NLP Final Project

An Exploration of Hybrid Neural Document Summarization Task

Gennaro Petito Alessandro Pecora Vincenzo Marciano'

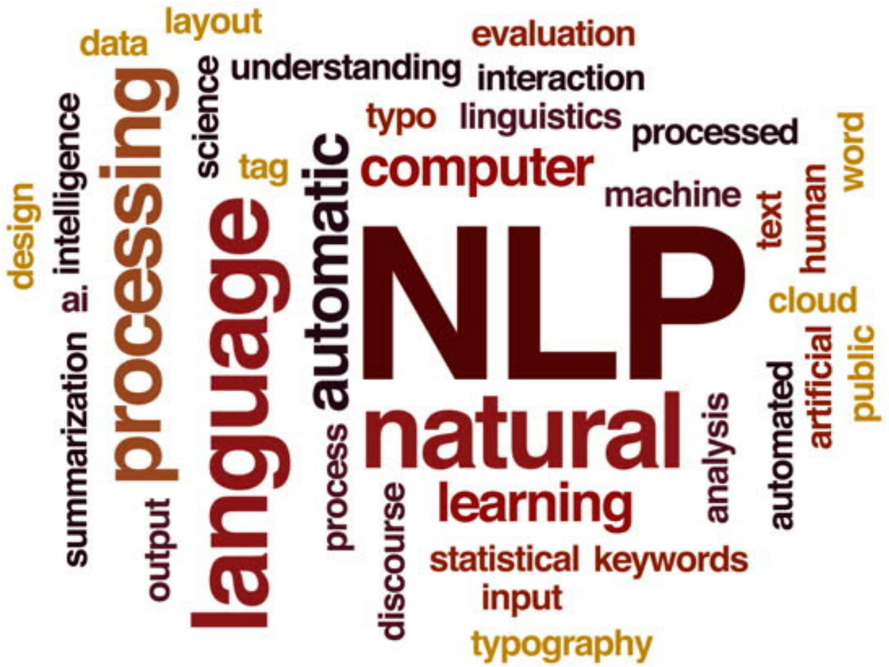
Main Steps

1. Introduction
2. Preprocessing
3. Extractor - Abtractor
4. Reinforcement Learning
5. Results and Conclusion

Introduction

Starting from related works on the **summarization task**, we explore the two main approaches: **extraction** and **abstraction** with an hybrid end-to-end computational graph.

To conduct our experiments we use **two** different **dataset** usefull to explore the two different approaches and the effectiveness of the method on different domain.



Preprocessing

Datasets

WikiLingua

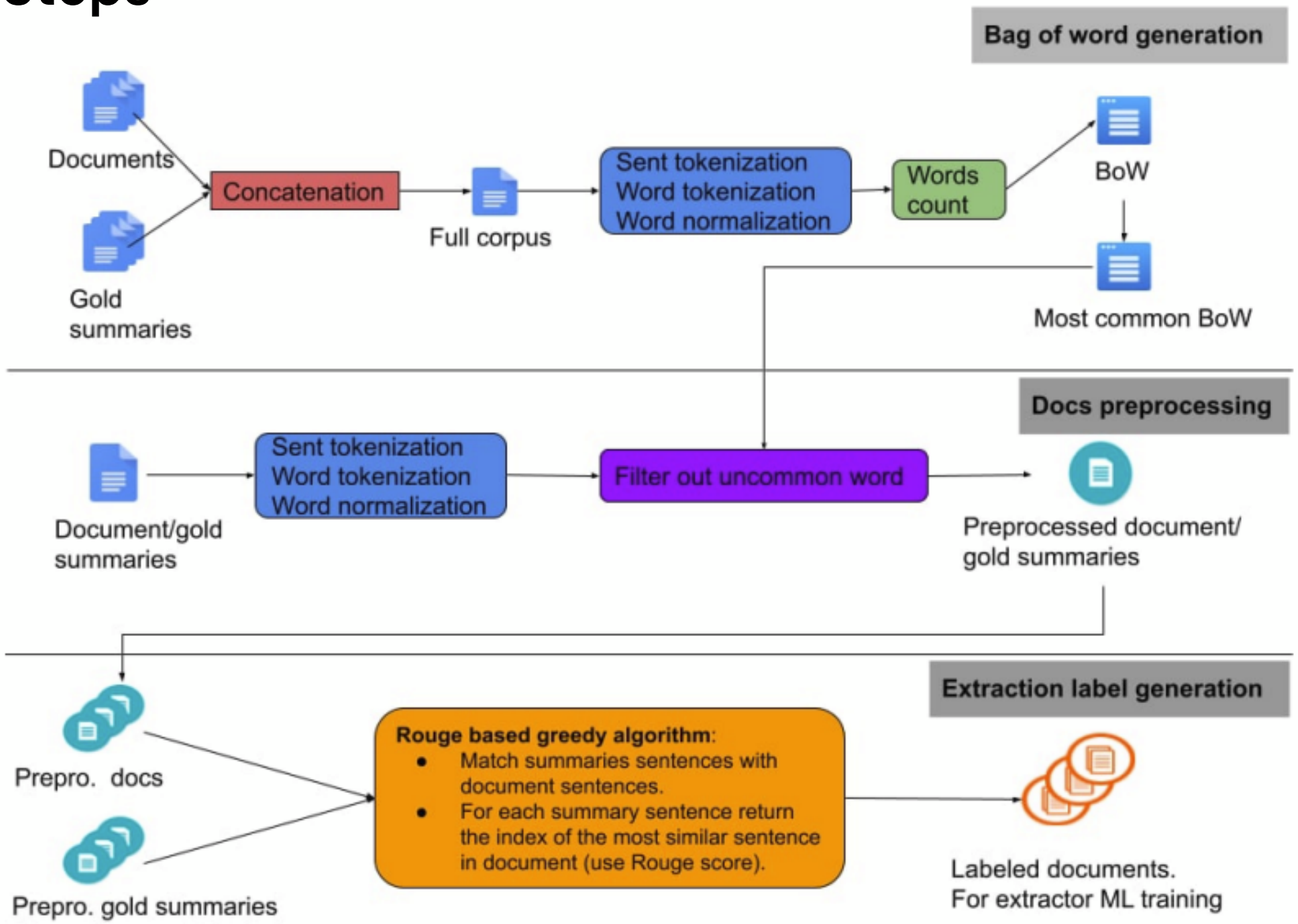
WikiLingua is a cross language and **open domain** dataset extracted from WikiHow, for each document an **abstractive** and coicise summary, in the style of "how-to-do", is provided.
We use only the italian documents for our purpose, in particular we extract 10k document with each corrispondent summary.
Both documents and respective summaries are **short in terms of number of sentence and word.**

Financial Narrative Summarization (FNS-2021)

FNS 2021 is the dataset provided by the challenge which takes its name from it, cotains more than 3000 reports in the **financial domain**, in particular are annual reports produced by UK firms listed on The London Stock Exchange (LSE), for each report an **extractive** summary is provided.
Respect WikiLingua the reports and the summaries have many **more sentences and words**, and this represent the main reason that lead to different results.

Preprocessing

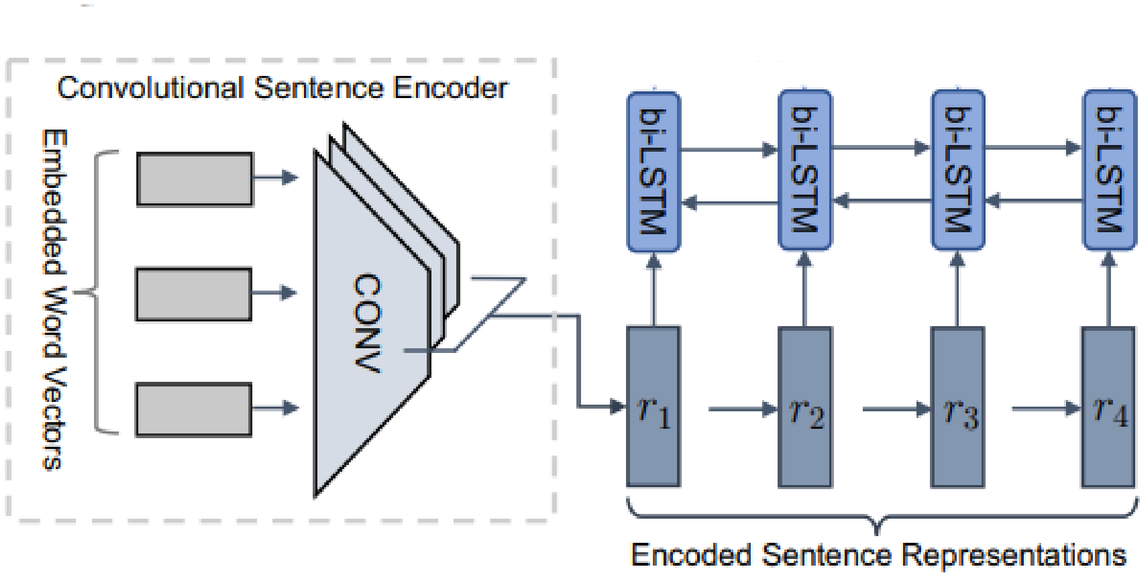
Steps



The data processing pipeline can be divided in **2+1 main phases** and are the same for the abstractive and extractive task except for the BoW generation that for the extractive dataset can be done by using only the concatenation of all documents without golden summary since all tokens in summaries are already present in the document.

After the first 2 preprocessing phase, for the training of the extractor we generate **proxy labels** by matching summaries sentences with documents sentences that achieve the high **ROUGE-L score.**

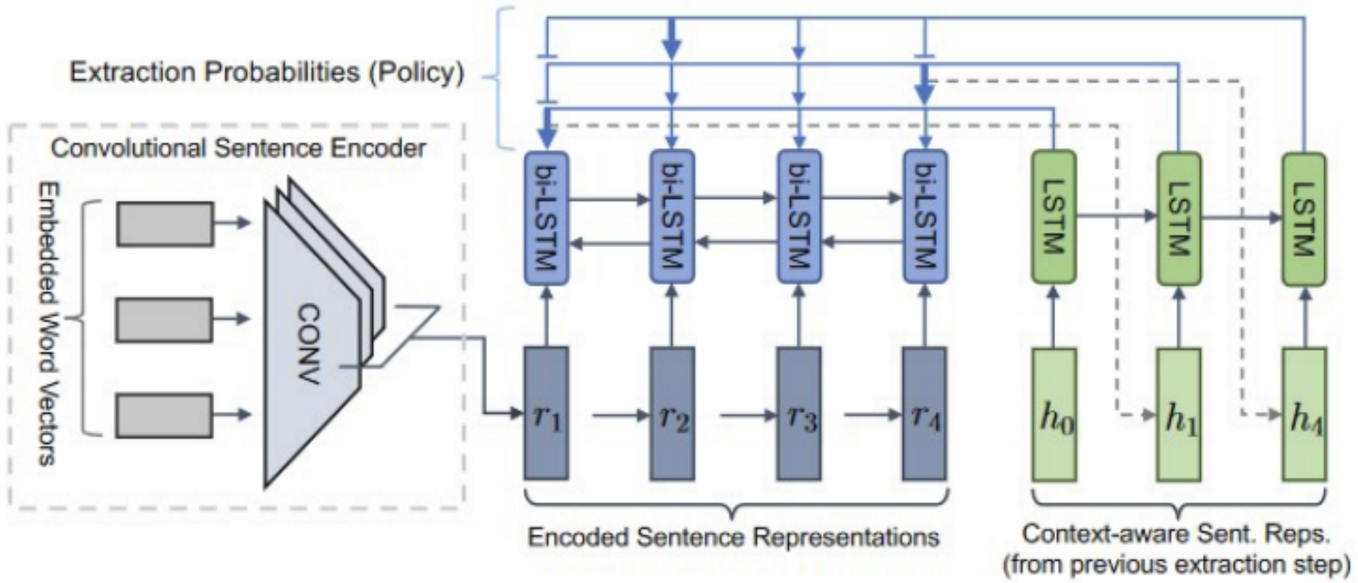
Extractor



ENCODER

1. Convert the Word2Vec embeddings into a sentence representation r
2. Bi-LSTM computes context-aware representation h

Extractor

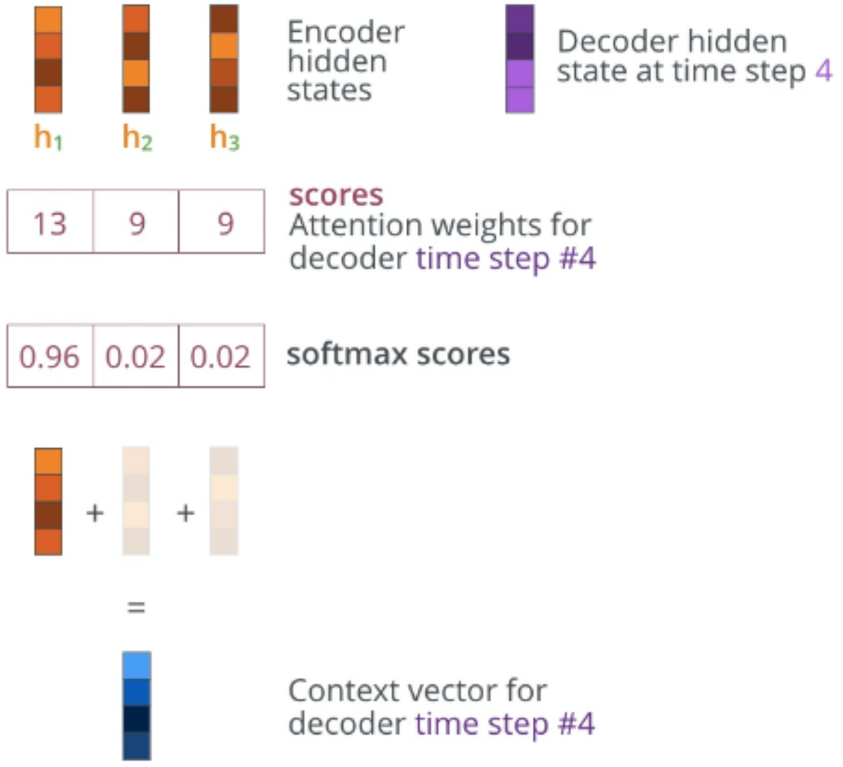


$$a_j^t = v_g^T tanh(W_{g1} h_j + W_{g2} z_t) \qquad \text{scores}$$

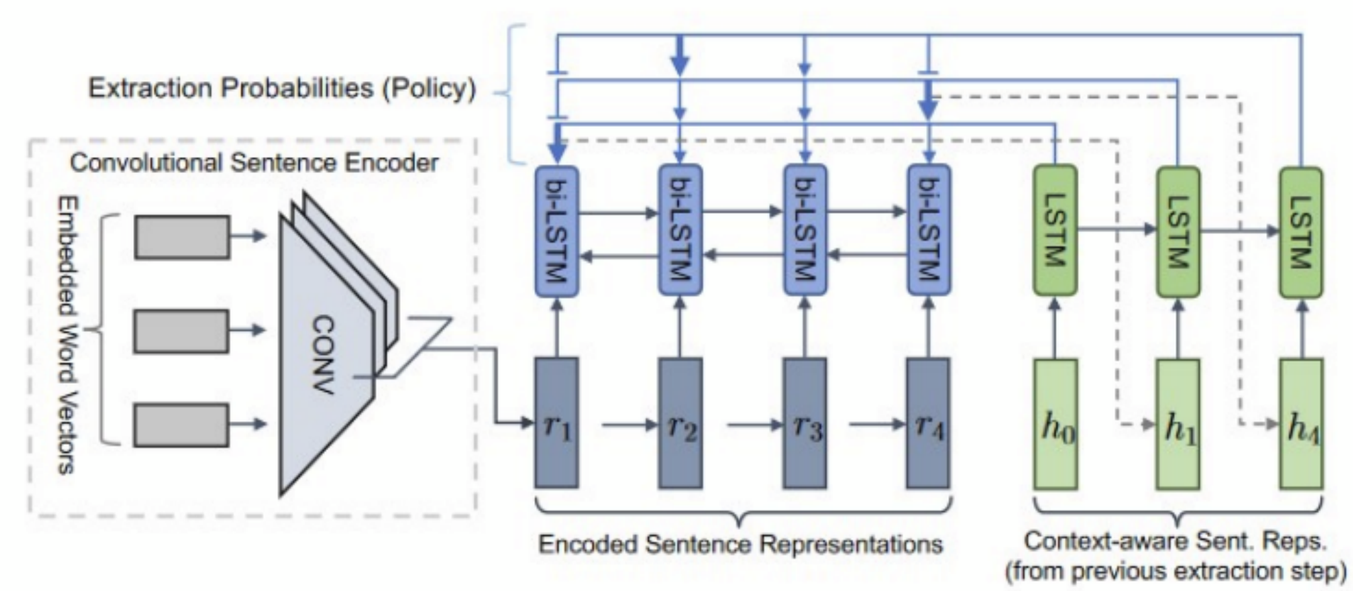
$$\alpha^t = softmax(a^t)$$

$$e_t = \sum_j \alpha_j^t W_{g1} h_j \qquad \text{context vector}$$

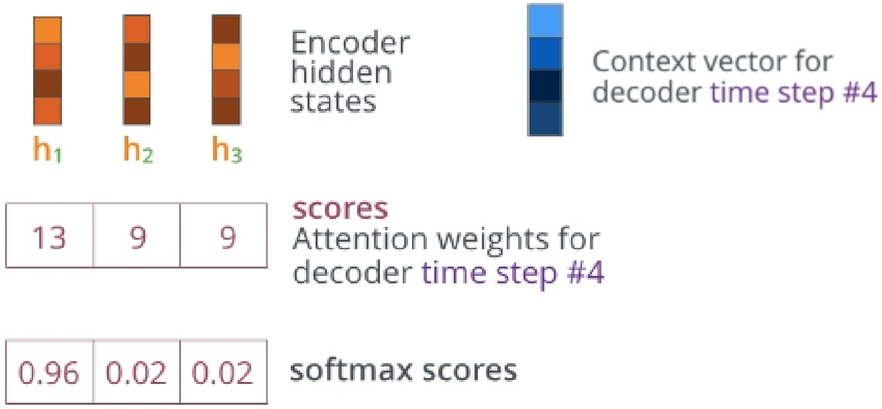
DECODER – HOP 1
Compute context vector
from encoder hidden states



Extractor



DECODER – HOP 2
Compute the extraction probability starting from the context vector

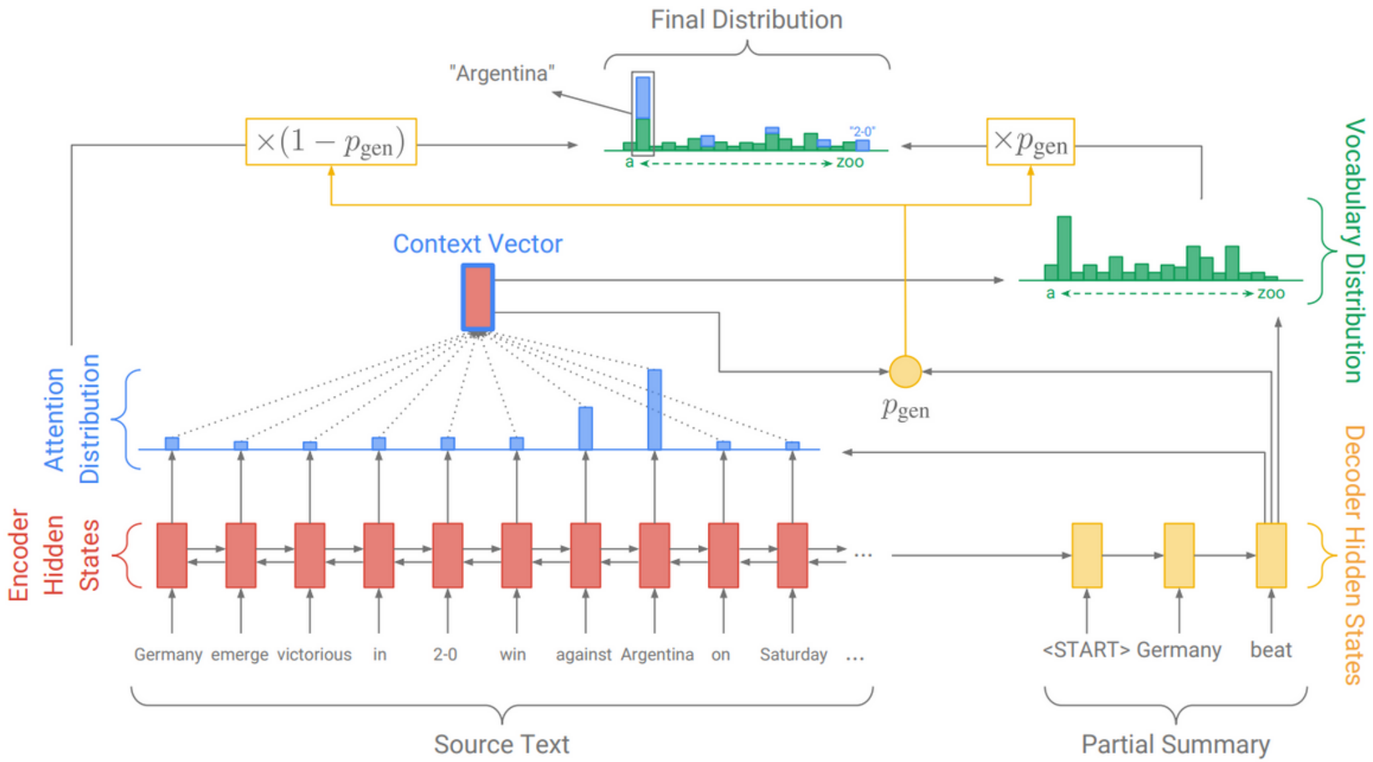


$$u_j^t = v_p^T tanh(W_{p1} h_j + W_{p2} e_t)$$
$$P(j_t | j_1, ..., j_{t-1}) = softmax(u_t)$$

scores
extraction prob.

At this point we either use the logit to calculate the loss for the backward pass (during training) or extract the sentence with the highest softmax score

Abstractor



In the abstractor instead we have to follow 2 routes:

1. Compute the context vector and the attention distribution (that will be used to copy)
2. Compute the distribution over the Word2Vec vocabulary

Finally we sum the two distributions according to the generation probability:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}})$$

- h* = context vector
- s = decoder hidden state
- x = concat input and previous decoder state

Reinforcement Learning – A2C

To mitigate the high variance due to the algorithm, we train a Critic Network, minimizing the MSE LOSS between the baseline produced by the critic and the actual reward of that imputed action. This mechanism is called **A2C (Advantage Actor Critic)**

Definitions

1. The **baselines** are computed as the expected reward for all possible actions in a specific extraction step
2. The **advantage** is the difference between the computed reward and baselines. It's needed to notify the network how "well" it's going.
3. Each **reward** is a combination of different ROUGES, including also the stop-reward score

Reinforcement Learning – A2C

Pseudocode

```
for each article in train step:
   (baselines, indices), probs = agent(article)
   summ = abstractor(indices)
   for each extracted sentence, gold in indices, golds:
      reward = [reward_summ(summ, gold)+stop_reward]
      R = discounted_reward (reward)
      rewards.append(R)
   for each idx, prob, b, r in (indices, probs, baselines, reward):
      advantage = r - b
      tot_advantage += advantage
      norm_adv = advantage / len(indices)
      losses.append(log_prob(prob * norm_adv))
   critic_loss = MSE(baselines, rewards)
```

This approach is based on a **Markov Decision Process**: at each extraction step the agent observes the current state to extract the next document sentence, receiving a reward. Penalization is inferred to "far" states.

Results

Adopted Metrics

1. **ROUGE-1:** to find the common unigrams
2. **ROUGE-2:** to find the common bigrams
3. **ROUGE-L:** to find the largest common substrings

For FNS dataset a subset of **max N sentences** for each document is selected due to the low-memory hardware (empirically **N=200**)

We investigate the **best reward formula** to use, starting from the proposed study among all the proposed ROUGE metrics. Results are then shown in the following tables:

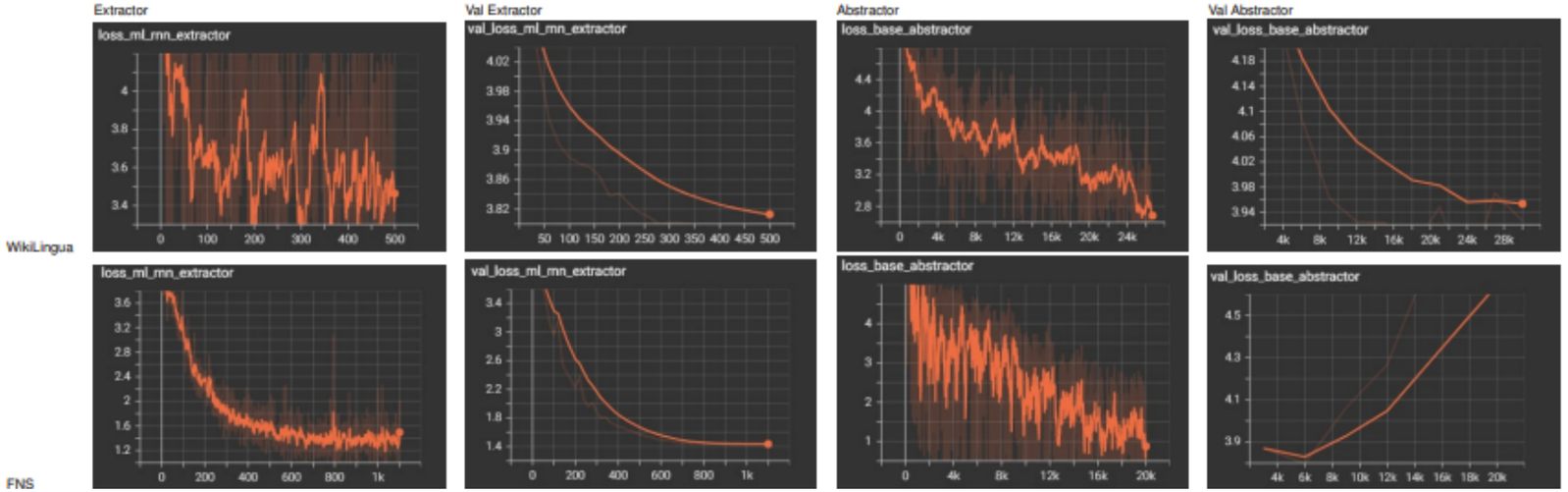
The comparison with the original paper lead us to different results: analyzing that study, we thought that the **authors used a different rouge score** w.r.t. the reward described for the final evaluation.

	WikiLingua ROUGE2	WikiLingua ROUGE-L	WikiLingua Avg-ROUGEs	FNS 21 ROUGE-2	**FNS 21 ROUGE-L**	FNS 21 Avg-ROUGEs
Avg ROUGE-1	**0.0282**	0.0273	0.0255	0.0413	**0.218**	0.0732
Avg ROUGE-2	**0.154**	0.149	0.132	0.139	**0.348**	0.237
Avg ROUGE-L	**0.152**	0.145	0.131	0.134	**0.344**	0.228

Results

Overfitting and Loss

We can see how the **extractor** on FNS performs much better than the extractor in the WikiLingua task, however in both cases **no overfitting** is revealed since we tried several Regularization techniques, eventually adopting a **smaller mono-layer structure** for Wikilingua. For the **abstractor** instead we can see how on the FNS dataset there is **a lot of overfitting for the FNS** that is not present for WikiLingua. We suppose these results are due to the structure of the datasets: in WikiLingua we have a lot more sentences per summary than the FNS.



Conclusion

In this project, we have reported on our solution for the Financial Narrative Summarisation (FNS2021) shared task using actor critic reinforcement learning approach. Although the architecture behaves quite well, the results may be improved a lot more by using **a powerful hardware** that could allow us to use the whole FNS dataset and achieve consistent performances.

Another interesting variation for future works is the integration of **BERT score** for broader investigation of the sentence selection and paraphrasing, and including it as a possible reward for the RL actor-critic architecture.

Moreover, we noticed that the PointerNet in some cases would **not extract any sentences at all** with respect to a certain article must be solved. In our case, we used a greedy approach by selecting the first 5 sentences of the article, if it happened.

Finally, we did some **ablation studies** by including or not the Abstractor for the FNS dataset: although its absence gave us promising results, we decided to not involve this experiment for the training nature of the model that is quite "poor" due to the memory limitations. For this reasons we prevented any kind of generalization about the Abstractor actual usage.

Thank you !