Multipage document classifier (Machine Learning 2022 Course)

Evgenia Komleva [1] **Kirill Vayser** [1] **Georgy Mkrtchyan** [1] **Vadim Artemov** [1]

Abstract

We create model, which can split large pdf with several documents of the same type in small documents. For example, 10 pages of which the first document is 5, the second is 3 and the third is 2 and we want to split it into these three documents. That is, we need to build a classifier for pages that will predict whether the page is the first last or is in the middle. The document type is known, but not the quantity in the large batch.

Github repo: https://github.com/EvgeniaKomleva/Multipage-document-classifier
Video presentation:

https://www.youtube.com/watch?v=nH27c3OG5UQt=3s

1. Introduction

Even with the present level of technological progress most of the business is done using documents and the amount of paperwork involved will vary industry to industry. Many of these industries need to scan through scanned document images, which usually contain non-selectable text, to get the information for key index fields to operate their daily tasks. To achieve this, the first major task is to index different types of documents, which later helps in extraction of information and meta-data from a variety of complex documents. Apart from machine learning methods companies would execute this task by using manual or partially automated classification techniques i.e rule engines, template matching. The underlying problem which is faced by the current implementations is that the staff has to manually find and sort the documents present in the packages. Although, some degree of automation is achieved using keyword searches, regular expressions etc. The accuracy and robustness of such solutions are questionable and their manual workload reduction is still not satisfactory. Keyword searches and regular expressions means that these solutions need to account for every new document or document variations which are presented and also need to add rules for that. This in itself becomes a manual effort and only partial automation is achieved. There still remains a chance where the system might misclassify document between two classes, because of common rule present in both. Additionally, there is no degree of certainty towards an identification. More often than not, manual verification is still required. On top of that, if the manual work is too much, human error tends to increase.

Our solution consists of exploring different machine learning techniques to solve the problem of document classification. Proposed methods vary from classical algorithms like random forest to complex neural networks such as BERT(Devlin et al., 2018) and LSTM(Staudemeyer & Morris, 2019) cells.

The main contributions of this report are as follows. We implemented LSTM model over the BERT representation. We trained page embeddings using BERT and feed LSTM as input. This approach gives the best result - 87 percent accuracy.

2. Related work

A similar problem of classifying documents containing multiple unordered pages was considered in (Gordo & Perronnin, 2010). Authors proposed novel bag-of-pages document representation. The authors of (Frasconi et al., 2001) exploited bayesian approach of text categorization in multipage documents. In (Gordo et al., 2013) a method for the segmentation of continuous page streams into multipage documents and the simultaneous classification of the resulting documents is presented.

3. Algorithms and Models

You can see our baseline on 1

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Evgenia Komleva <Evgenia.Komleva@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

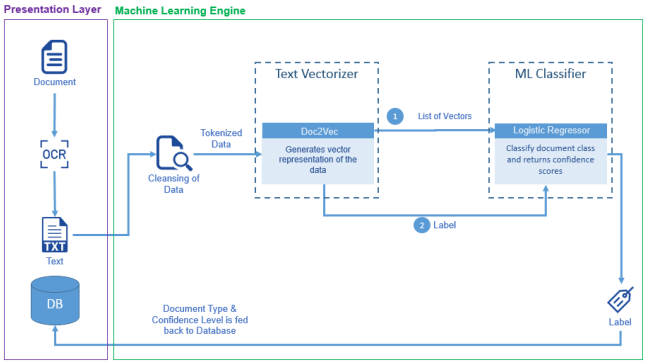


Figure 1. Baseline

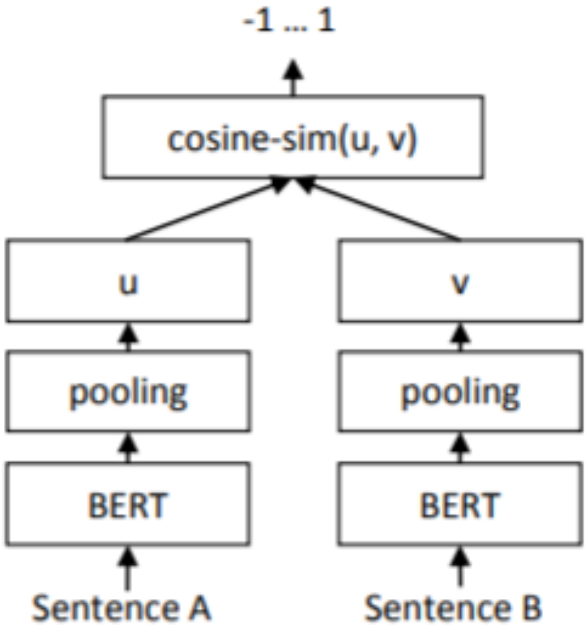
3.1. Data preprocessing

Embeddings Word embeddings(Le & Mikolov, 2014) are a type of word representation which stores the contextual information in a low-dimensional vector. In our work we exploited two types of embeddings.

TF-IDF(Das & Chakraborty, 2018) is a statistical measure used to determine the mathematical significance of words in documents. TF stands for term frequency. In the simplest terms, term frequency is the ratio of the number of target terms in the document to the total number of terms in the document. IDF stands for inverse documet frequency. The IDF value is the logarithm of the ratio of the total number of documents to the number of documents in which the target term occurs. At this stage, it does not matter how many times the term appears in the document. It is sufficient to determine whether it has passed or not. The TF-IDF value is obtained by multiplying the TF and IDF values.

Doc2Vec model is based on Word2Vec(Chen & Sokolova, 2018), with only adding another vector corresponging to paragraph ID to the input. Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.Word2Vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words.

Sentence BERT is an advanced embedding method, which incorporates strength of BERT architecture. SBERT adds a pooling operation to the output of BERT to derive a fixed sized sentence embedding. Usage of siamese and triplet networks is able to derive semantically meaningful sentence embeddings.



3.2. ML algorithms

Logistic Regression Logistic regression(Chung, 2020) is a base widely-used classification model, which we employed as one of the baseline models. The idea of this method is the division of space of objects into areas corresponding to different classes.

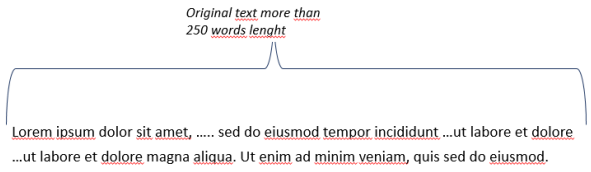
Random Forest Random forest(Louppe, 2014) is the classical example of bagging approach to the classification task. It consists of a set of Decision Trees and exploits decrease of variance by averaging many base estimators.

Stacking Classifier Stacked generalization consists in stacking the output of individual estimator and use a classifier to compute the final prediction. Stacking allows to use the strength of each individual estimator by using their output as input of a final estimator. Used base estimators : Random Forest Classifier with 100 trees, non-limited depth, 1 sample per leaf; Extra Trees Classifier with same parame-

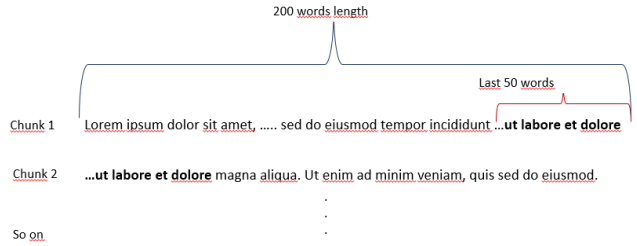
ters, Decision Tree with depth equal 15 and 5 samples per leaf; KNN classifiers with 10, 5, 1 neighbors. Used meta estimator: Random Forest Classifier with 100 trees.

BERT BERT(Reimers & Gurevych, 2019) (stands for Bidirectional Encoder Representations from Transformer) is a Google's Deep Learning model developed for NLP task which has achieved State-of-the-Art Pre-training for Natural Language Processing in multiples task. However one of its "limitation" is on application when you have long inputs, because in BERT the self-attention layer has a quadratic complexity $O(n^2)$ in terms of the sequence length n.

For solving that problem we are going to segment the input into smaller text and feed each of them into BERT, it mean for each row we are split the text in order to have some smaller text (200 words long each), for example:



We must split it into chunk of 200 word each, with 50 words overlapped, just for example:



Now we ready to fine-tune BERT. Create the model - basically is create a single new layer that will be trained to adapt BERT to our task.

We choose several parametrs: BATCH SIZE = 16, LEARNING RATE = 2e-5, NUM TRAIN EPOCHS = 1.0, WARMUP PROPORTION = 0.1,SAVE CHECKPOINTS STEPS = 300 , SAVE SUMMARY STEPS = 100

```
{'eval_accuracy': 0.7998267,
 'false_negatives': 73.0,
 'false_positives': 104.0,
 'loss': 0.55226713,
 'true_negatives': 50.0,
 'true_positives': 927.0,
 'global_step': 288}
```

BERT is amazing it achieved a good 80 procent just with fine tuning, however we are using the vector representation of this fine tuned model as input for another more simple model as follow.

LSTM model over the BERT representation Now we are going to build a simple LSTM model having as input the vectors created before, however in this case or when you have long text secuences the most of the time this sequences are variable, I mean there will be text with number of words 300, 550, 1000, etc, so the number of 200-length chunk is not fixed, so our vector of representacions are variable length.

Finally we train the model, using the keras callback named ReduceLROnPlateau which reduce the hyperparameter learning rate if the validation's accuary does not improving

```
from keras.callbacks import ReduceLROnPlateau
call_reduce = ReduceLROnPlateau(monitor='val_acc', factor=0.95, patience=3, verbose=2,
                                mode='auto', min_delta=0.01, cooldown=0, min_lr=0)
```

```
model.fit_generator(train_generator(df_train), steps_per_epoch=batches_per_epoch, epochs=20,
                    validation_data=val_generator(df_val), validation_steps=batches_per_epoch_val, callbacks =[call_reduce] )
```

```
Epoch 1/20
4571/4571 [==============================] - 63s 14ms/step - loss: 0.3412 - acc: 0.9098 - val_loss: 0.4273 - val_acc: 0.8867
Epoch 2/20
4571/4571 [==============================] - 61s 13ms/step - loss: 0.3025 - acc: 0.9169 - val_loss: 0.4091 - val_acc: 0.8882
Epoch 3/20
4571/4571 [==============================] - 61s 13ms/step - loss: 0.2928 - acc: 0.9175 - val_loss: 0.4100 - val_acc: 0.8848
Epoch 4/20
4571/4571 [==============================] - 62s 14ms/step - loss: 0.2845 - acc: 0.9210 - val_loss: 0.4036 - val_acc: 0.8882

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0009500000451225787.
Epoch 5/20
4571/4571 [==============================] - 63s 14ms/step - loss: 0.2813 - acc: 0.9202 - val_loss: 0.4090 - val_acc: 0.8921
Epoch 6/20
4571/4571 [==============================] - 61s 13ms/step - loss: 0.2780 - acc: 0.9206 - val_loss: 0.3948 - val_acc: 0.8965
Epoch 7/20
```

```
num_sequences_val = len(df_test['emb'].to_list())
batch_size_val = 4
batches_per_epoch_val = 343
assert batch_size_val * batches_per_epoch_val == num_sequences_val
num_features= 768
model.evaluate_generator(val_generator(df_test), steps= batches_per_epoch_val)
```

```
[0.41612950315069047, 0.8731778425655977]
```

Getting an accuracy of 87 procent and loss 0.41

4. Experiments and Results

To begin with, the target metric of the analysis was chosen to be accuracy. At the first step of the analysis our data was converted into the word embeddings via 3 different approaches : TF-IDF, Doc2vec, BERT sentence tagger. First, step of analysis was assigned to the preprocessing of the data. During this step input were subject to the automated preprocessing: tokenization and removal of stopwords and punctuation signs. As it was stated above the baseline approach – Logistic regression showed accuracy at the level of 0.37. Thus, 2 directions of the further work was chosen: Classical ML classificators and Neural based approaches. As candidates on the in the first Group Random Forest and Staking Classifier consisted of 3 KNN classifiers, 2 Decision Trees, a Random Forest and an Extra tree base models were chosen. The approach to choose the hyper parameters of the Random forest was chosen to be Randomized 3-fold Cross-Validation on the set of the main parameters:

- 1. Number of trees in random forest
- 2. Number of features to consider at every split
- 3. Max number of levels in tree

- 4. Min number of samples required to split a node
- 5. Min number of samples required at each leaf node

The distribution of resulted accuracies of the estimator of best parameters in each embedding was following : TF-IDF – 0.9, Sentence BERT – 0.89 and DOC2VEC – 0.73. What is interesting is that evaluation on the Stacking classifier showed the same pattern of the results, however , DOC2VEC embeddings showed significant deterioration of the performance to the 0.65 , while TF-IDF remains the leadership with 1 per cent increase up to 0.91 accuracy and second place is remained by Sentence BERT embedding with 0.9 accuracy. Such an observation may lead to a conclusion that for such a problem, where corpus of the documents is very broad TF-IDF is able to capture all the important information even under the loss of explained variance due to PCA, however, dimension reduction ,which was applied only on TF-IDF embeddings may have helped it to omit overfit and outperform remaining techniques. However, it is important to mention, that pre-trained on the wide range of the documents Sentence Bert showed similar performance for all of the classification approaches.

Moving to Neural approaches, in our analysis we considered two approaches LSTM and BERT. Both model are pretrained on the big corpus of different texts, however, to adjust the weights of the Neural Networks fine-tuning process was completed (details in section 3.2). After evaluation of both models on the test set the desired metric had following distribution: LSTM showed competitive to the Random Forest and Stacking classifier result of 0.87 accuracy, while BERT NN showed lower result of 0.8 accuracy. Generally, Neural approaches should outperform the classical ones , however, due to the fact that training data set contained only 6000 documents , such an amount of observations may be not enough for Networks to adjust to the data set, therefore , there are two ways of further development , either improve the size of the training set or switch to the non-neural approaches which constantly outperform BERT and LSTM on our data set.

5. Conclusion

So, now we would like to finish our discussion and provide a brief conclusion to our work. In our work we have tried to build a page classifier, which would automatically determine the type of page in a multipage documents: first, middle or last. We took a dataset provided by our team member, it was already classified. We took some time to check it, and then converted it into a machine-learning compatible format. We used Word2Vec based models, such as Sentence BERT, Doc2Vec and TF-IDF. Then we have trained a simple logistic regression as a classical benchmark to start with, accuracy there was 37%. After that we proceeded to train random forest classifier with some parameter tuning. Accuracy there increased to 85%. Followed by stacking classifier of several random forests and KNN's with random forest as a meta classifier we managed to obtain 79% accuracy. The next step for us was to try more sophisticated methods, such as neural networks. The first one of them was BERT. We fine tuned it, and obtained accuracy of 80%. We have decided then to try another model, and trained LSTM model, and obtained our best score of 87%.

Overall we have managed to implement a good model for the document page classification, which uses latest models availible with good accuracy.

References

Chen, Q. and Sokolova, M. Word2vec and doc2vec in unsupervised sentiment analysis of clinical discharge summaries, 2018.

Chung, M. K. Introduction to logistic regression, 2020.

Das, B. and Chakraborty, S. An improved text sentiment classification model using tf-idf and next word negation, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Frasconi, P., Soda, G., and Vullo, A. Text categorization for multi-page documents: A hybrid naive bayes hmm approach. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '01, pp. 11–20, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133456. doi: 10.1145/379437.379440. URL https://doi.org/10.1145/379437.379440.

Gordo, A. and Perronnin, F. A bag-of-pages approach to unordered multi-page document classification. In *2010 20th International Conference on Pattern Recognition*, pp. 1920–1923, 2010. doi: 10.1109/ICPR.2010.473.

Gordo, A., Rusiñol, M., Karatzas, D., and Bagdanov, A. D. Document classification and page stream segmentation for digital mailroom applications. In *2013 12th International Conference on Document Analysis and Recognition*, pp. 621–625, 2013. doi: 10.1109/ICDAR.2013.128.

Le, Q. V. and Mikolov, T. Distributed representations of sentences and documents, 2014.

Louppe, G. Understanding random forests: From theory to practice, 2014.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

Staudemeyer, R. C. and Morris, E. R. Understanding lstm – a tutorial into long short-term memory recurrent neural networks, 2019.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Evgenia

- Reviewing literate on the topic
- Coding the main algorithm BERT and LSTM over the BERT representation
- Preparing the GitHub Repo
- Preparing the Section about BERT and LSTM of this report
- Coding Logistic Regression

Kirill

- Working with TF-IDF embeddings.
- Coding stacking classifier.
- Preparing Introduction, Data Preprocessing and Classical ML methods part of this report.

Georgy

- Text cleaning and Tokenization.
- Dimension reduction.
- Creation of Doc2Vec and Sentence BERT embeddings.
- Working on Random Forest GridSearch.
- Description of the Experiments and Results Section.

Vadim

- Coding random forest model
- Preparing slides
- Preparing report
- Working on video

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: Own-0.95/Taken 0.05

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☐ Yes.
   ☑ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☐ Yes.
    ☐ No.
    ☑ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    - ☐ Yes.
    - ☐ No.
    - ☑ Not applicable.

    Students' comment: None