Title Detection System

Marc Torrellas Socastro

July 12th, 2019

1 Abstract

The package at hand provides a system for Machine Learning (ML) as a service, where the user can train, predict, and evaluate on data describing title candidates in text documents. Sample data for testing, and a model, are also provided, in case the user does not have labelled data to train with. With few data (less than 5k samples), the system achieves high ROC AUC score. Besides, a number of potential future tasks to improve the system in different aspects are proposed.

2 Problem statement

The system works with data in a CSV format as follows. Each line represents information about one particular block of text from the document. This information contains:

- Text: the raw text of the section as interpreted by the OCR;
- IsBold: is the section bold or not;
- IsItalic: is the section italic or not;
- IsUnderlined: is the section underlined or not;
- Left: the left coordinate on the page;
- Right: the right coordinate on the page;
- Top: the top coordinate on the page;
- Bottom:the bottom coordinate on the page
- Label: a label to state if the section represents a title (1) or normal text (0).

The objective is to predict, for each sample, whether or not it is a title, i.e. the label, using the rest of features. Notice the difference between Text and the rest of fields, as the former cannot be directly used to feed a ML model, and some numericalisation is required.

3 Proposed model architecture

The system provides ML as a service, i.e. the user can train, predict, and evaluate given the data. Fig. 1 describes the high level pipelines associated to each of these 3 functionalities[1].

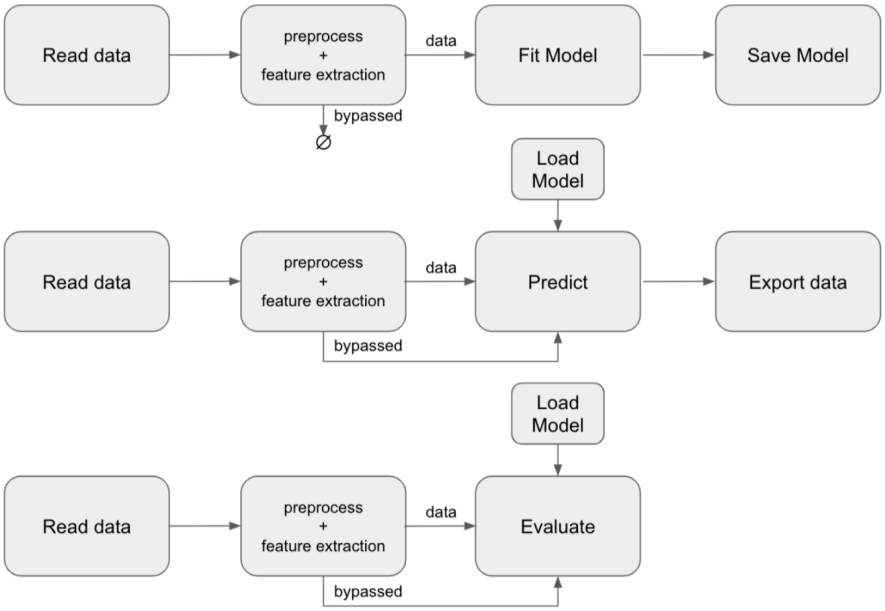

Figure 1: Pipelines for train (top), detect (middle), and evaluate (bottom).

Notice that some samples are definitely not titles, e.g. those containing only non-ASCII characters. We found many of those samples in the data, and they are denoted hereafter as *bypassed samples*.

Given that we have two types of features, we propose to use two models:

- Slave model: predicts whether the sample is a title or not just by using the text field. We have used a model based on the FastAI library [1], where first a language model is fine-tuned to the current data, and then the classifier is fit to the task at hand, following the 3-stage ULMFiT methodology [2].
- Master model: predicts whether the sample is a title or not using all features but the text feature. Here we use not only the fields described in the previous section, but also the output of the slave model, and other spaCy-based features. In this case, we have used a simple feedforward neural network with one hidden layer.

These two models are used in the fit and predict blocks from Fig. 1 as shown in Fig. 2.

[1]Notice we call *detect* to the pipeline associated to predict, to differentiate from the block called *predict*.

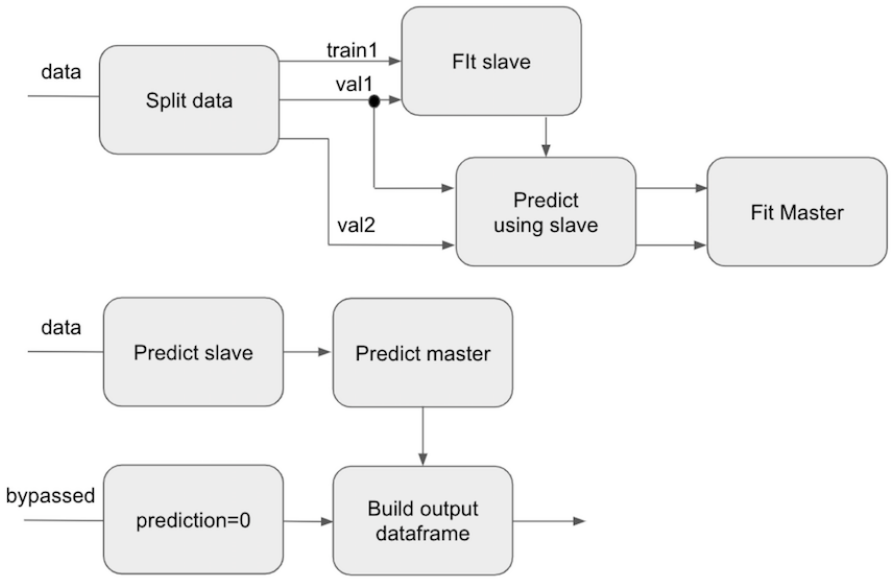

Figure 2: Pipelines for fit (top), and predict (bottom).

4 Model evaluation

The proposed architecture has been evaluated for the sample data, both enabling and disabling the slave model.

4.1 Slave disabled

In this case, training is really fast, entailing only 8 secs to carry out the train pipeline[2]. The results for each dataset in terms of loss and ROC AUC score are detailed next:

- Train (Loss, Score): 0.1354, 0.9869
- Val (Loss, Score): 0.1428, 0.9846
- Test (Loss, Score): 0.1428, 0.9846

4.2 Slave enabled

In this case, training is really slow, unless we use GPUs. The following durations (using the same machine as above) were recorded:

- Train the language model: 16 min

[2]All figures in this report refer to using a MacBook Pro with 4 cores and 8GB RAM

- Train the text classifier: 13 min
- Predict on validation set 1 with the text classifier: 33 min
- Predict on validation set 2 with the text classifier: 23 min
- Train master model: 4 secs

while the results for each dataset were:

- Train (Loss, Score): 0.1106, 0.9906
- Val (Loss, Score): 0.1216, 0.9882
- Test (Loss, Score): 0.08549, 0.9954

4.3 Analysis

Taking into account that we are approaching to the perfect classifier, the improvement from enabling the slave model is significant. However, our recommendation is to enable this mode only in case of having access to GPUs. In our experiments, the durations above for training and prediction divided by 4.

5 A brief overview of code architecture

See README.md for some examples

TODO: did not have time to complete this section

6 Future work

While there are many TODOs throughout the code, here we have collected some of the most relevant pending/tentative tasks, and ideas for the following version.

6.1 Model: performance

- Optimise master and slave jointly. At the moment, the optimisation of slave does not take into account that there is a master model taking its predictions as inputs.
- Analyse the data to see if there are more candidate patterns to be bypassed
- Using spaCy large model might help, as it contains a better tokeniser
- Augment feature space using more spaCy based features: NER counts, PoS counts, LDA...

- Use soft predictions (the score rather than a binary output) from the slave model. If so, it is fundamental to calibrate scores [3].
- Tune hyper-parameters and use early stopping. Use nested cross-validation (CV) if not a lot of data, to avoid too optimistic performance estimations.
- In a real situation, given this data we would use all data to train after having measured performance with test
- The problem can be transformed to sequence tagging if the document each sample belongs to was provided. There are many popular architectures for this, e.g:
  - Use handcrafted features and a Conditional Random Field (CRF)[4]
  - Use the popular BiLSTM + CRF architecture [5]
  - Use a Transformer[6]-based architecture like BERT [7]
- Experiment using newer stuff like BERT, GPT, XLnet

6.2 Model: usability

- Allow the user to get soft scores from command-line arguments.
- Columns are changing names in the output dataframe. This might be an issue for some users. If so, it's not difficult to keep the original names.
- If the code is too slow, a non deep learning solution (with the data provided, at least) provides similar performance and probably faster, see notebooks.
- Turn the config file to raw text file, and allow the user to pass a config file to use alternative settings
- Fix all random seeds for the sake of reproducibility. Clients expect to see same results after carrying out same operations with same data

6.3 Model: evaluation

- Use cross-validation to get a more accurate evaluation of performance, together with a measure of confidence for the score
- Use other scorers, e.g. F-score given that data is imbalanced, or even better, F-score gain [8].
- The loss when *evaluate* is not considering the bypassed. Probably it would not change dramatically, but the figures are not rigorously correct.

6.4 Coding: readability

- Some of the lines in pipelines.py are very similar, or even duplicated. Structuring the code in a better way would allow (a) to improve readability, and (b) to update all occurrences more efficiently.
- Add more docstrings and review naming

6.5 Coding: efficiency

- Because the lack of time, we did not have time to investigate how to predict more efficiently using the fastAI model. Hence, we implemented a for loop and predicted sample by sample. We proposed in the TODOs a way to parallelise using concurrent.futures, but didn't have time to test everything again. With some small experiments, it seems to divide by about the number of cores the prediction time. This is however not a problem in case of having GPUs, see experiments in notebook using Google Colab.
- It does not seem necessary to save the two fastAI-based models (language model and classifier) to disk.

6.6 Coding: others

- Write tests to reduce the cost of deployment, and releasing new versions
- Because the lack of familiarity with packaging and setuptools (it's the first time used), we did not manage to find a way to default to sample data without being in the project folder. We suspect a MANIFEST.ini is required, but didn't have time to finish this.
- setup.py is a bit hacky right now, calling "python -m spacy download en_core_web_sm". We opt for this because we did not find a way to install the spaCy language model using setuptools.

6.7 Report

- Explain more in detail what features are used.
- Explain the different hyper-parameters used, e.g. batch size, epochs, etc
- Explain the architecture of slave and master models more in detail.
- Explain the code architecture, and user interface

- Present the results in a more compact and visual way, and more results, e.g. confusion matrices.
- Cite the published papers, not the ArXiv ones, when available.

References

[1] FastAI-team, "https://github.com/fastai/fastai," 2018.

[2] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018.

[3] M. Kull, T. S. Filho, and P. Flach, "Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 623–631, PMLR, 20–22 Apr 2017.

[4] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[5] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," 2015.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018.

[8] P. Flach and M. Kull, "Precision-recall-gain curves: Pr analysis done right," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 838–846, Curran Associates, Inc., 2015.