Measuring the effect of pretraining a hierarchical autoencoder for binary intent classification on data with similar structure

Emilien Jemelen†
ENSAE - IP Paris
emilien.jemelen@ensae.fr

Christophe Morau†
ENSAE - IP Paris
christophe.morau@ensae.fr

Abstract

In this article, we examine the potential of the hierarchical autoencoder presented in (Chapuis et al., 2020) for classifying movie reviews. Our objective is to determine how effective this model, which is tailored for processing conversations and utterances, is when applied to tasks with different hierarchical structures.

Our findings indicate that pretraining the hierarchical model on a dataset of movie subtitles unexpectedly fails to enhance prediction accuracy and performs worse than the same model without any pretraining. This suggests that the structure learned from subtitle conversations may not always generalize well to written paragraphs. These results prompt a discussion on the requirements that both data and models should meet to qualify for an accuracy boost through hierarchical pretraining.

In conclusion, our study provides novel insights into the potential of hierarchical autoencoders for natural language processing tasks and emphasizes the importance of comprehending the specific characteristics of data to optimize model performance.

1 Introduction

Sentiment analysis, the process of determining the emotional tone or attitude expressed in a piece of text, is becoming increasingly important in various fields such as marketing, social media (Witon* et al., 2018), and customer service . The sentiment of a sentence can provide valuable insights into the opinions, preferences, and needs of individuals, groups, or communities.

In the context of social media, for example, sentiment analysis can be used to track the public's response to a product launch, political event, or social issue. In customer service, sentiment analysis can help companies identify and address customer complaints or feedback in real-time (Jalalzai* et al., 2020; Colombo* et al., 2019), thus improving customer satisfaction and loyalty. In addition, sentiment analysis (Pichler et al., 2022; Colombo et al., 2022, 2021) can be used to filter out inappropriate or offensive content from online platforms and detect potential threats or risks.

Overall, understanding the sentiment of a sentence is crucial in today's digital world, where large amounts of text are generated and consumed every day. Sentiment analysis provides a powerful tool for extracting valuable insights from text data and improving various NLP applications, ultimately benefiting individuals, businesses, and society as a whole (Colombo, 2021).

In this paper, we studied an extension of the hierarchical autoencoder proposed by (Chapuis et al., 2020) to a different type of problems: predicting if a movie review is either positive or negative. Our idea was that the hierarchical encoder, which captures signals both from the utterance and dialog levels, might also be able to model the interactions between the sentences of a movie review.

2 Related works

We relied heavily on the methodology introduced in (Chapuis et al., 2020), in which the authors proposed a neural net approach for sentiment recognition based on the pretraining of a hierarchical autoencoder. By masking pieces of utterances and - at a higher level - utterances themselves in the context of dialogs, the pretraining allows the autoencoder to get a deep understanding of the language structure itself before proceeding to any specific posterior classification task. The model is composed of a first layer of

†With equal contribution

several transformers responsible to process the utterances, a second layer responsible to process the whole dialog through another transformer and a decoder. The hierarchical autoencoder was trained using MLM loss and GAP loss. After the pretraining, the encoder was reused to finetune a classification task on the identification of dialog acts and of sentiments and emotions in dialogs.

We were also inspired by the works of (Garcia et al., 2019). Considering that opinions have inherently a hierarchical structure, they proposed a hierarchical model to classify opinions expressed through videos. In such a model, each sentence contributes to the final score which determines the classification. Our aim was then to study how well the hierarchical model described in (Chapuis et al., 2020) would fare in a similar context : identifying if written movie reviews are either positive or negative (*i.e.* the viewer liked or disliked the movie).

3 Method

3.1 Pretraining

Since we were trying to assess how well the hierarchical autoenconder proposed by (Chapuis et al., 2020) could be generalized to another classification problem, we mostly reused its exact same structure in our project. The first layer was composed of several transformers, each one of them being responsible to process one utterance. The output of these transformers were then concatenated and fed into another transfomer, in charge of processing information at the dialog level. Following the methogology of the original paper, a dialog was defined as a sequence of utterances with less than 6 seconds between them. Finally, a transformer decoder was in charge of retrieving the original information fed into the encoder for a given utterance.

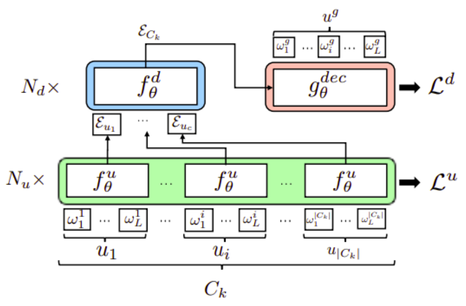
$\mathcal{E}_{C_k}$
u^g
w_1^g ... w_i^g ... w_L^g
$N_d\times$
f_θ^d
g_θ^{dec}
$\mathcal{L}^d$
$\mathcal{E}_{u_1}$ ... $\mathcal{E}_{u_c}$
$N_u\times$
f_θ^u ... f_θ^u ... f_θ^u
$\mathcal{L}^u$
w_1^1 ... w_L^1 ... w_1^i ... w_L^i ... $w_1^{|C_k|}$ ... $w_L^{|C_k|}$
u_1
u_i
$u_{|C_k|}$
C_k

Fig n°1, taken from (Chapuis et al., 2020) : general structure of the proposed hierarchical dialog encoder, with a decoder: f_θ^u, f_θ^d and the sequence label decoder g_θ^{dec} are colored respectively in green, blue and red

To pretrain the model, we chose to focus entirely on the MLM loss. The Masked Language Model (MLM) loss, from (Devlin et al., 2018), (Liu et al., 2019), and (Lan et al., 2019) is an objective function that relies on the maximization of the probabilities to retrieve successfully the true value of some tokens that were masked. For a given utterance u_i in which the set of tokens $\mathcal{M}$ were masked:

$$\mathcal{L}^u(\theta, u_i, \tilde{u_i}) = \mathbf{E}[\sum_{t\in\mathcal{M}} \log(p_\theta(w_t|\tilde{u_i}))]$$

where w_t is the original token before it was masked as t and $\tilde{u_i}$ is the corrupted utterance. As in (Chapuis et al., 2020), we generalize the MLM loss to also apply masks at the utterance level instead of the tokens level in order to better train our hierarchical model. Considering an utterance S that we fully mask, our definition of the generalized MLM was:

$$\mathcal{L}^C(\theta, C, \tilde{C}) = \mathbf{E}[\sum_{t\in S} \log(p_\theta(w_t|\tilde{C}))]$$

with $\tilde{C}$ the corrupted corpus. Finally, given a corrupted version of the corpus $\tilde{C}^t$ randomly masked at the token level and a corrupted version of the corpus $\tilde{C}^S$ masked at the utterance level, our objective function is:

$$\mathcal{L}^{obj}(\theta, C) = \lambda_1 \sum_{\tilde{u}\in\tilde{C}^t} \mathcal{L}^u(\theta, u, \tilde{u}) + \lambda_2 \mathcal{L}^C(\theta, C, \tilde{C}^S)$$

with λ_1, λ_2 convex weights.

3.2 Finetuning

The hierarchical autoencoder, once pretrained, should be able to understand the structure of a text not only at the token level (a token can be predicted based on its context) but also at the utterance level. Considering that predicting a masked utterance from a time-split sequence of OpenSubtitles utterances could be similar to predicting a punctuation-split movie review, we expected this to be helpful to classify movie reviews.

Instead of relying on the presence and contribution of some key tokens (*e.g.* "bad", "good") to make a prediction, a finetuned model using the pretrained architecture would better understand opinions at the sentence level (*e.g.* "The movie was a bit disappointing" / "But overall it was still fun") and use this information to better understand the review.

The information about the structure of a text and the way utterances interact with each other was learnt by the two layers of transformers of our pretrained model : the hierarchical encoding part. The decoding part which had been used to pretrain the model could then be dropped. In our architecture, the output of the hierarchical encoder was of dimension $W \times D$, with W the total number of tokens of the corpus and D the embedding dimension. Since we wanted to measure the contribution of each utterance, we first reallocated each token to its utterance in order to have a tensor of dimension $S \times D'$ with S the total number of sentences in the review and $D' = D * P$ with P the padded size of each sentence.

We then added one last layer with 2 output neurons on top of our pretrained model, plus a softmax was applied to the outputs of the model to have the probabilites of belonging to each class (positive / negative review). The final output is of dimension $S \times 2$ and each line can be interpreted as the contribution of a sentence to the decision: with some level of confidence (the probabilities), the columns express whether the review will be negative or positive according to each sentence. To make a final decision on the opinion of the review, we summed the probabilities (to take into account all the different "votes") and took the class with the maximum value (the most popular vote).

That last layer is the one that is finetuned. To train the model, we used the cross-entropy loss on the (normalized) sum of probabilities that is used to make the final decision.

4 Experiments Protocol

All the code implementation is available with open access on our Google Colab webpage[1].

4.1 Data preprocessing for the pretraining

After loading one half of the French OpenSubtitles database[2] (for a total of 1.2 GB of data), we followed the preprocessing steps from (Chapuis et al., 2020) and associated two successive utterances in a common conversation when the time between them was less or equal than 6 seconds.

For the OpenSubtitles conversations to be treated batch-wise by a Dataloader module (which is the only way we found to do the pretraining without reaching the limits of the Colab environment at our disposal), we had to proceed to an additional padding step both at the sentence and at the dialog levels. Given the distribution of the sizes of tokenized sentences and of conversations, a limit of 30 tokens per sentence and a limit of 10 sentences per dialog seemed reasonable.

Since embeddings are required as inputs of the model, we first built a tokenizer based on the vocabulary of the OpenSubtitles utterances. In a second time, we contemplated using Word2Vec embedding module to turn tokens into embeddings of a certain size (100 in our case, due to limited computation capacity). Yet, Word2Vec seemed to struggle with punctuation marks, which would be important in our case, so a custom FastText embedding model trained on our OpenSubstitles utterances was prefered instead.

4.2 Protocol of the pretraining

The pretraining of the autoencoder was done on one third of the preprocessed OpenSubtitles data using a stochastic gradient descent on all the parameters. No particular adjustment was required (gradients clipping, for instance) as the learning phase showed a clear and smooth decrease of the batch-wise MLM loss. Nonetheless, each pretraining epoch would require a very high computation capacity, so only 3 epochs could be conducted entirely.

Epoch	**Cumul. MLM loss**
1	**119 397**
2	**52 462**
3	**34 029**
4 (estimated)	Estimation based on 5% of train set : **25 180**

Fig. n°2 : per-epoch MLM losses

Remarks on *Fig. n°2* :

We had to stop the pretraining long before reaching the flat part of the learning curve.

The gradient descent was performed on the parameters of the autoencoder, so there is no notion

[1] https://colab.research.google.com/drive/1riDtBYltVsyup8Vr6-hCATJvPJNWaCmR?usp=sharing

[2] https://opus.nlpl.eu/OpenSubtitles-alt-v2018.php

of classification so far. To be more precise, we made the hypothesis that the decrease of the MLM loss (*i.e.* the improvement of the model parameters in terms of prediction of the masked elements of the OpenSubtitles training corpus) would result later in a more accurate classification layer. This assumption will be examined later in this paper.

4.3 Protocol of the training for classification of movie reviews

The data used for training and testing the classification layer of the model were taken from the Allocine movie reviews available in the Huggingface database[3].

The training of the classification layer was performed by stochastic gradient descent using all the reviews in the Allocine train set (for a total of 160,000 reviews and their labels : ”positive” or ”negative”) and the cross entropy loss function from Pytorch.

Conversely to the pretraining gradient descent, the training of the classification layer quickly appeared to be problematic, due to the explosion of oscillating gradients (vectors of higher and higher norms with nearly exact opposite values). A reason for this explosion could have been that the loss was stuck at a local yet not optimal mode.

After trying - without much improvement - to fine-tune the learning rate and momentum parameters of the descent, plus trying another method of parameters optimization (*Adam* from Pytorch), a clear improvement came from limiting the norm of the gradients, a method also known as ”clipping”. Different values were tested, and a clipping of the gradients norms at 1 seemed best, allowing the loss to decrease slowly but consistently through epochs.

[3]https://huggingface.co/datasets/allocine

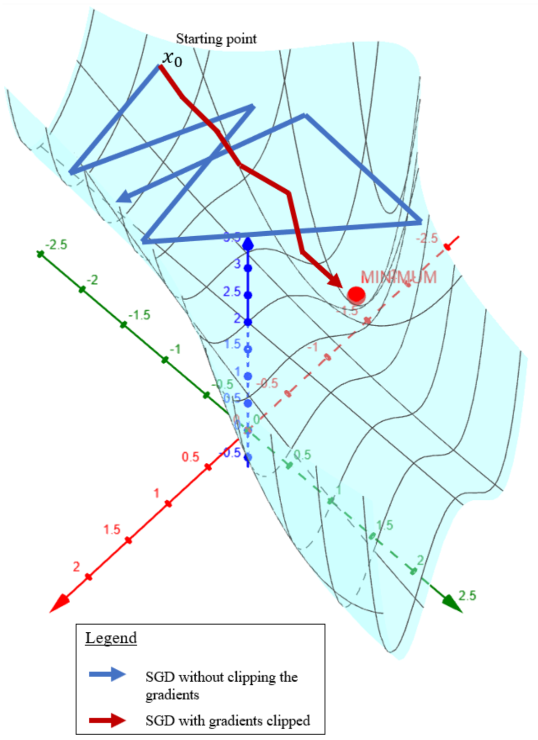
Starting point
x_0
MINIMUM
Legend
SGD without clipping the gradients
SGD with gradients clipped

Fig. n°3 : Geogebra illustration of the interest of clipping the gradients in the case of narrow slopes region

After 30 training epochs, it seemed that we had reached a flatter part of the learning curve in terms of cross-entropy. The losses through epochs are displayed below :

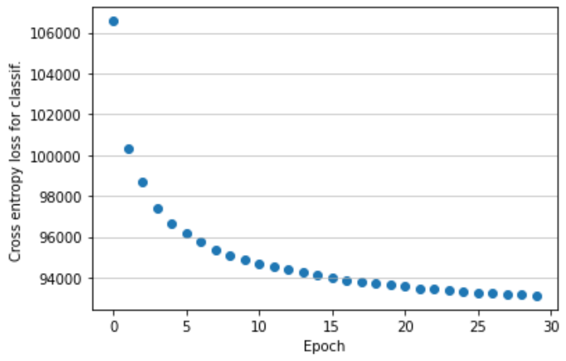

Fig. n°4 : evolution of the per-epoch cross-entropy during the training of the classification layer

Remark on *Fig. n°4* : the per-epoch cross-entropy loss went down from around 107,000 to approximately 93,000. Even on the train set, it seems that the classification layer training results are quite limited.

5 Results

To assess the accuracy of our pretrained model for the classification of Allocine movie reviews, we drew randomly 50 bootstrapped sets of 20 reviews from the whole test dataset (which contained 20,000 reviews, so any statistic computed on the base of the 10,000 bootstrapped reviews should represent the whole test data) and computed the average accuracy of review classification per bootstrapped set. Below is the distribution of the accuracy over the bootstrapped sets of reviews :

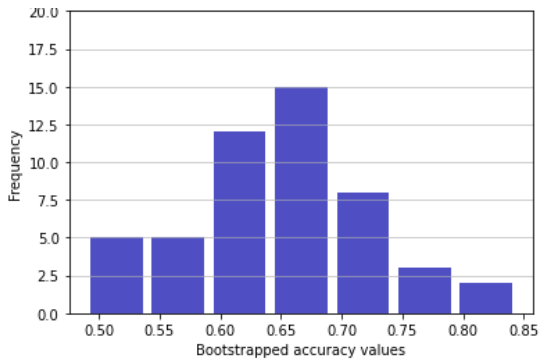

Fig. n°5 : distribution of the classification accuracies over bootstrapped sets of reviews

Remarks on *Fig. n°5* :
The classification model does not happen to perform worse than randomness.
The global average accuracy of prediction over the bootstrapped sets of movie reviews is 64%.

6 Discussion/Conclusion

After a computationally greedy pretraining of our hierarchical autoencoder and a training of a classification layer, we reached an accuracy of movie reviews classification of 64%. One can wonder how much the pretraining is accountable for this accuracy result : would a not pretrained model have performed the same way or even better after a similar training for classification ?

To answer that question, we instanciated our model without any pretraining (ie with random weights) and subsequently proceeded to the same training of the classification layer as for our pretrained model (30 epochs of gradient descent with the exact same parameters).

As a result, the accuracy of the model trained for classification but without any pretraining scored an accuracy of 68% of correct predictions for movie reviews classification (our pretrained model had scored only 64%). Therefore, the pretraining as we did it seems to have been no more than a poor initialization position for the gradient descent of the classification layer.

There might be different reasons that could explain the poor performance of hierarchical pretraining in our case. Notably, it could be that our custom FastText model trained only on the limited OpenSubtitles data has overfitted these data and could not really capture the signals of movie reviews, which are utterances of quite different lexical fields. Another possibility could be that we wrongly assumed that predicting a masked utterance from a time-split sequence of OpenSubtitles utterances would be similar to predicting a punctuation-split movie review.

To get a clear-cut answer about the relevance of the hierarchical pretraining for the movie reviews classification problem and maybe find a way of improvement of the baseline accuracy score, it would be interesting to pretrain the autoencoder on a dedicated portion of the Allocine movie reviews dataset, in order to elude the potential issues of too different lexical fields and the time-split utterance/punctuation-split sentence similarity assumption.

.

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*.

Wojciech Witon*, Pierre Colombo*, Ashutosh Modi, and Mubbasir Kapadia. 2018. Disney at iest 2018: Predicting emotions using an ensemble. In *Wassa @EMNP2018*.

Pierre Colombo*, Wojciech Witon*, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia. 2019. Affect-driven dialog generation. *NAACL 2019*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for selfsupervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Alexandre Garcia, Pierre Colombo, Slim Essid, Florence d'Alché Buc, and Chloé Clavel. 2019. From the token to the review: A hierarchical multimodal approach to opinion mining. *arXiv preprint arXiv:1908.11216v3*.

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. *arXiv preprint arXiv:1907.11692*.

Hamid Jalalzai*, Pierre Colombo*, Chloé Clavel, Éric Gaussier, Giovanna Varni, Emmanuel Vignon, and Anne Sabourin. 2020. Heavy-tailed representations, text polarity classification & data augmentation. *NeurIPS 2020*.

Emile Chapuis, Pierre Colombo, Matteo Manica, Matthieu Labeau, and Chloé Clavel. 2020. Hierarchical pre-training for sequence labelling in spoken dialog. *arXiv preprint arXiv:2009.11152v3*.

Pierre Colombo. 2021. *Learning to represent and generate text using information measures*. Ph.D. thesis, (PhD thesis) Institut polytechnique de Paris.

Pierre Colombo, Chloe Clavel, and Pablo Piantanida. 2021. A novel estimator of mutual information for learning to disentangle textual representations. *ACL 2021*.

Georg Pichler, Pierre Jean A Colombo, Malik Boudiaf, Günther Koliander, and Pablo Piantanida. 2022. A differential entropy estimator for training neural networks. In *ICML 2022*.

Pierre Colombo, Guillaume Staerman, Nathan Noiry, and Pablo Piantanida. 2022. Learning disentangled textual representations via statistical measures of similarity. *ACL 2022*.