NLP PROJECT

Project 3. Fake Job Postings Detection

Denys Kaliuzhnyi, Denys Krupovych, Pavel Chizhov

2022

Introduction

The aim of this project is to build a model that classifies job postings into two classes: fake (1) and real (0). The data is a multi-column table of job postings, containing textual, numeric and categorical features. The model is applicable in many ways: it can be used by job postings aggregators to filter and strike suspicious job offers and employers.

In this project we had to pay special attention to textual features, because they are most likely to contain helpful information. The most challenging parts were related to model pipeline choice and the structure of the data: the dataset is highly unbalanced: positive class labels make up less than 5% of the data.

On the initial stage the data was given as train and dev partitions. The holdout test set was hidden and used in further evaluation. The main evaluation metrics were F1-score and its micro and macro averaged variations.

Related work

The main search for related papers was done in two directions at once: the goal was to find articles with effective pipelines for classification using features of different types (including textual) and to read deeply about the processing of textual features for classification.

BERT[1] was chosen as the main model architecture for textual features as it is a powerful NLP model capable of solving a multitude of tasks, e.g. text classification, question answering, etc. Its novelty lies in a new bidirectional approach of sequence processing and it was trained with the help of masked language modeling technique along with the next sentence prediction objective. Such training setup is quite beneficial as it can be achieved in an entirely unsupervised strategy, consequently the corpus BERT was pretrained on is enormous (BooksCorpus 800M words + Wikipedia 2,500M words). In the current project a decision to use BERT provides a very flexible toolkit: we can either fine-tune the model further on the downstream task or just use its pre-trained weights as a feature extraction stage for textual data. Both approaches should work and they are a question of accuracy-simplicity trade-off.

In terms of architecture, the paper about Hybrid Deep Neural Networks for Mixed Inputs[2] is highly relatable. The authors introduce a new general-purpose machine learning technique called HDNN to build a model over multiple-type input features, which can include images, sequences and tabular data. The idea is to conceptually divide the training process into two stages: feature learning and target learning. The first stage is simply a set of ML models designed to process the

features they are useful for, e.g. some sort of CNN for images, RNN for sequences and MLP for tabular data. Then, the output of these models is concatenated into an ensemble feature and passed to a second stage. Lastly, the target learning part is an MLP model that acts as a final predictor. That approach is closely related to our task as we also have multiple-type input features. Although the paper describes model application in a context of reservoir production prediction (geography domain), the authors also explained the concept in a domain-independent way, that's why this methodology is applicable in case of the current project.

One of the works with a pipeline for multi-featured data that impacted the solution is "Enriching BERT with Knowledge Graph Embeddings for Document Classification"[3]. This article proposes a multilabel classification ensemble model consisting of BERT and an MLP for concatenated BERT features and non-textual features from the data. Architecture-wise, the proposed model combines textual and non-textual data in one model, using BERT for extracting numerical features from textual data and combining them in one model along with metadata (usual features) and other embeddings built during preprocessing. Also, the article contains some techniques for non-textual features preprocessing.

Data

In this project we use a “fake job position detection” dataset provided by the task itself. The data is in a tabular format and has 18 features in total. Each feature describes a job position from some aspect and their names are self-explanatory (e.g. *requirements*). Altogether they form a comprehensive overview about the job position. There is one special feature called *fraudulent* which tells us whether the job position is fake or real. In this project it is a target feature we aim to predict.
Among all the features we distinguish 4 types present:

- binary (e.g. *has_company_logo*)
- categorical (e.g. *required_experience*)
- short text (e.g. *title*)
- long text (e.g. *company_profile*)

Not all features seem to be useful. There are almost useless features like *salary_range* which has most of the values set to NA and those that are filled are quite fuzzy and not standardized in currency format. So, feature selection and preprocessing is definitely required for this dataset.

The next figures describe the sentence length distribution of tokenized long-text features that has to be checked before applying truncation for the BERT model. As we can see, truncation at the maximum of 512 tokens will not do much harm.

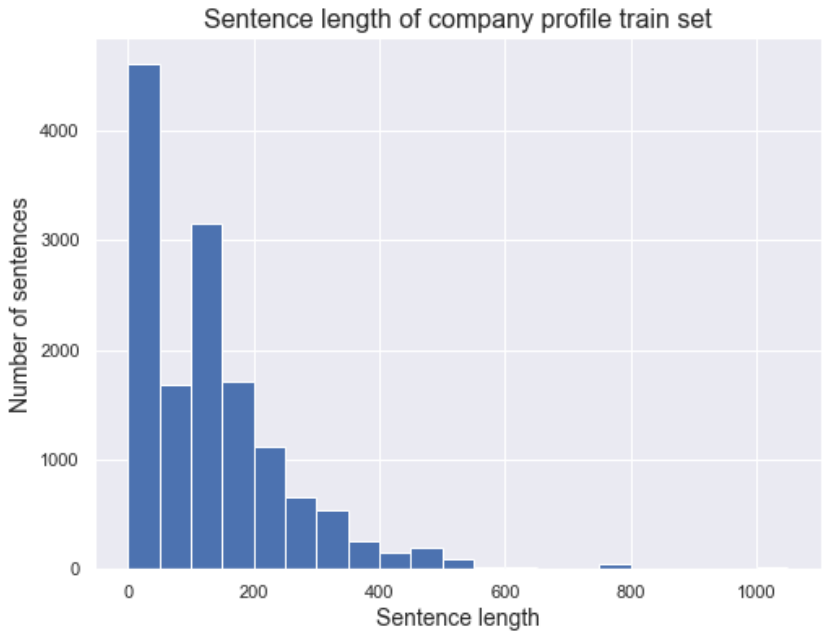

Figure 1 – Distribution of tokenized sentences of "company_profile" features in the train set

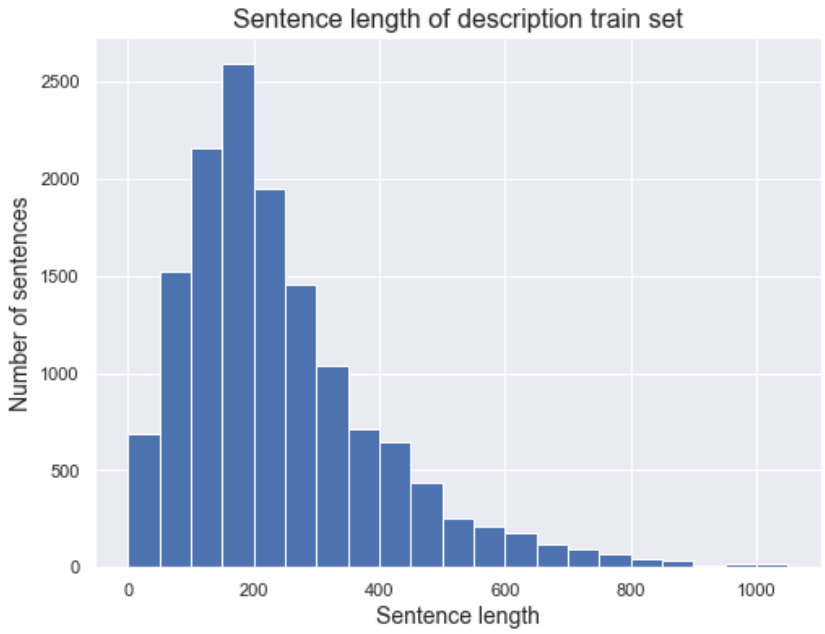

Figure 2 – Distribution of tokenized sentences of "description" features in the train set

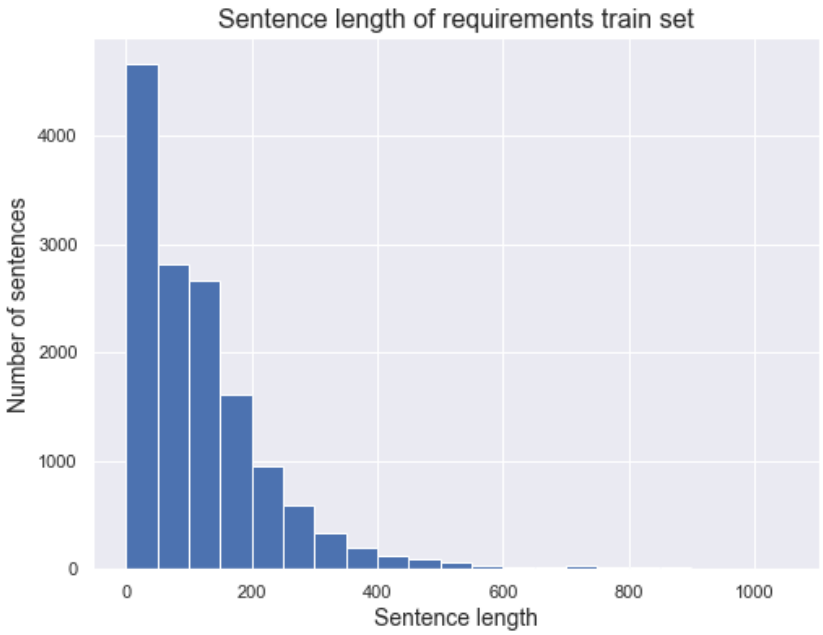

Figure 3 – Distribution of tokenized sentences of "requirements" features in the train set

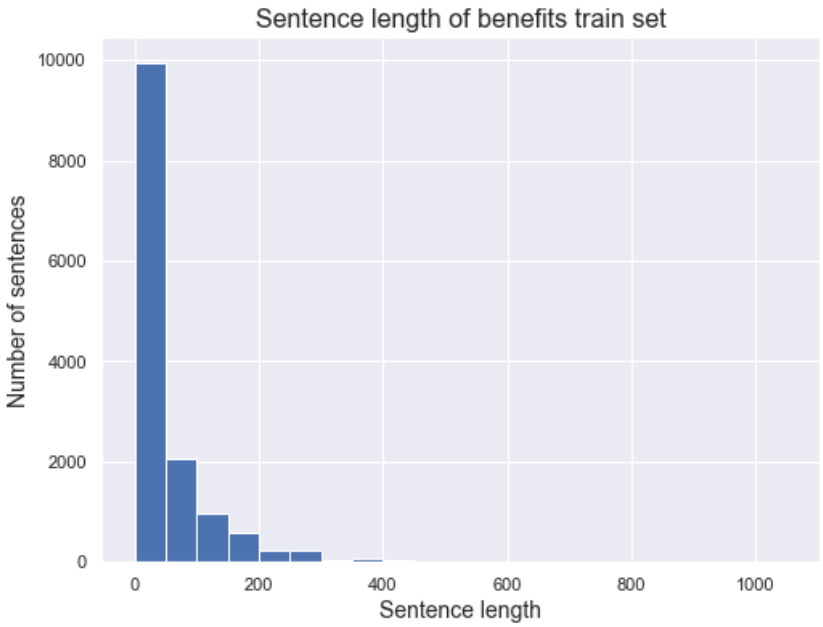

Figure 4 – Distribution of tokenized sentences of "benefits" features in train set

Method

The pipeline build for classification (see Figure 5) can be described in several steps:

1. Each long-text feature is tokenized with a pretrained tokenizer for “bert-base-uncased” and truncated if it is longer than 512 tokens.
2. Pretrained BERT model is used to extract 768-dimensional features from each long-text feature separately. Extracted embeddings are concatenated together and with numerical features.
3. The concatenated result is processed by a multi-layer perceptron with dropout, weight decay, and batch normalization as regularization methods.

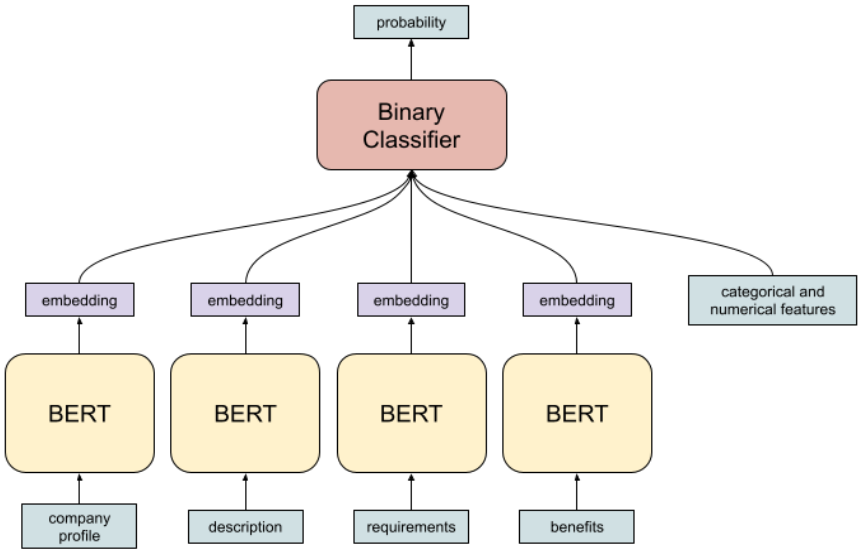

Figure 5 - Model architecture

The implementation was done using Python 3 language and libraries PyTorch and HuggingFace *transformers* and *datasets*. Preprocessing was performed using *datasets* library instruments and feature extraction was done by pretrained BERT model “bert-base-uncased” from *transformers*. The model was implemented as a PyTorch module using AutoModel from *transformers* as a base class.

The training was implemented using Trainer class from *transformers*. The model was trained for 20 epochs with Adam optimizer, learning rate of 0.0001, batch size 64 and 300 warm-up steps evaluating F1 score on dev set.

Special point has to be made about loss function. Usual binary cross-entropy loss was chosen, but due to high imbalance in data labels, the loss values were weighted: negative-positive class weights corresponded to the relation 1:4.

Results

In the process of training BERT embeddings were evaluated individually compared to simple Bag of words text representations in order to check if there is any profit from using BERT. The results are shown in Figure 6. Obviously, BERT representations are beneficial even for individual features.

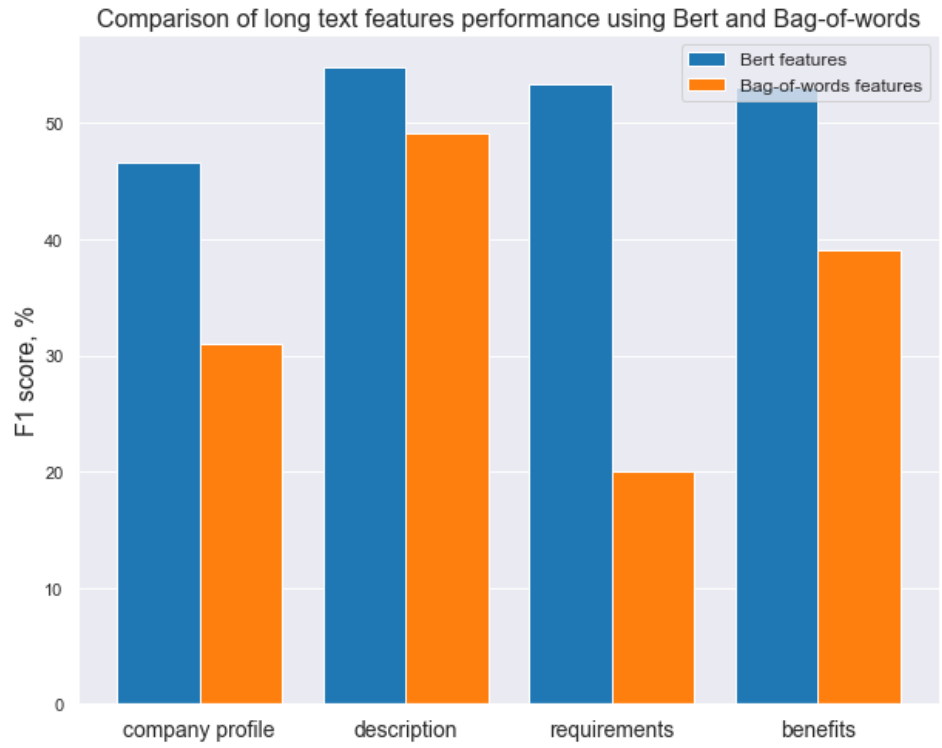

Figure 6 - BERT versus BoW

In addition, BERT representations were projected onto a 2-D plane using t-SNE dimensionality reduction technique. The results are in Figure 7. Green points correspond to normal job positions while red ones are fraudulent. In general, two classes are messed up, but still there are red points lying outside the main blob.

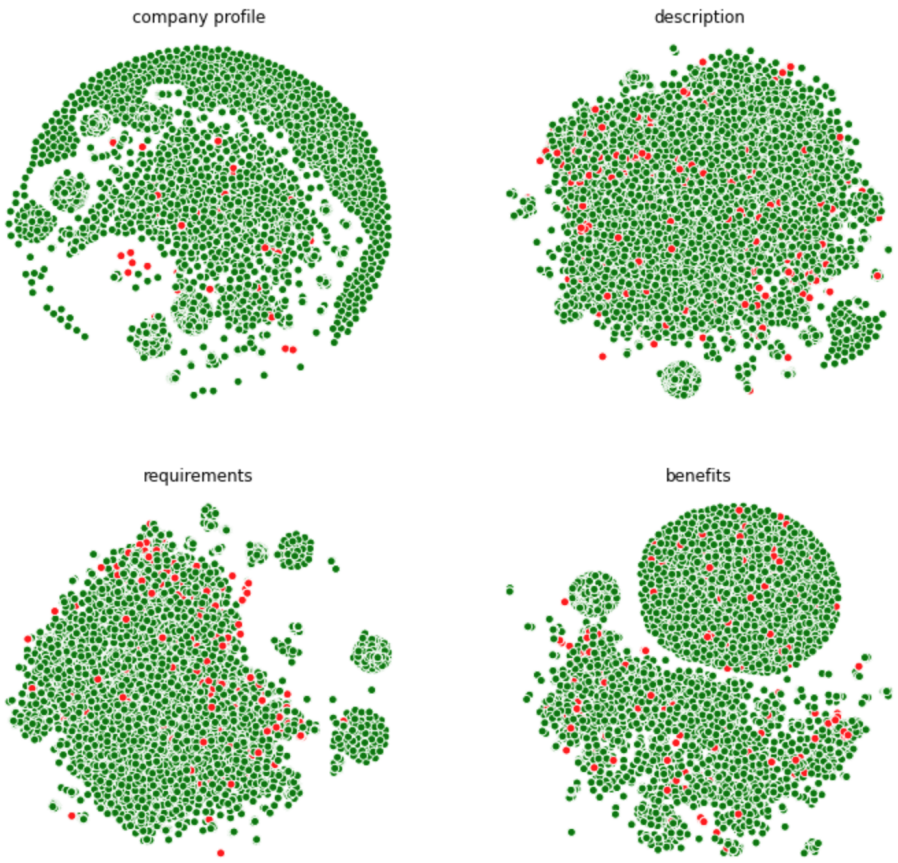

Figure 7 – t-SNE for BERT features

After the final evaluation achieved results on test set are the following:

- F1-score for the "fake" class (main metric): 0.86
- Micro-averaged F1-score: 0.99
- Macro-averaged F1-score: 0.93

This result corresponds to the third position in the leaderboard.

Discussion

Comparing the implemented model with initial plans and viewing the final results, it is necessary to make several remarks about possible ways to improve the proposed solution.

First of all, not all the data features were properly used in the model. Categorical and short-text features may be included into the model as well, given that they are properly preprocessed. With deeper understanding of categorical features, it is possible to regroup and/or reorder the values, impute missing values, and binarize the features. This may improve the score.

It may also be useful to try different approaches to pipeline design. For instance, separate multi-layer perceptrons can be applied to each textual feature representation. Furthermore, as described in one of the reviewed articles[4], the model on top of the features can take classification probabilities as inputs, so that a separate classifier is applied to each textual feature and non-textual feature set.

Finally, in the project only fully pretrained version of BERT was used for feature representations. It may be reasonable to reorganize the training procedure so that the top levels of BERT are involved in training (fine-tuning). This may lead to significant progress, though being more computationally challenging.

Conclusion

During this project a model for fake job postings classification was implemented using BERT feature extraction and an ensemble model based on a multilayer perceptorn. The results are rather promising, despite the dramatic imbalance in data, the model managed to produce a very competitive F1-score. In addition, the ways of further improvement of the model, that will likely improve the score, have been described. Continuing by those steps and proposing new ideas for improvement would probably lead to creation of a very robust fake job postings detector.

References

1. Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. ArXiv abs/1810.04805 (2018)
2. Yuan, Zhenyu, Yuxin Jiang, Jingjing Li and Handong Huang. “Hybrid-DNNs: Hybrid Deep Neural Networks for Mixed Inputs.” ArXiv abs/2005.08419 (2020).
3. Malte Ostendorff, Peter Bourgonje, Maria Berger, Julian Moreno-Schneider, Georg Rehm, Bela Gipp. “Enriching BERT with Knowledge Graph Embeddings for Document Classification.” ArXiv abs/1909.08402 (2019)
4. Maël Fabien, Esau Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. “BertAA : BERT fine-tuning for Authorship Attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)”*, pages 127–137, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI). (2020)