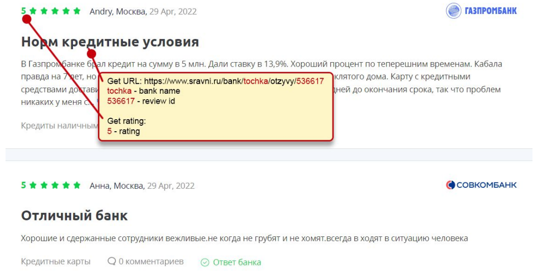
Natural Language Processing course

Sentiment analysis model for online reputation management system (ORM)

Aleksey Domnenko
@domnenko_a_n

Get data

Parsing sravni.ru -> links.csv



5 ★★★★★ Andry, Москва, 29 Apr, 2022 ГАЗПРОМБАНК

Норм кредитные условия

В Газпромбанке брал кредит на сумму в 5 млн. Дали ставку в 13,9%. Хороший процент по теперешним временам. Кабала правда на 7 лет, но … клятого дома. Карту с кредитными средствами достави… дней до окончания срока, так что проблем никаких у меня с…

Кредиты наличным…

Get URL: https://www.sravni.ru/bank/tochka/otzyvy/536617
tochka - bank name
536617 - review id

Get rating:
5 - rating

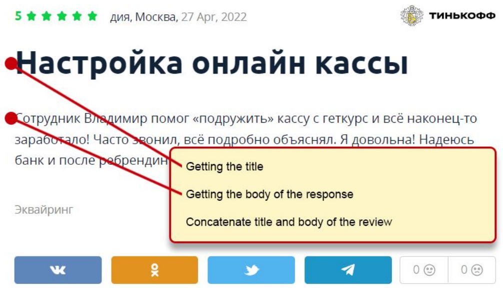
5 ★★★★★ Анна, Москва, 29 Apr, 2022 СОВКОМБАНК

Отличный банк

Хорошие и сдержанные сотрудники вежливые.не когда не грубят и не хомят.всегда в ходят в ситуацию человека

Кредитные карты 0 комментариев Ответ банка

Get Reviews -> data.csv



5 ★★★★★ дия, Москва, 27 Apr, 2022 ТИНЬКОФФ

Настройка онлайн кассы

Сотрудник Владимир помог «подружить» кассу с геткурс и всё наконец-то заработало! Часто звонил, всё подробно объяснял. Я довольна! Надеюсь банк и после ребрендин…

Эквайринг

Getting the title

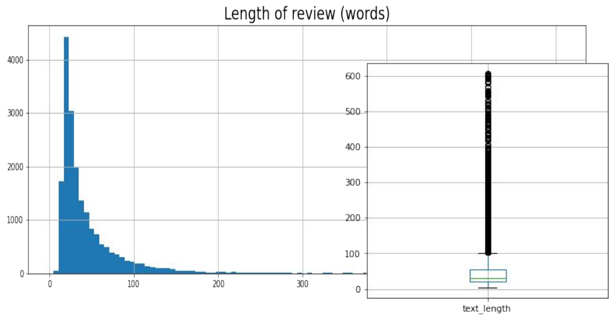
Getting the body of the response

Concatenate title and body of the review

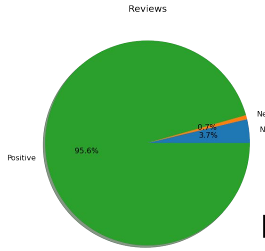
0 0

Data processing

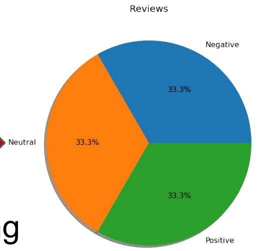
Data analysis



Data processing



class balancing



Solution

Sentiment analysis model

The text preprocessing and fine-tuning BERT multilang. model

- training data is 99,97%
- validation data 96,76%

My multinput model

fine-tuning BERT multilang. model
input 1 - text
input 2 - 3 features

- training data is 99,71%
- validation data 96,88%

1. Introduction

Development of a sentiment analysis model for online reputation management system (ORM). The sentiment analysis model classifies reviews into 3 classes:

- negative;
- neutral;
- positive.

Training takes place on our balanced dataset.

Peculiarities:

data: parsing reviews from the site sravni.ru
language: russian
data preparation:

- data analysis;
- working with missing values
- balancing data by class

model:

- text preprocessor (“bert_multi_cased_preprocess/3”)
- BERT model “bert_multi_cased_L-12_H-768_A-12/4”
- BERT fine-tuning classifies reviews into 3 classes: -negative, -neutral, -positive
- improved accuracy - multi-input model - implementation of additional input for features

2. Related Work

There are various methods and approaches for text classification.

1. Machine learning
Using machine learning to classify text - Logistic Regression, Naive Bayes (LDA a Bayesian version of pLSA), Support Vector Machines (SVM), Stochastic Gradient Descent (SGD), K-Nearest Neighbors (KNN), Random Forest, bigARTM — ARTM, and other.

2. Deep learning.
In text classification, the main deep learning models are widely used: convolutional neural networks (CNN) and recurrent neural networks (RNN), transformers.

One of the best solutions - based on BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google.

BERT and its preprocessing were originally published by Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018.
https://arxiv.org/abs/1810.04805

The solution is based on "Classify text with BERT" from TensorFlow
https://www.tensorflow.org/text/tutorials/classify_text_with_bert
and is an improved version that shows higher accuracy

3. Model

3.1. Sentiment analysis, BERT model to fine-tune

The sentiment analysis model classifies reviews into 3 classes: -negative, -neutral, -positive. Training takes place on our balanced dataset - **data_balanced.csv.**

The preprocessing model: **bert_multi_cased_preprocess/3**

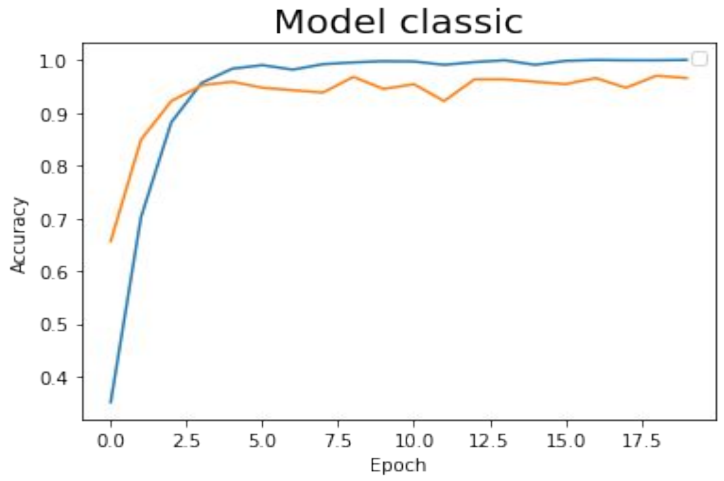
Text inputs need to be transformed to numeric token ids and arranged in several Tensors before being input to BERT. TensorFlow Hub provides a matching preprocessing model for each of the BERT models discussed above, which implements this transformation using TF ops from the TF.text library. It is not necessary to run pure Python code outside your TensorFlow model to preprocess text

BERT model: **bert_multi_cased_L-12_H-768_A-12/4**

Use the model output **pooled_output** after which add our layers for classification

The accuracy on this model is:

- training data is 99,94%
- validation data 97,03%



3.2. Sentiment analysis, my model

TensorFlow example model

Classical approach to sentiment analysis classification on BERT model with fine tuning from tensorflow: https://www.tensorflow.org/tutorials/keras/classification

mean accuracy: 0.9997, **mean** val_accuracy: 0.9676,

max accuracy: 1.0, **max** val_accuracy: 0.9763,

My multinput model

The essence of the approach is to use BERT fine-tuning, but at the same time provide additional features to the input 2:

- review length / max review length
- for text uniqueness (in review)
- mean word length

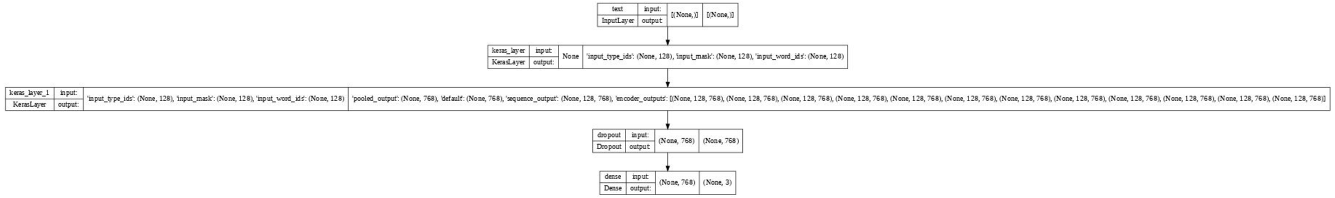
Let's create a neural network with several inputs, separately for the BERT layer ['pooled_output'] and separately for the feature layer

my model **mean** accuracy: 0.9971, **mean** val_accuracy: 0.9688

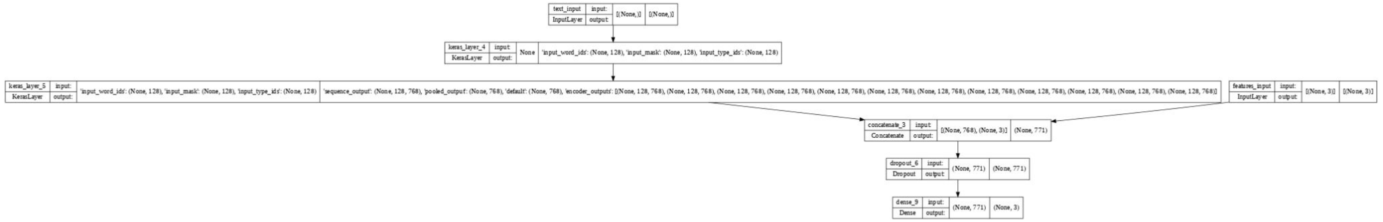
my model **max** accuracy: 1.0, **max** val_accuracy: 0.9771

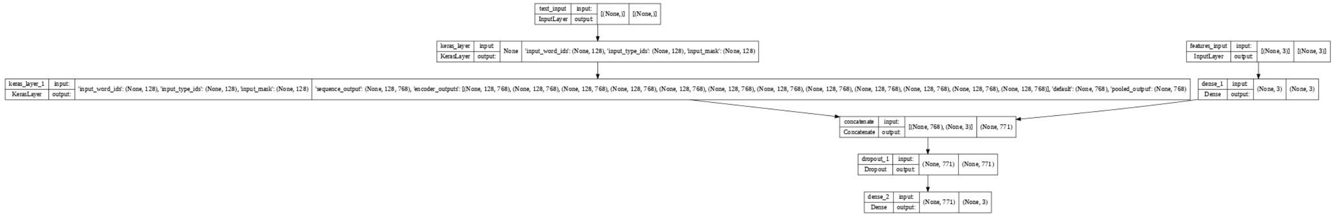
3.3.TF example model and my model, layers

TensorFlow example model:


text | input: | [(None,)] | [(None,)]
InputLayer | output
keras_layer | input: | None | 'input_type_ids': (None, 128), 'input_mask': (None, 128), 'input_word_ids': (None, 128)
KerasLayer | output:
keras_layer_1 | input: | 'input_type_ids': (None, 128), 'input_mask': (None, 128), 'input_word_ids': (None, 128) | 'pooled_output': (None, 768), 'default': (None, 768), 'sequence_output': (None, 128, 768), 'encoder_outputs': [(None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768)]
KerasLayer | output:
dropout | input: | (None, 768) | (None, 768)
Dropout | output
dense | input: | (None, 768) | (None, 3)
Dense | output:

My multi-input model, variant 1, 2:

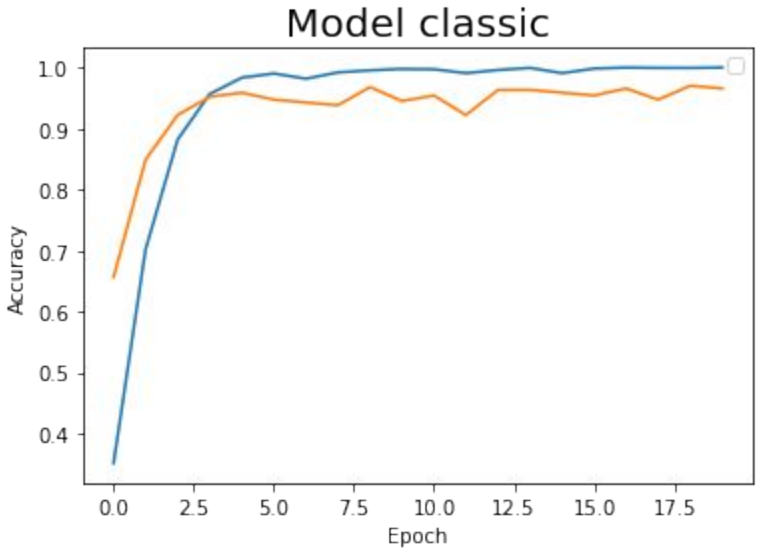

text_input | input: | [(None,)] | [(None,)]
InputLayer | output
keras_layer_4 | input: | None | 'input_word_ids': (None, 128), 'input_mask': (None, 128), 'input_type_ids': (None, 128)
KerasLayer | output:
keras_layer_5 | input: | 'input_word_ids': (None, 128), 'input_mask': (None, 128), 'input_type_ids': (None, 128) | 'sequence_output': (None, 128, 768), 'pooled_output': (None, 768), 'default': (None, 768), 'encoder_outputs': [(None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768)]
KerasLayer | output:
features_input | input: | [(None, 3)] | [(None, 3)]
InputLayer | output
concatenate_3 | input: | [(None, 768), (None, 3)] | (None, 771)
Concatenate | output:
dropout_6 | input: | (None, 771) | (None, 771)
Dropout | output
dense_9 | input: | (None, 771) | (None, 3)
Dense | output:


text_input | input: | [(None,)] | [(None,)]
InputLayer | output
keras_layer | input: | None | 'input_word_ids': (None, 128), 'input_type_ids': (None, 128), 'input_mask': (None, 128)
KerasLayer | output:
features_input | input: | [(None, 3)] | [(None, 3)]
InputLayer | output
keras_layer_1 | input: | 'input_word_ids': (None, 128), 'input_type_ids': (None, 128), 'input_mask': (None, 128) | 'sequence_output': (None, 128, 768), 'encoder_outputs': [(None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768)], 'default': (None, 768), 'pooled_output': (None, 768)
KerasLayer | output:
dense_1 | input: | (None, 3) | (None, 3)
Dense | output:
concatenate | input: | [(None, 768), (None, 3)] | (None, 771)
Concatenate | output
dropout_1 | input: | (None, 771) | (None, 771)
Dropout | output
dense_2 | input: | (None, 771) | (None, 3)
Dense | output:

3.4. An example of a TF model and my model, accuracy on an example model from the notebook

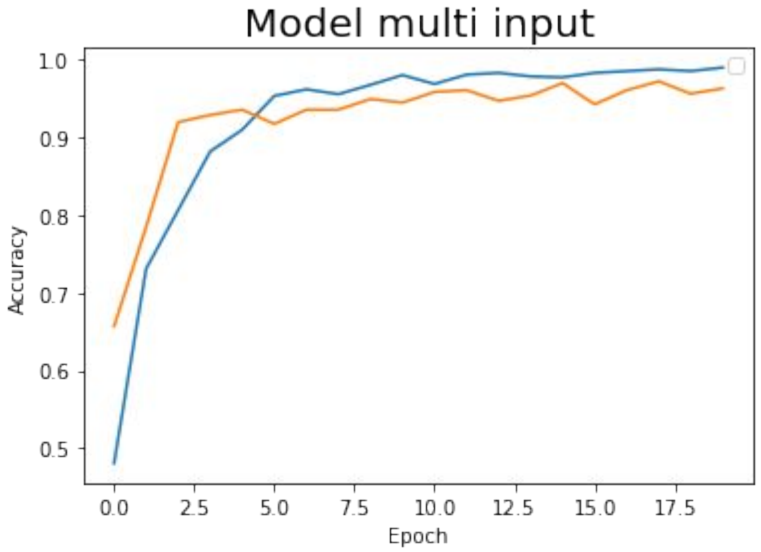
TensorFlow example model:

acc: 0.9994, val_acc: 0.9703



My multi-input model:

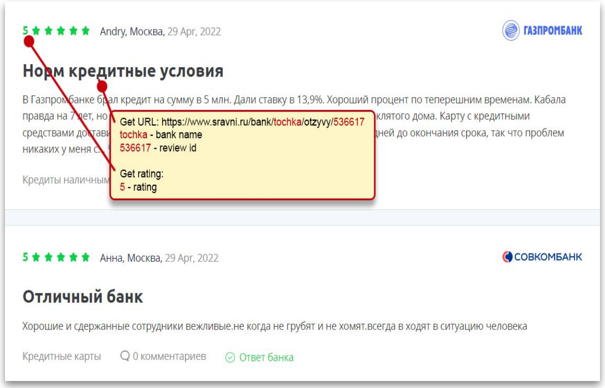
acc: 0.9966, val_acc: 0.9725



4. Dataset

4.1. Getting data

1.



5 ★★★★★ Andry, Москва, 29 Apr, 2022 ГАЗПРОМБАНК

Норм кредитные условия

В Газпромбанке брал кредит на сумму в 5 млн. Дали ставку в 13,9%. Хороший процент по теперешним временам. Кабала правда на 7 лет, но … клятого дома. Карту с кредитными средствами достави… дней до окончания срока, так что проблем никаких у меня с…

Кредиты наличным…

Get URL: https://www.sravni.ru/bank/tochka/otzyvy/536617
tochka - bank name
536617 - review id

Get rating:
5 - rating

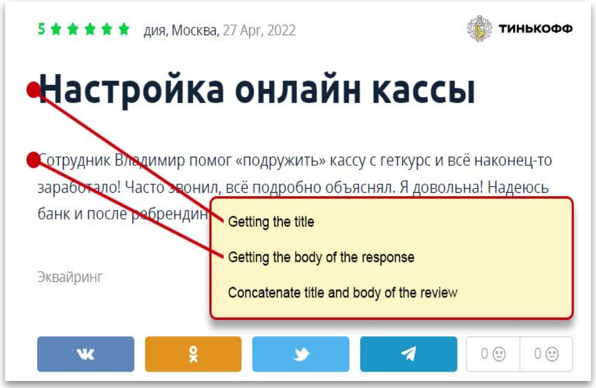
5 ★★★★★ Анна, Москва, 29 Apr, 2022 СОВКОМБАНК

Отличный банк

Хорошие и сдержанные сотрудники вежливые.не когда не грубят и не хомят.всегда в ходят в ситуацию человека

Кредитные карты 0 комментариев Ответ банка

2.



5 ★★★★★ дия, Москва, 27 Apr, 2022 ТИНЬКОФФ

Настройка онлайн кассы

Сотрудник Владимир помог «подружить» кассу с геткурс и всё наконец-то заработало! Часто звонил, всё подробно объяснял. Я довольна! Надеюсь банк и после ребрендин…

Эквайринг

Getting the title

Getting the body of the response

Concatenate title and body of the review

0 0

Data (reviews, names of banks, rating) are obtained from the site sravni.ru using parsing in 3 stages:

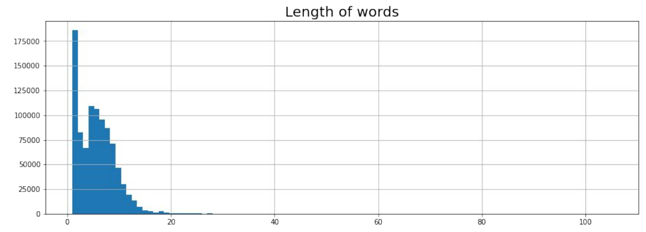
1. Parsing pages like https://www.sravni.ru/banki/otzyvy/?page=N, where N is the page number for parsing. We parse pages to obtain data in the format: id, link, bank, rating. Parsing is carried out in parts. We connect separate files and get the final dataset **list.csv**

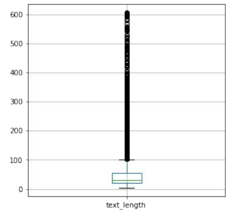
2. Parsing reviews from pages like https://www.sravni.ru/bank/B/otzyvy/ID, where B is the bank, ID is the review id. Links are taken from a previously generated dataset. We parse pages to get data in the format: 'id', 'text', 'bank', 'rating'. Parsing is carried out in parts.

3. Connect separate files and get the final dataset **data.csv**. Result: **list.csv**, **data.csv**

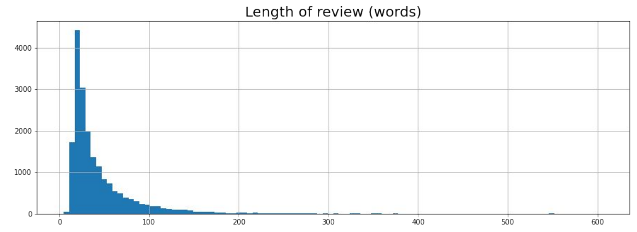
4.2. Text data analysis and class balancing

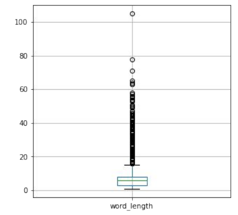
Word count in reviews



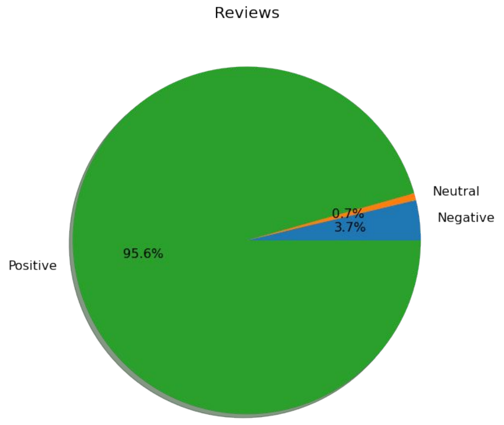


Number of letters in a word



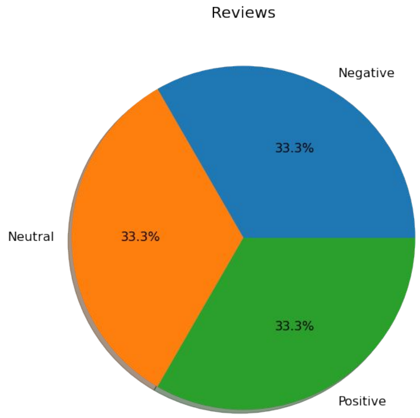


Classes are not balanced



Balancing classes

Classes are balanced

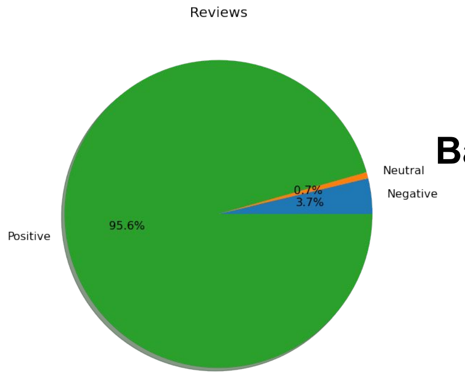


5. Experiment

5.1. Metrics

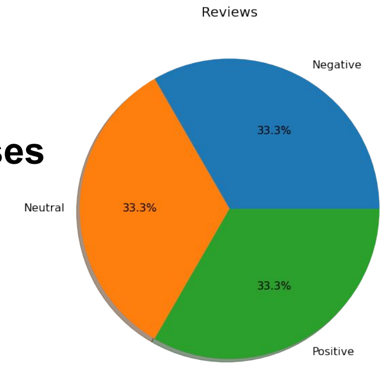
The processed data that we received by parsing is not balanced. Let's balance the classes. For a balanced dataset, we can use the metric “Accuracy”

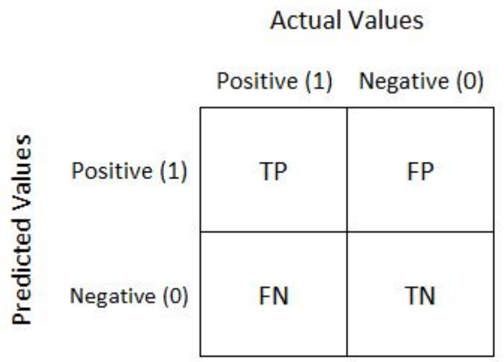
Dataset **data.csv** - bed
Classes are not balanced
Metriks: Precision, Recall, F1 Score



Balancing classes

Dataset **data_balanced.csv** - good
Classes are balanced
Metriks: Accuracy





Predicted Values		Actual Values: Positive (1)	Actual Values: Negative (0)
	Positive (1)	TP	FP
	Negative (0)	FN	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

5.1. Experiment setup

Dataset: **data_balanced.csv** (my data parsing from the site compare.ru). Dataset cleared of null, balanced, total number of records 1747, 3 classes

I will extract **3 additional features** from the dataset:

- comment length / max comment length
- for text uniqueness (in review)
- mean word length

We break it down into test and test samples.

IMPORTANT! Additional features come as an additional set of data supplied to a separate input, so random_state must be the same for both the data supplied to the BERT input and for these features.

Parameters:

- learning **rate: 1e-5**
- number of epochs: **30**
- the preprocessing model: **bert_multi_cased_preprocess/3**
- BERT model: **bert_multi_cased_L-12_H-768_A-12/4**
- metrics: **Accuracy**
- callbacks = **[checkpointer]** to keep the best weights

6. Results

№	TF model, acc	TF model, val_acc	My model, acc	My model, val_acc
1	0.9994	0.9703	0.9966	0.9725
2	1.0	0.9763	0.9891	0.9725
3	1.0	0.9703	1.0	0.9680
4	1.0	0.9634	0.9989	0.9771
5	0.9989	0.9611	0.9931	0.9611
6	1.0	0.9680	1.0	0.9703
7	0.9994	0.9680	0.9994	0.9634
8	1.0	0.9634	1.0	0.9657
Total	**0.9997**	**0.9676**	**0.9971**	**0.9688**

Unfortunately, there is no way to check on a large number of tests, so I draw conclusions only on my data.

7. Conclusion

My model trains slower because the extra features have significantly less accuracy than the "pooled_output" BERT, but later when the network goes into overfitting mode, the extra features increase the accuracy of the model a bit. However, the increase is very small and I did a small number of experiments - it is premature to draw conclusions. The difference in accuracy is in the noise level.

I tried a similar approach on convolutional networks, got an increase of exactly 0.3 - 0.4%. But there was another building - where there are a lot of swear words. Swear words are shorter, swear words are shorter than regular reviews, and this approach with additional inputs gave a tangible increase. In the case of BERT - and in this corpus of words - I am disappointed with this approach.

Conclusion - for each corpus of words it is necessary to select the architecture of the model individually.

About me

Domnenko A. N.

telegram: **@domnenko_a_n**
email: **domnenko@gmail.com**