how many times an instances of $class_i$ (on row) is predicted as $class_j$ (on columns); and as expected in this metric the lower the samples of the class the lower the results, indeed for role 'topic' with 1403 samples in the dataset we get it correct 94% of times, while for 'material' with only 11 samples a miserable 0%. It's interesting to outline that the 'agent' role has a score of 85% although it counts 7581 copies, the problem here is that these labels are shared across frames and 'agent' is really general, hence common for many predicates.

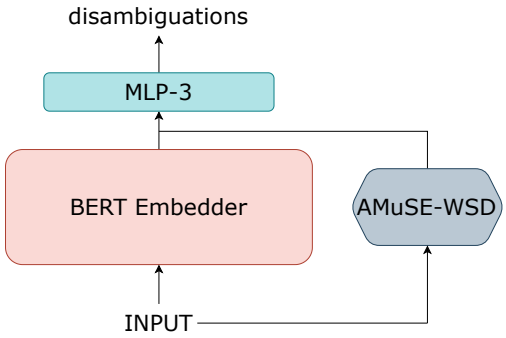
6 Conclusions

I'm satisfied with the result illustrated, however if I had access to more physical and temporal resources I would have liked to explore bigger networks such as $BERT_{large}$ and trf_spaCy, fine-tuning them without being scared of fill up the vRAM, to see how far my scores could go. If one is really interested in maximizing the score for other languages the best Transformer to use is RoBERTa (Liu et al., 2019), maybe performing a multi-language training merging all the samples, and since AMuSE-WSD has some problems with the verbs in these datasets a better approach could be to put aside it and use additional semantic or syntactic information, like the dependency_relations. Eventually to master the core task of this homework could be gripping to build a system that refines the final predictions by enforcing a coherence between the output roles.

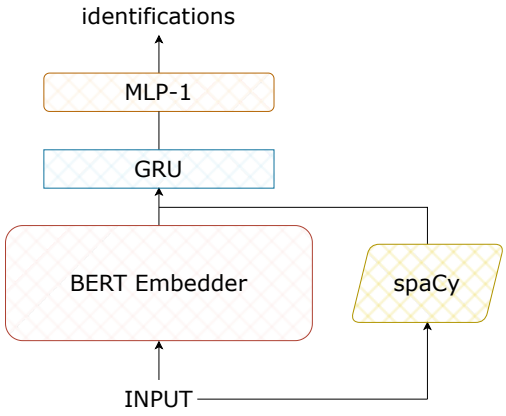
Figures and Tables[2]



(a) Sketch of tasks-3&4's model



(b) Sketch of task-2's model



(c) Sketch of task-1's model

Figure 1: Sketches of the models that constitute the pipeline, from the last to the first. Note that MLP-X means a Multi Layer Perceptron with X layers and the spaCy and AMuSE predictions are embedded automatically by the networks.

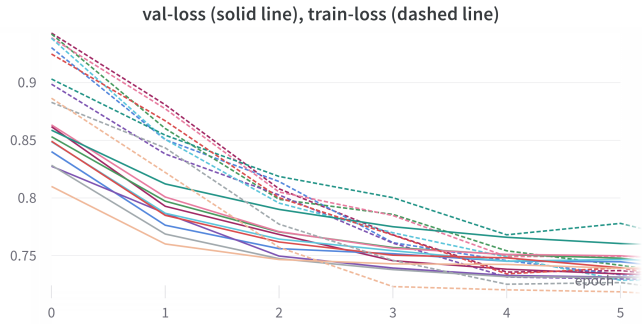


Figure 2: Ensemble of losses for dev and train set plotted during tasks-3&4 training.

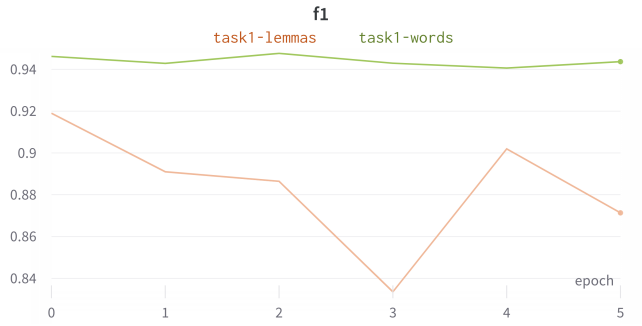


Figure 3: A plot that shows the evolution of the F1 score during the training for task-1, it's clear that for the model is easier to learn using words instead of lemmas what is a predicate, the oscillatory behaviour we can observe in the lemmas' case is probably because it's confusing words that have the same lemma eg as noun and verb.

Figure 4: A plot that shows the evolution of the F1 score for tasks-3&4. In this experiment lemmas performs better than words during the entire training period.

[2]zoom the plots and images if you want to see them better

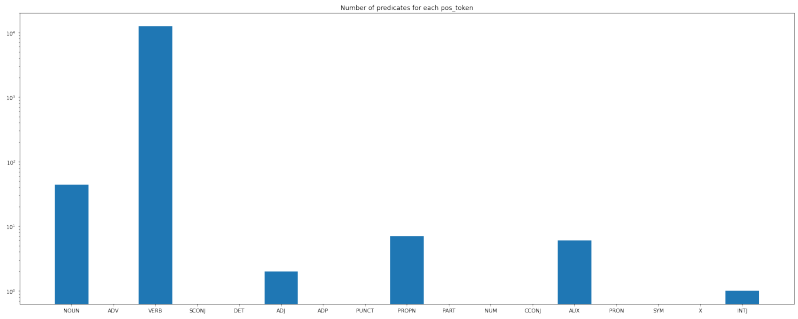


Figure 5: A bar plot that shows the correlation between pos-tags and predicates in the english dataset. Note that to have a clear visualization of numbers a logarithmic scale has been used. As expected the verb quantity is orders of magnitude greater with respect to the other tags.

batch-size	256
frames_embedding-dim	256
language model	bert-base-uncased
learning rate	1e-3
optimizer	Adam
pos_tag_embedding-dim	232
rnns_embedding-dim	400
rnns_bidirectionality	True

Table 1: Hyperparameters for the different training

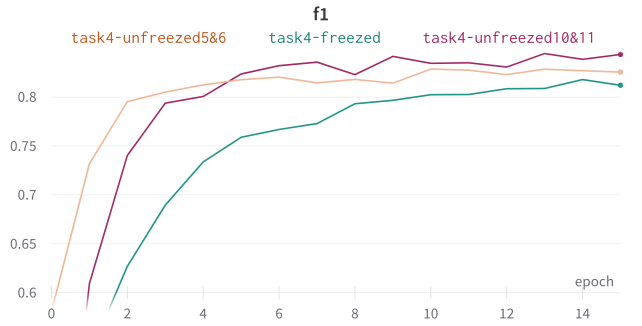


Figure 6: A plot that shows the evolution of the F1 score during the training for task-4 comparing three models: one with freezed embeddings (the worst), one with only layers 10 and 11 unfreezed (the best) and one with layers 5 and 6 unfreezed.

spaCy pos-tags' accuracy

input	on all	on verbs
words	0.879	0.840
lemmas	0.851	0.678

Table 2: Accuracy of spaCy pos-tagger. While in general performances are comparable, for verb tags there is a huge difference when the inputs are words and when lemmas in our dataset.

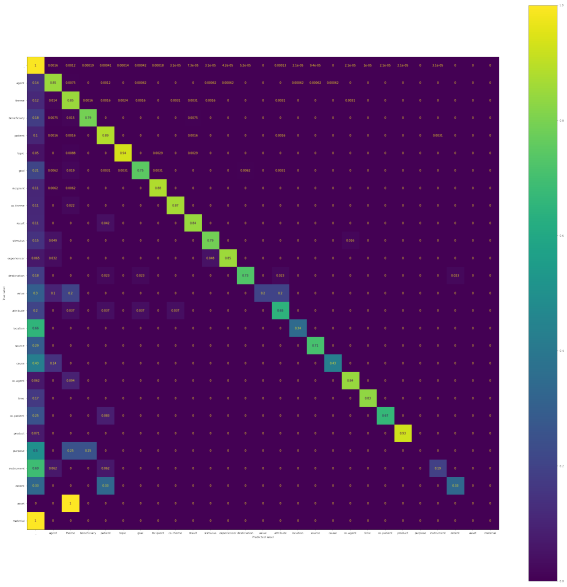


Figure 7: A confusion matrix for argument classification, the best I've obtained during my experiments on the english dataset. The more yellow a cell, the higher the chances to predict the row class as the column class.

AMuSE F1 pre and post refinement

language	pre-layer	post-layer
EN	0.730	0.754
ES	0.527	0.541
FR	0.229	0.235

Table 3: F1 on the dev-set of AMuSE predicate disambiguation: pre and post the 'most-likely' refinement described (on the train-set).

model	pred. identification			pred. disambiguation			arg. identification			arg. classification		
	P	R	F	P	R	F	P	R	F	P	R	F
34-EN	-	-	-	-	-	-	.905	.867	**.886**	.876	.840	**.858**
34-ES	-	-	-	-	-	-	.865	.658	.748	.802	.610	.693
34-FR	-	-	-	-	-	-	.808	.699	.750	.740	.640	.686
234-EN	-	-	-	.877	.866	**.871**	.901	.856	.878	.843	.800	.821
234-ES	-	-	-	.584	.565	.574	.857	.636	.730	.688	.510	.586
234-FR	-	-	-	.510	.409	.454	.798	.564	.661	.612	.433	.507
1234-EN	.935	.959	.947	.826	.848	.837	.856	.843	.849	.803	.791	.797
1234-ES	.981	.955	**.968**	.573	.558	.565	.848	.631	.724	.678	.505	.579
1234-FR	.940	.752	.835	.503	.402	.447	.787	.554	.650	.605	.426	.500

Table 4: This table collects my best results (precision P, recall R, F1 F) for all the tasks and all the languages of this homework. In bold the highest values for each F of the tasks.

References

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

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Dan Jurafsky and James H. Martin. 2022. *Speech and Language Processing (3rd ed. draft)*. stanford-edu.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 716–724, Beijing, China. Coling 2010 Organizing Committee.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ines Montani, Matthew Honnibal, Matthew Honnibal, Sofie Van Landeghem, Adriane Boyd, Henning Peters, Paul O'Leary McCann, Jim Geovedi, Jim O'Regan, Maxim Samsonov, Duygu Altinok, György Orosz, Daniël de Kok, Søren Lind Kristiansen, Lj Miranda, Explosion Bot, Roman, Peter Baumgartner, Leander Fiedler, Richard Hudson, Madeesh Kannan, Grégory Howard, Edward, Wannaphong Phatthiyaphaibun, Yohei Tamura, Sam Bozek, murat, Ryn Daniels, and Flusskind. 2022. explosion/spaCy: v3.4.0: Updated types, speed improvements and pipelines for Croatian.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden. Association for Computational Linguistics.

Riccardo Orlando, Simone Conia, Fabrizio Brignone, Francesco Cecconi, and Roberto Navigli. 2021. AMuSE-WSD: An all-in-one multilingual system for easy Word Sense Disambiguation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 298–307, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Kun Xu, Han Wu, Linfeng Song, Haisong Zhang, Linqi Song, and Dong Yu. 2021. Conversational semantic role labeling.