IFT-7022 Natural Language Processing

Assignment 3

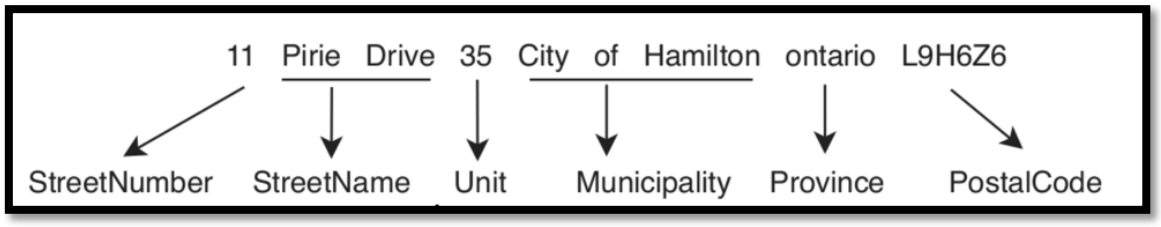
Ayoub. E *Oussama. A* *Marouane. M*

Presented to
Mr. Luc Lamontagne

Faculty of Science and Engineering
Laval University
Fall 2022

Sequence Labeling with a Transformer – Postal Address Analysis

- Task: label the different parts of an address in order to locate the civic number, street name, city, postal code, etc.
- The problem must be solved with a Transformer.
- Choose a pre-trained model and fine-tuning it on the training data provided.



- A few points:

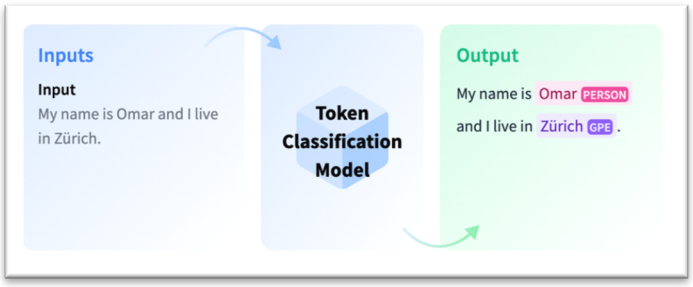
- *It is **recommended** that you use a small subset of the data made available by the authors of the tutorial. Otherwise training the model with 1 million examples could take several days and cause problems at runtime, which is not desirable in an academic context. The goal is to learn how to use transformers and not to reproduce this work entirely.*

- *You can limit yourself to the use of data from a geographic sub-region (e.g. Quebec or Ontario) if you wish. You can also randomly choose the examples.*

- *Present the results obtained with your model and make an analysis of some errors made by the model.*

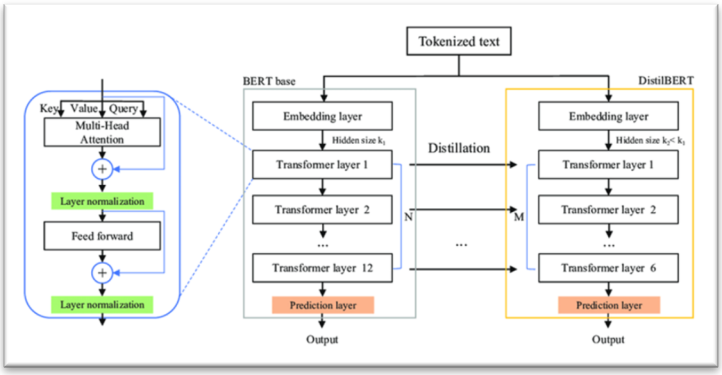
1) Transformer Architecture

Although LSTMs became very popular for NLP tasks, their design still had the problem of information loss and difficulties in training due to the passing of information through an extended series of recurrent connections. Furthermore, the sequential nature of their computational graph limited any usage of parallel acceleration, which led to the development of solution that solved this kind of issue. The model we are asked to use in this work is one of those solution, which is called the Transformer Architecture, a powerful approach for sequence processing which eliminated the usage of recurrent connections and is now used for task ranging from natural language processing to computer vision with the recent breakthrough of Vision Transformers. Now the task mentioned above can be understood as **Token classification** (See figure 1), which can be seen as assigning labels to individual tokens in a sentence. This can be well done by fine-tuning a pre-trained transformer on our training dataset and then using it for inference. Logically, we will be using a **Encoder-based Transformer**, which are known to convert input sequences (text) into a rich numerical representation that is well suited for classification tasks. A popular example is BERT, also known as Bidirectional Encoder Representations from Transformers.



- Figure 1: Example of Token Classification task.

After some research, we decided to use a type of pre-trained Transformer called ***"DistilBERT base model (Cased)".*** This model is known to be a general-purpose pre-trained version of BERT, but 40% smaller, 60% faster, and is able to retains 97% of the language understanding capabilities. DistilBERT was pretrained on the same corpus as BERT in a self-supervised fashion, which means it used only the raw texts without humans labelling. And like BERT, it used the concept of Masked language modeling (MLM) in its training loss, where 15% of the words in the input were randomly masked and the model needed to predict the masked words. In addition, a key concept implemented in this model is what is known as Knowledge distillation, which can be seen as a compression technique where a compact model (the student) is trained to reproduce the behaviour of a larger model (the teacher), which in this case was BERT (See figure 2). Therefore, DistilBERT had the same general architecture as BERT but was acting as the student of BERT via the crafting of its objective function so it is able to generate hidden states as close as possible to BERT and also the same output probabilities. Additionally, the version we used is the "*cased*" version, which means it makes a difference between uppercase and lowercase vocabulary. We made this choice as we hypothesised that this may be important for the classification of Tags like street names, cities, and postal codes.



- Figure 2: Distillation Training and DistilBERT architecture.

2) Address Labeling & Data Processing

The original dataset we had in hand consisted of 1,010,987 complete French and English Canadian addresses and their associated tags. The author of the blog we had access to used 728 789 examples for their training set, 182 198 examples for their validation set, and 100 000 examples for their test set. Now in order to avoid long training times and perform fast iterations on the models, we decided to divide this split by a constant (=300), so our dataset was randomly divided into 2429 training examples, 607 validation examples, and 333 testing examples. The training set can be seen as a list of tuples where the first element is the full address, and the second is a list of tags corresponding to the ground truth (8 classes). For example, here is two training examples:

(['213', 'london', 'road', 'newmarket', 'on', 'l3y', '6h8'],
['StreetNumber', 'StreetName', 'StreetName', 'Municipality', 'Province', 'PostalCode', 'PostalCode'])

(['368', 'rue', 'de', 'chalifoux', 'appt', 'a', '2', 'gatineau', 'qc', 'j8r', '2y6'],
['StreetNumber', 'StreetName', 'StreetName', 'StreetName', 'Unit', 'Unit', 'Unit', 'Municipality', 'Province', 'PostalCode', 'PostalCode'])

For the tokenization and the encoding of our tokens/tags, we used the DistilBert tokenizer with truncation and padding as True in order to make sure all of our sequences are padded to the same length and are truncated to be no longer model's maximum input length. To resolve the problem of vocabulary mismatch between our corpus and DistilBert's vocabulary, we used WordPiece Tokenization where single words are split into multiple tokens such that each token is likely to be in the vocabulary. To resolve the created problem of having only one tag per token, we used the tag labels for the first subtoken of a split token only. This was done using the offset_mapping function from the hugging face library. Finally, we turned our labels and encodings into a Dataset object in PyTorch using the "torch.utils.data.Dataset" object.

3) Training

The training of the Transformer corresponded to the fine-tuning of the DistilBert model on our training data. The process was performed with the hyperparameters in the table 1. Overall, the training was pretty fast, and accuracy was already high after 5 epochs. But in order to really compare the utility of using a Transformer, we decided to also train a bi-directional LSTM from scratch on the training/validation/test data. FastText embeddings were used for this model and it took 150 epochs for getting good accuracies as this model was not pre-trained.

The batch size was kept at 64 for the two models for effective comparison and the main accuracy metric correspond to the proportion of correct predictions among the total number of cases processed. In this context, a single prediction corresponds to the tag associated to an element of an address. Hence, it is not the right label predictions for all the elements in an address. Finally, the Transformer took 33 minutes to train while the LSTM took 125 minutes.

Type	Transformer (DistilBERT-base)	Bi-Dir. LSTM
Batch size	64	64
No. of epoch	5	150
Optimizer	Adam	SGD
Hidden dimension	3072	200
No. of heads	12	____
Learning rate	$5e^{-5}$	0.1 (decay)
Activation	gelu	ReLu
Dropout	0.1	0
Vocab_size	28 996	____

- Table 1: Hyperparameters used for the two models

4) Results

Model	Accuracy (%)
Transformer	99.47
Bi-Directional LSTM	97.95

- Table 2: Test Accuracy for both the Transformer model and the Bi-Directional LSTM.

Class	F1 Score	Instances
Unit	89.36	23
Street Number	99.21	314
Postal Code	100	333
Province	100	333
Street Name	99.31	725
Municipality	98.95	333

- Table 3: Evaluation metrics in function of the class for the Transformer model.

Misprediction 1	**Address**: ['nº', '154', 'labonte', 'longueuil', 'qc', 'j4h2p4']
	True Tag: ['StreetNumber', 'StreetNumber', 'StreetName', 'Municipality', 'Province', 'PostalCode']
	Model Prediction: ['Unit', 'StreetNumber', 'StreetName', 'Municipality', 'Province', 'PostalCode']
Misprediction 2	**Address**: ['3', 'albert', 'st', 'w', 'norwood', 'on', 'k0l', '1y0']
	True Tag: ['StreetNumber', 'StreetName', 'StreetName', 'StreetName', 'Municipality', 'Province', 'PostalCode', 'PostalCode']
	Model Prediction: ['StreetNumber', 'StreetName', 'StreetName', 'Municipality', 'Municipality', 'Province', 'PostalCode', 'PostalCode']
Misprediction 3	**Address**: ['6th', 'stre', 'burnaby', 'b.', 'c', 'v3n3n4']
	True Tag: ['StreetName', 'StreetName', 'Municipality', 'Province', 'Province', 'PostalCode']
	Model Prediction: ['StreetNumber', 'StreetName', 'Municipality', 'Province', 'Province', 'PostalCode']
Misprediction 4	**Address**: 'rue', 'du', "couvent', 'gatineau', "كيبيك'" , 'j9h', '3e1']
	True Tag: ['StreetName', 'StreetName', 'StreetName', 'Municipality', 'Province', "PostalCode", "PostalCode"]
	Model Prediction: ['StreetName', 'StreetName', 'StreetName', 'Municipality', Municipality', "PostalCode", "PostalCode"]

- Table 4: Examples of mispredictions made by the Transformer model.

5) Analysis & Conclusion

As we can see from the results above, the two models can be considered as highly performant as the Transformer achieved 99.47% accuracy on test data while the Bi-directional LSTM achieved 97.95%. We decided to train as a baseline a bi-directional LSTM directly instead of a traditional LSTM as the former is known to be more powerful in this context. Instead of simply considering the sequence from the start to the end, the bi-directional approach can see both the sequence from the start to the end and from the end to the start by using two layers of hidden state for each direction, which means that the model is able to gathers information from both directions before making a prediction. Although the former achieved a slightly better results than the ladder, it is important to consider that the training regime was not the same as the DistilBERT-base model was a pre-trained model very similar to the popular BERT that was fine-tuned on our training corpus for 5 epochs while the LSTM was trained from scratch on our training corpus for 150 epochs. The major result that can be deduced from this experimentation is the computational advantage of using pre-trained models instead of training models for each individual task that share a degree of similarity. This related to NLP tasks but also to other type of tasks like Vision tasks.

Looking at the table 3 where the F1 score of the Transformer is displayed in function of the respective classes, we can see that the Postal Code & Province classes achieved the highest score while the Unit class achieved the lowest one, although its number of instances was much lower than the others with 33 samples only in the test data. This means that the predictions for this specific class may be of lesser quality than the other ones. And looking deeper at this Unit class, we saw that its Precision metric was 87.5 while its recall was higher at 91.3, indicating that the model did better at predicting the positive examples than the negative ones. Also, the f1 score in this context represents the harmonic mean of the precision and recall metrics, which can be computed with the equation: F1 = 2 * (precision * recall) / (precision + recall).

Finally, looking at the misprediction examples from the Transformer network displayed in the table 4, we can see that many of the mistakes made when predicting the right tag were made with out-of-distribution samples that even a human or a rule-based system could have done. Indeed, looking at the misprediction 4, we can see an address with an entry from another language (Arabic) which translates to Quebec in English, which was the entry to put from the start. To conclude, we can say that the Transformer trained on this task is highly performant and using pre-trained models fine-tuned on the desired task is a powerful paradigm that should be applied in almost all learning tasks when possible. Nonetheless, we think that training a Transformer on this task did not really test the key performance factor of this architecture, which is the ability of modeling long term dependencies really well.