TASK REPORT

Author: Tomáš Kliment (03.03.2022)

Task: Prepare a ML model which will be used to predict "user suggestion" category for future game reviews in production.

Available data:
- game_overview.csv
- sample_submission.csv
- test.csv
- train.csv

The goal of the task is to create a classifier that would be able to classify "user suggestion" (0 or 1) based on the information provided in train.csv (which are ordered in the columns). The trained classifier should then predict the "user suggestion" on the data listed in test.csv. This is therefore a classification problem.

I approached the task as follows:
- Analysis of information in the available data, in the delivered .csv files (A1 part).
- Data preprocessing (A2 part).
- Dividing the data from train.csv into two new sub datasets: my new train dataset and my new test dataset. Such a division was necessary to determine the metrics of selected algorithms (A3 part).
- I chose 2 different classifiers – two different ways to solve this task. Therefore, it will be possible to compare the results by two different approaches (B1 part).
- Classifier training on the new created training dataset (created from train.csv) (B1 part).
- The metrics was evaluated with both classifiers (based on my new test set - C1 part).
- A "user suggestion" was determined for the original test.csv by two independent classifiers.

More detailed information on solving these problems can be found below.

A1 – Analysis of delivered data

First, I analyzed whether the given classes: 0 and 1 as "user suggestion" are balanced. Class balance is important for classifier training, for correct prediction of "user suggestion" in both classes. The balance is as follows:
- "user suggestion" 0: 7526 from 17494 = 43 %
- "user suggestion" 1: 9968 from 17494 = 57 %

The classes are relatively well balanced. Due to this fact, "accuracy" could be chosen as the evaluation of the classifier. Other metrics such as recall, precision and f1-score will also be calculated. The data provided in train.csv contains 44 unique games (column "title") and 8 unique years ("column year"). The missing data were only in the "year" column (178 records were missing).

As information for classifier was determined:

- "user reviews" placed in train.csv – feature for classifier,
- "title" placed in train.csv – feature for classifier,
- "user suggestion" placed in train.csv – label for classifier.

Other information will not be used at this moment. The resulting "feature" for the classifier was created by combining two strings "title" and "user review".

A2 – Text data preprocessing

Feature "user revies" was preprocessed as follows:
1. string was converted to lowercase,
2. "early access review" was removed (I assumed that these words do not carry information about the sentiment),
3. words that have 15 or more letters have been removed (I considered them as a typing error),
4. punctuation was removed,
5. numbers were removed,
6. stop words were removed,
7. lemmatization was applied.

From "title" was only whitespaces removed. The result string as a feature for classifier training looked as follows:

"SakuraClicker: term free idle game get lot better honestly really want game particular ..."

A3 – Train and test dataset creation from original train data (train.csv)

As mentioned, the data in the original training file (train.csv) were divided into new train and new test datasets. In this way, a new test dataset was created, which also contained the necessary labels for classifiers metrics evaluation and was 15% of the original data.

B1 – classifiers description and training

To solve this problem, I decide to use two independent approaches - two classifiers:

- Linear SVC (support vector classifier) with TF-IDF Vectorizer (implemented from scikit-learn machine learning library).
- ANN (artificial neural network) classifier implemented as deep neural network with text vectorization on the first layer (implemented from tensorflow library).

Description of Linear SVC:

Preprocessed text data (listed in A2) must be converted to numeric format using the TF-IDF vectorizer. It combines 2 concepts, Term Frequency (TF) and Document Frequency (DF). Data converted in this way from text to numeric form is the input for Linear SVC. Subsequently, training using SVC was performed.

Description of ANN classifier:
It is a deep neural network that consists of several layers. The first layer contains an encoder that vectorizes the text into numeric form. The dictionary size was chosen as 3000 in this case. ANN also contains two layers where bidirectional LTSM units were used. They are followed by 3 Dense layers that have full connectivity. These 3 layers use a "relu" activation function. Each Dense layer is followed by a Dropout layer, whose job is to suppress the potential problem of overfitting. The last layer is the output layer. The network output is mapped to 2 discrete output classes of “user suggestion”: 0 and 1. To prevent overtraining, a stop criterion has been used.
Network training was completed after 8 epochs based on stop criteria.

C1 – classifiers metrics

The following table evaluates the metrics of both classifiers. The accuracy, recall, precision and f1-score for both algorithms were evaluated.

Tab. 1 – Metrics of the classifiers

Class	Precision		Recall		F1-score		Accuracy	
	SVC	ANN	SVC	ANN	SVC	ANN	SVC	ANN
0	0,86	0,84	0,82	0,83	0,84	0,83	0,87	0,86
1	0,87	0,87	0,90	0,88	0,88	0,87		

The classifiers were then used to predict “user suggestion” from the data in test.csv. There are two files available, one as Linear SVC output and the other as ANN classifier output.