Project report : Rationalizing neural predictions

Ariane Cwiling, Blaise Delattre et Marie-Julie Picas

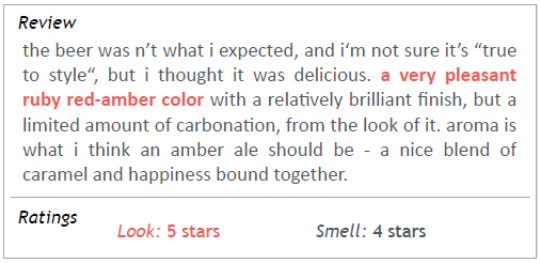
April 2021

1 Introduction

Powerful neural networks have achieved great performance in many machine learning applications, but the problem is that these models often fail to provide good interpretability about why certain predictions were made given the model's input.

To provide some justifications behind the predictions, Lei et al. (2016) propose a model where they learn to extract pieces of input text as justifications, which are called rationales.

They illustrate their model on two applications: multi-aspect sentiment analysis and question retrieval. In this project, we only focus on the first one.



In the example above, if the model predicts five star rating for the color of the beer, it should also identify the sentence "a very pleasant ruby red-amber color" as the rationale. This highlighted sentence is the rationale and justifies the prediction.

"Rationales are simply subsets of the words from the input text that satisfy two key properties. First, the selected words represent short and coherent pieces of text (e.g., phrases) and, second, the selected words must alone suffice for prediction as a substitute of the original text." Lei et al. (2016)

Plan of the project

First, we present the model developed in Lei et al. (2016).

Second, using available code described bellow, we investigate the structure of the model in more details, and study the impact of the hyperparameters.

Then we replace GloVe, Jeffrey Pennington and Manning (2014), with another word embedding method, the BERT model, Jacob Devlin (2018).

We also try different kinds of neural network architectures of the generator. Indeed, we noticed that the article gives great attention to the different types of encoders possible, that's why we wanted to test different kinds of structures for the generator too.

Finally, we open discussions on different topics.

Data and code

We didn't reproduce the original Theano code from Tao Lei, but the Pytorch version of Adam Yala, as we are more familiar with Pytorch. A main difference in the implementation from Adam Yala is that he implemented generator training using the Gumbel Softmax, Eric Jang (2016), instead of using REINFORCE, Junzi Zhang (2020).

We couldn't use the same dataset as in the paper, as it wasn't available. That's why we used the "20newsgroups" dataset from scikit-learn. The classification objective is a bit different. With the beer dataset we aimed at predicting ratings for different aspects of the beer (and the corresponding rationales). With the newsgroups dataset, we want to predict the category of a text (and the corresponding rationales).

2 Model

In this section we gather essential points of the article Lei et al. (2016), as an overview of the model.

The approach combines two components, generator and encoder. The generator specifies a distribution over text fragments, which means that it takes the original document and produces a distribution of the possible rationales. Then, the encoder takes the selected rationales and makes the prediction.

Generator and encoder are trained jointly to minimize a cost function that favors short, concise rationales while enforcing that the rationales alone are sufficient for an accurate prediction.

To sum up, we want the generator to select the most relevant text fragments so that the encoder prediction using only these fragments is close to the prediction with the entire text.

Consider a text x. The rationale for this given sequence can be defined in terms of binary variables $z = \{z_1, \ldots, z_l\}$ where each $z_t \in \{0, 1\}$ indicates whether the word x_t is selected or not. The pair (z, x) is the actual rationale generated (selections, input). The generator actually specifies a probability distribution over the binary selections ($z \sim gen(x) \equiv p(z|x)$).

The generator is guided in two ways during learning, to assure sufficiency, sparsity and coherency in the rationale selection. First, as explained previously, the rationale must be sufficient as a replacement for the input text. We want the target vector (or sentiment) arising from the rationale to be close to the one arising from the original text. In other words, enc(gen(x))(=enc(z,x)) should result in nearly the same target vector as enc(x)(=y). To quantify the difference between these two predictions, we use a sentiment loss function:

$$\mathcal{L}(z, x, y) = \|enc(z, x) - y\|_2^2.$$

Second, the generator should select only a few words and those selections should form phrases (so consecutive words) rather than represent isolated, disconnected words. To that end, we introduce an additional regularizer over the selections, where the first term penalizes the number of selections while the second term discourages transitions (so it encourages continuity of selections):

$$\Omega(z) = \lambda_1 \|z\|_1 + \lambda_2 \sum_t |z_t - z_{t-1}|.$$

The final cost function we want to minimize is the combination of the loss function and the regularizer.

$$\text{cost}(\mathbf{z}, \mathbf{y}) = \underbrace{\text{loss}(\mathbf{z}, \mathbf{y})}_{\substack{\textit{sufficiency} \\ \textit{correct prediction}}} + \underbrace{\lambda_1 |\mathbf{z}|_1}_{\substack{\textit{sparsity} \\ \textit{rationale is short}}} + \underbrace{\lambda_2 \sum_i |\mathbf{z}_i - \mathbf{z}_{i-1}|}_{\substack{\textit{coherency} \\ \textit{continuous selection}}}$$

3 Hyperparameters

As announced in the plan of the project, we studied the impact of the hyperparameters of the model as a first approach. There is the temperature parameter τ of the Gumble-Softmax distribution, Eric Jang (2016). For our text generation, we need to sample discrete data. The authors use the Gumbel-Softmax distribution to approximate the sampling process of discrete data from a categorical distribution, which is not differentiable. The parameter τ is the temperature parameter that controls how closely the samples approximate discrete vectors. Samples from Gumbel-Softmax distributions are identical to samples from categorical distributions as tends τ to 0. Whereas when τ tends to infinity, the sample vectors become uniform.

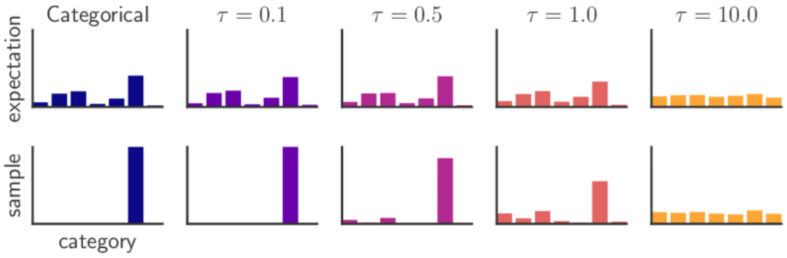


Figure 1: Illustration of different values of the temperature parameter τ

There is also a lot of arguments defined in the framework such as the number of layers, the embedding dimension or the batch size for example. Most of these arguments are already optimized by the authors, so we tried to change the hyperparameters λ_1 and λ_2 from our cost function to observe the impact on the outcome. We used 20 epochs so that the code would not take too long to run while using a reasonable number of epochs. The code had $\lambda_1 = 0.001$ and $\lambda_2 = 0$ as initial values. We always took the value $\lambda_2 = 2\lambda_1$ and tested $\lambda_1 \in \{0.008, 0.01, 0.012\}$. The higher λ_1 is, the fewer words are selected, as we can see below, and the loss is growing.

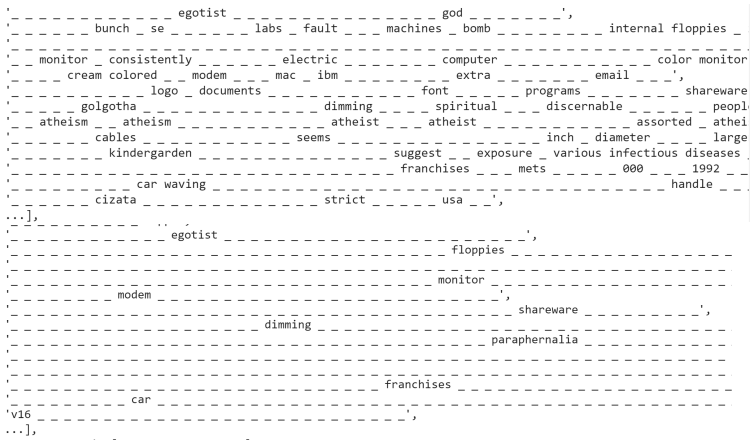


Figure 2: Rationales with $\lambda_1 = 0.001$ and $\lambda_2 = 0$ (top) and $\lambda_1 = 0.012$ and $\lambda_2 = 2\lambda_1$ (bottom)

4 Word embedding

We wondered if the rationales are selected because of some specific words or if the model understands phrases, linguistically speaking. Consider the examples of the original paper given bellow. It seems that selections are coherent phrases (expressions on a same topic, linked with connectors), which is why we wondered if the model would still be performing if we only kept "meaningful" words.

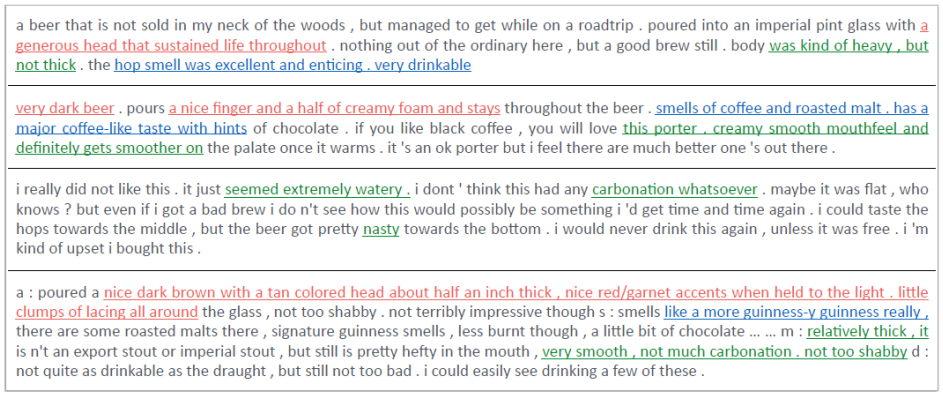


Figure 3: Examples of extracted rationales indicating the sentiments of various aspects. The extracted texts for appearance, smell and palate are shown in red, blue and green color respectively. The last example is shortened for space.

We considered different approaches:

- Test the robustness of the model by filtering the imputed texts, removing linking words and connectors to keep only the most "meaningful" words;
- Using model probing methods to study linguistic properties of the representation, see Hewitt (2019), Faldu (2020).

However, these properties seem to be related mostly to the way word embedding are produced. So we decided to use existing word embedding structures to test the language knowledge of the model. In Adam Yala's code, the word embeddings architecture is GloVe. GloVe is a context independent model for word embedding, so we decided to study another model for *contextualized* word embedding like BERT, to see if we got better results.

BERT implementation

The first difficulty was to implement BERT properly. The original architecture of the code is based on GloVe, which is a context independent word embedding method. It is sufficient to load the dictionary of word embedding before starting reading texts. Then, each text is represented through a vector of indices, each index mapping a word to its embedding in the dictionary. However, with a context dependent word embedding method like BERT, we can not use a dictionary. Indeed, we can represent the same word with different embedding depending on the context of the sentence. We have changed the structure of the code so that word embedding are produced using the BERT model when the text is passed through the network.

Results (or not ?)

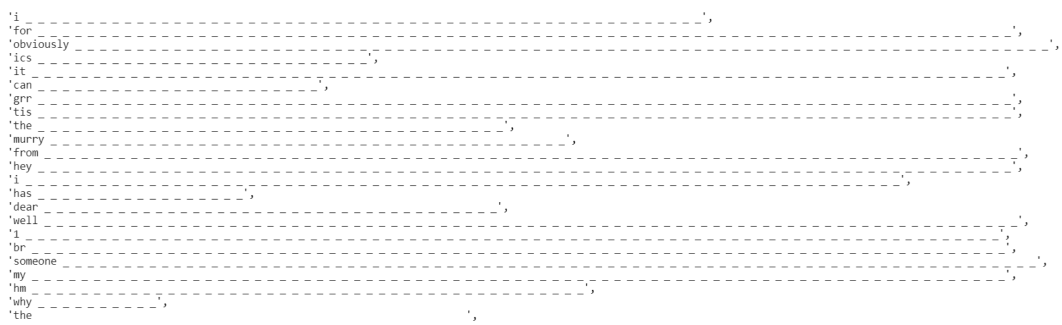
We tested the implementation for different numbers of epochs (4 or 20 epochs) and of texts:

- 1000 texts in train, 1000 texts in dev, 1000 texts in test;
- 9051 texts in train, 2263 texts in dev, 7532 texts in test (the whole dataset);

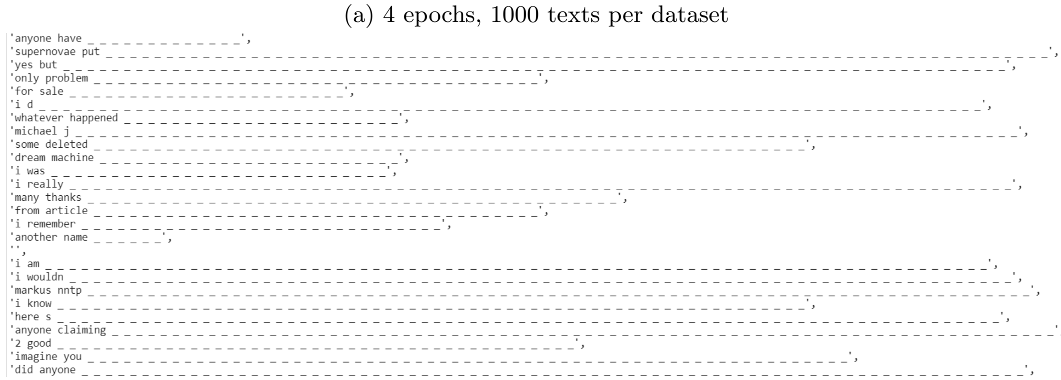
Figure 3 gives illustrations of the results. It seems like the implementation using BERT only selects the first words in each text, no matter if we increase the number of texts or of epochs.

It is difficult to know the reason for this result, however we emitted several hypotheses:

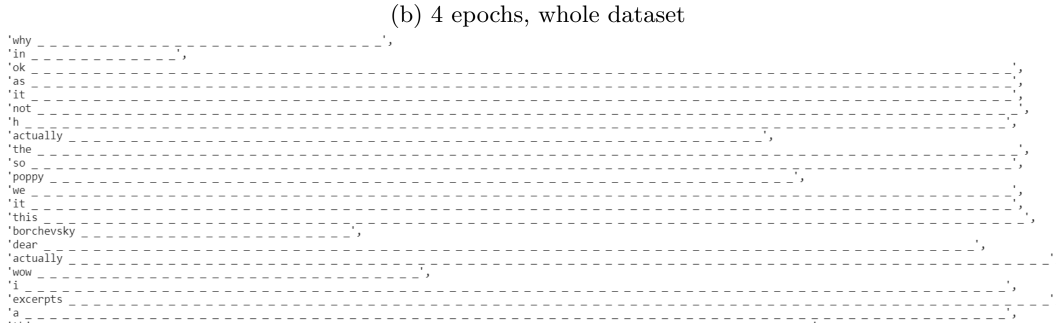
- Error(s) in the code and/or theoretical misunderstanding(s);
- Specificity(ies) of the BERT model, for example BERT provides its own tokenizer, and words can be split into smaller subwords and characters, like ”embeddings” into ['em', '##bed', '##ding', '##s'];
- Or it could be an illustration that the generator can't understand words in their context, and can only learn in a context independent way that some words appear more frequently in some categories than in others.



(a) 4 epochs, 1000 texts per dataset



(b) 4 epochs, whole dataset



(c) 20 epochs, 1000 texts per dataset

Figure 3: Some rationales obtained with different parameters using the BERT model.

5 Architecture

We start from an implementation from Yala (2019).

5.1 Generator

For the benchmark generator we used a non static pretrained embedding layer from GloVe plugged in a CNN Kim (2014), as illustrated in Figure 4, which extracts the local features of a words sequence. However we do not perform max pooling over time as we do not wish to reduce the dimension of the input in order to have a mask on the text input of the same dimension.

This operation is followed by a differentiable sampling performed with Gumbel-Softmax Eric Jang (2016). It returns a vector of probabilities which can be viewed as the probability for each word to be selected and passed to the encoder. This probability vector can be hard pass as it is for training or threshold to binary value at testing time.
We took 300 filters for convolution 100 for each kernel size $3, 4, 5$.

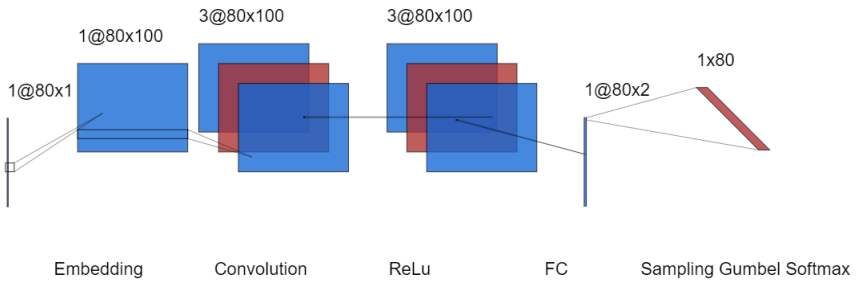


Figure 4: One layer convolutional architecture for generator for an input of length 80 with embedding dimension of 100.

Instead of a CNN, we implemented and test a recurrent architecture for the generator a bidirectional long short-term memory (LSTM) RNN as in Haşim Sak (2014) with one layer and a hidden dimension of 300, so that the number of parameters is the same as in the previous generator model. Except this we kept everything the same.

5.2 Encoder

The encoder uses a CNN architecture Kim (2014) for classification, as illustrates in Figure 5, and it receives as an input the output of the generator. The latter is a mask which is applied to the original text to determine which words are being used to make the classification.

A max pooling over time Ronan Collobert (2011) is performed to capture the most important feature and it reduces the input dimension.

6 Training

Training is done by applying the Adam Diederik P. Kingma (2014) solver for SGD with a batch-size of 64, dropout of 0.5, learning rate of 0.0001, $\beta_1 = 0.9, \beta_2 = 0.999$ for 50 epochs, with $\lambda_1 = 0.01$ and $\lambda_2 = 0.01$.

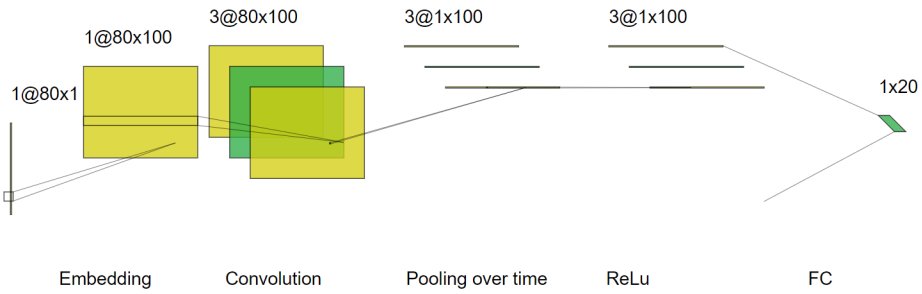


Figure 5: One layer convolutional architecture for encoder for an input of length 80 with embedding dimension of 100, with 3 filters and 20 classes

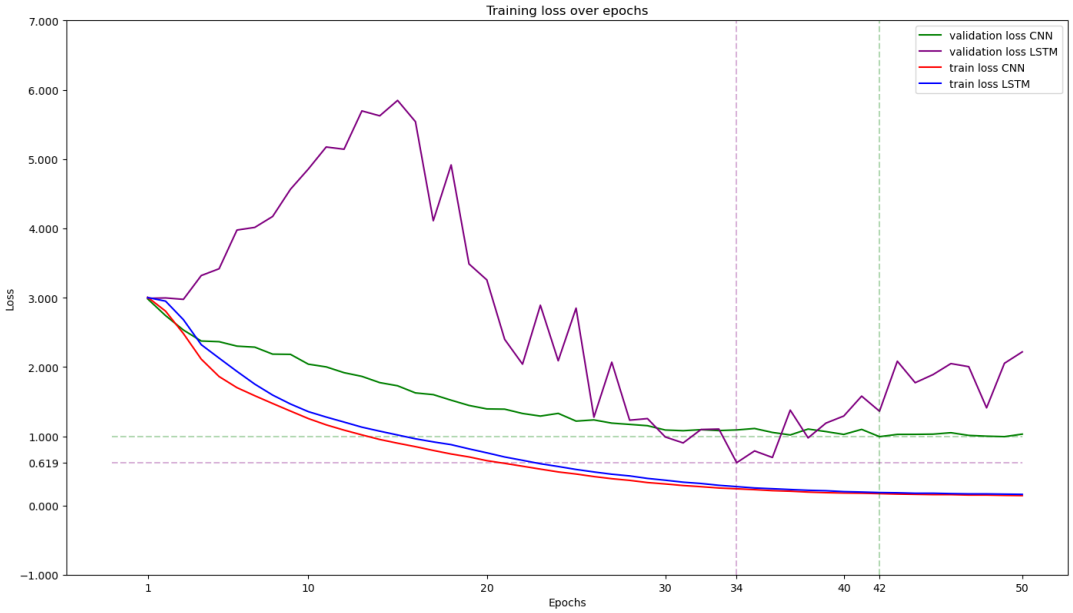


Figure 6: Train and validation losses over epochs for CNN-encoder CNN-generator and CNN-encoder LSTM-generaor.

We see in Figure 6, that the training with LSTM-generator adds more variance in comparison with CNN-generator one.
We get best validation loss at epoch 34 for the LSTM-generator. After this point the loss does not diminish, we tested smaller rates but could not have convergence.

6.1 Results

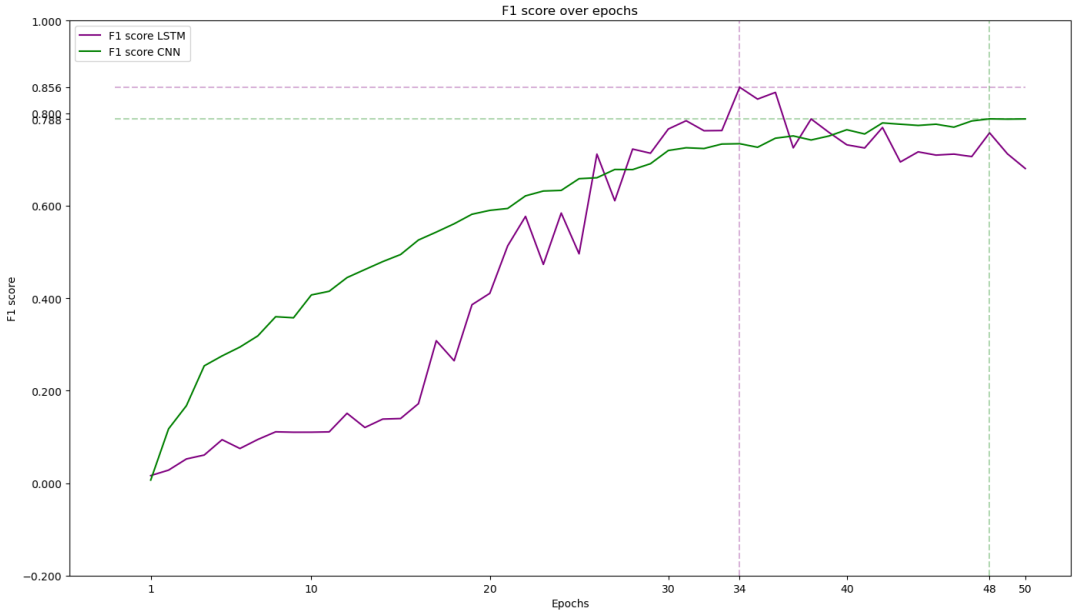


Figure 7: F1 score over epochs for CNN-encoder CNN-generator and CNN-encoder LSTM-generaor.

As seen, in Figure 7, we obtain better F1 score and validation loss at epoch 34 for CNN-encoder and LSTM-generator. The maximum F1 score (0.856) for the model CNN-encoder and LSTM-generator outperforms the maximum F1 score (0.788) for the CNN-encoder and CNN-generator model.

7 Discussion

Variance reduction

To quote Lei et al. (2016), "we could also apply variance reduction techniques to increase stability of stochastic training". They mention variance reduction techniques from Weaver and Tao (2001), Mnih et al. (2014), Ba et al. (2015) and Xu et al. (2015). Our goal was initially to add one of their variance reduction techniques to the model. However, they are all based on the REINFORCE algorithm, which was replaced by the Gumbel Softmax in Adam Yala's code.

Medical imaging

A possible extension could be in medicine. If a model is trying to help doctors predict whether or not a patient has a high risk of recurring cancer given his medical record, doctors would need and want evidence to trust the model. For example in medical imaging, the model would help localize the problematic region of an image (MRI, ultrasound...) which suggests that the patient could have a disease. Unfortunately, the limited time given for the project didn't allow us to adapt the code to images.

8 Conclusion

The study of hyperparameters helped us grasp ans become handy with the code. We illustrated the impact of the hyperparameters of the cost function on the rationales given in the output. Augmenting the values of the parameters implies the selection of fewer words and the increase of the loss.

Results about BERT embeddings are hard to interpret, as the inconclusive results could be linked to a misunderstanding of the model or a BERT specificity. It is then difficult to conclude about the language knowledge of the model that we wanted to quantify.

Considering a CNN-encoder LSTM-generator model instead of a full CNN encoder generator model improved the F1-score on the dataset. Most likely a full RCNN Tao Lei (2015) model would have give better results by gathering local and recurrent features.

This project should be seen more as an introduction to further work on the subject than as a fully accomplished research paper. However, the work done (i.e. read papers and codes related to the subject, adapt our own code, study results) allowed us to learn a lot, both theoretically about CNN, RCNN, LSTM, transformers, the BERT model, rationalizing a network, and practically about the way to code all of these architectures. Beyond the scope of the subject, the project provided us an insight into the research process, which was really interesting and instructive.

References

Ba, Jimmy, Volodymyr Mnih, and Koray Kavukcuoglu (2015). "Multiple object recognition with visual attention". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma, Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980*.

Eric Jang Shixiang Gu, Ben Poole (2016). "Categorical Reparameterization with Gumbel-Softmax". In: *arXiv preprint arXiv:1611.01144*.

Faldu, Keyur (2020). *Discovering the Encoded Linguistic Knowledge in NLP models*. URL: `https://towardsdatascience.com/encoded-linguistic-knowledge-in-nlp-models-b9558ba90943`.

Haşim Sak Andrew Senior, Françoise Beaufays (2014). "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition". In: *arXiv preprint arXiv:1402.1128*.

Hewitt, John (2019). *Designing and Interpreting Probes*. URL: `https://nlp.stanford.edu/~johnhew/interpreting-probes.html`.

Jacob Devlin Ming-Wei Chang, Kenton Lee Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805*.

Jeffrey Pennington, Richard Socher and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *https://nlp.stanford.edu/pubs/glove.pdf*.

Junzi Zhang Jongho Kim, Brendan O'Donoghue Stephen Boyd (2020). "Sample Efficient Reinforcement Learning with REINFORCE". In: *arXiv preprint arXiv:2010.11364*.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *arXiv preprint arXiv:1408.5882*.

Lei, Tao, Regina Barzilay, and Tommi Jaakkola (2016). "Rationalizing Neural Predictions". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 107–117.

Mnih, Volodymyr, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu (2014). "Recurrent Models of Visual Attention". In: *Advances in Neural Information Processing Systems (NIPS)*.

Ronan Collobert Jason Weston, Leon Bottou Michael Karlen Koray Kavukcuoglu Pavel Kuksa (2011). "Natural Language Processing (almost) from Scratch". In: *arXiv preprint arXiv:1103.0398*.

Tao Lei Hrishikesh Joshi, Regina Barzilay Tommi Jaakkola Katerina Tymoshenko Alessandro Moschitti Lluis Marquez (2015). "Semi-supervised Question Retrieval with Gated Convolutions". In: *arXiv preprint arXiv:1512.05726*.

Weaver, Lex and Nigel Tao (2001). "The optimal reward baseline for gradient-based reinforcement learning". In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). "Show, attend and tell: Neural image caption generation with visual attention". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Yala, Adam (2019). *Text NN*. URL: `https://github.com/yala/text_nn`.