Experiment Design for Data Sciene: Exercise 2

Paper Reproduction: Option 1

Konstantin Damnjanovic
01151948

Moritz Leidinger
11722966

Dzan Operta
11935976

ABSTRACT

In this project we are trying to reproduce the results of the paper *TUD-MMC at MediaEval 2016: Context of Experience task* [1], mainly the three tables featured. The paper is using a data set described by the paper *Right inflight? A dataset for exploring the automatic prediction of movies suitable for a watching situation. In Proceedings of the 7th International Conference on Multimedia Systems* [2]

1 DATA PREPROCESSING

The paper worked with different modalities:

- Audio data
- Visual data
- Metadata
- User ratings

The training and test data split was already done, as the corresponding files were in separate folder. However, finding the corresponding target values, the binary variable if the movie is good to watch on an airplane or not, was not as straight forward. While the spreadsheet that listed all movies that were in the dev set, contained the column 'goodforairplane', the file for the test set did not. In the whole folder structure there were some spreadsheets labeled 'test set' that contained the target value, but none of these had a complete match with the data files in the folders. We ended up merging the test data file with the 'dataset_complete.xslx', in the tab 'test', from the mediaeval folder. This leaves us with 95 movies in the training set and 223 movies in the test set, for a total of 318. The detail on the steps on data processing that was done really varied, so we will be discussing the different types of data separately in more detail.

1.1 Audio Data

The audio data was comprised of one file per movie, each having 13 rows. The number of columns changed from movie to movie, but was constant for each file. We assumed the length was some kind of measure per frame or other increment of the triler. This one was probably the most well described part of the data sets inthe paper, so the preprocessing was pretty staright forward. They mention each row represents a Mel-frequency cepstral coefficient (MFCC) and that they took the average per row, filling any missing value with 0. We did just that and formed 13 audio features per movie.

1.2 Visual Data

The visual data was not as transparent. Again provided was one file per movie, but this time with a constant dimension, 2 rows and 816 columns. The paper mentioned these being JCD, or Joint Composit Desriptors, which are used to describe and compare images. However, despite reading the corresponding papers, it was unclear how this related to the given values. They varied vastly per row, and the overall distributions per row were completely different between movies. The only consistency was that per column values were pretty consistent, so we had two approaches to deal with this part of the data set: Chosing all entries as or taking the average per column. This left us with 1652 or 826 visual features.

1.3 Textual Data

In this case we had one file, one for the test and one for the training set. From the file name, 'tf_idf ', we deducted that these spreadsheets were term frequency–inverse document frequency matrices, which describe the importance of keywords in text. There were thousands of different keywords in the headers, varying in number between the two sets. The valus stopped after the 95th and the 223rd coolumn, making it quite obvious that the matrix was supposed to be transposed and the movies added as row names. We added the movies in alphabetical order, since that made the most sense. Since we did not know what exactly the values meant, we turned each entry into a dummy variable, so it showed whether a keyword was important or not. At last we removed any keyword from the test data that was not contained in the training data and inserted any missing keyword with zeros for all movies. This left us with 3284 textual features.

1.4 Metadata

1.5 User Score

2 REPRODUCING TABLE 1

The first table to reproduce used a rule-based PART classifier in WEKA to evaluate Precision, Recall and F1 Score for User ratings, Visual, Metadata, Metadata + user rating and Metadata + visual. The result of the paper and ours in comparison can be seen in the following two tables. This was quite clearly outlined in the paper, however, the results were exactly the same as the first table in the *Right inflight?*-paper, which made us susppect that they just copying the table as a whole. Our solutions vary a bit from the table in the paper, especially user ratings, which had basically no result for precision and F1.

3 REPRODUCING TABLE 2

To reproduce table 2 we used scikit versions 0.22.1 and 0.18.2. First we initialized a list of 10 candidate classifiers with default parameters as stated in the paper. Next, we loaded four training datasets of different modalities; audio, textual, visual and metadata.

The paper states, that they employed the Las Vegas Wrapper, so we analyzed it, found some pseudocode examples and created the Las Vegas Wrapper function in python, which basically chooses the best performing subset of features based on f1 score in a stated number of iterations n.

Then, we built a pipeline, where we load 10 classifiers and run them on 4 different dataset modalities with 10-fold cross-validation as stated in paper and save their scores of precision, recall and f1. Next step was to run the Las Vegas Wrapper to try to find best feature subset for each of the combinations and try to improve the f1 score. Note about Las Vegas Wrapper is that we need to input a number of iterations which is not stated in paper. We tried with different options; 50, 100, 200, 1000, 2000, 10000, and the option with 100 iterations yielded the best results. The results where quite different on each run, as Las Vegas Wrapper introduces randomness. In the end, we only add classifiers and modalities which had higher precision, recall and f1-scores than 0.5. The results are over 30 different combinations.

4 REPRODUCING TABLE 3

When running the pipeline of table 2 for the classifiers who had precision, recall and f1-score above 0.5 we also stored information about the selected classifier, the modality and selected feature subspace. We used this array for the reproduction of table 3

The task for table 3 is to employ three different stacking methods; Majority Voting, Label Stacking and Label-Feature Stacking. It was not clearly stated in the paper if scikit was used for this step. Scikit version 0.18.2 has a VotingClassifier, while scikit version has also StackingClassifier, but it is not possible to run them with different feature subspaces, rather only on one dataset. Thus, we assumed that the authors of the papers manually created custom functions for the stacking methods.

For majority voting we implemented a function to fit multiple estimators, each on its selected feature subspace. As it is stated that voting is run with cross validation on training set, and on test set, we created two functions; cv_estimator and test_estimator. Both of them return an array of predictions on all previously selected classifiers. Then, we employ majority voting on the predictions based on the majority voting function from scikit source code.

For label stacking, it is stated in the paper that the authors generated a matrix of all predictions given by each selected classifier, on which they try to build a second-level classifier for the final predictions. It is stated in the paper that label stacking was again, run with cross validation on training set, and on test set. Thus, we used the same functions as for majority voting to build the label matrix. It is not stated in the paper which second-level classifier they used, so we randomly chose to use LogisticRegression.

For label attribute stacking, authors state that they combined the label matrix with features, but they do not state with which features did they combine the label matrix. As this lack of information created confusion for the implementation we did not finish this method.

5 CONCLUSION

The results for table 2 are over 30 combinations as opposed to 21 combinations stated in paper. It can be seen from the table that our results outperform the results in paper. Afterwards, we employed t-test on two independent samples, taking the f1 scores from the paper and our f1 scores as samples. We tried changing the scikit version to 0.18.2, changing the number of iterations of Las Vegas Wrapper, position of Las Vegas Wrapper in code, and employing different preprocessing strategies. By employing the t-test, in each case, that paper f1 scores and our reproduced f1 scores were significantly different.

We can conclude that we were not able to reproduce the results of table 2. The reasons are mainly connected to the lack of more elaborate description of the steps in the paper. Some of the reasons that lead us to negative results can be, running the framework on different machine, employing different preprocessing strategies, Las Vegas Wrapper number of iterations, Las Vegas Wrapper position in code and version of scikit.

statsmodels, ttest_ind Significance f1: table 1 (user rating left out since it was NA for us): $p = 0.24 \rightarrow$ not significant

table 2: $p = 0.67 \rightarrow$ not significant

table 3: $p = 0.07 \rightarrow$ not significant (at a level of 0.95)

REFERENCES

[1] Cynthia C. S Liem and Bo Wang. *TUD-MMC at MediaEval 2016: Context of Experience task*, 2016.

[2] M. Riegler, M. Larson, C. Spampinato, P. Halvorsen, M. Lux, J. Markussen, K. Pogorelov, C. Griwodz, and H. Stensland. *Right inflight? A dataset for exploring the automatic prediction of movies suitable for a watching situation. In Proceedings of the 7th International Conference on Multimedia Systems*, pages 45:1–45:6. ACM, 2016.

A TABLES FROM THE PAPER

Table 1 from the paper

Features used	Precision	Recall	F1
User rating	0.371	0.609	0.461
Visual	0.447	0.476	0.458
Metadata	0.524	0.516	0.519
Metadata + user rating	0.581	0.6	0.583
Metadata + visual	0.584	0.6	0.586

Table 2 from the paper

Classifier	Modality	Precision	Recall	F1
K-Nearest neighbor	metadata	0.607	0.654	0.63
Nearest mean classifier	metadata	0.603	0.579	0.591
Decision tree	metadata	0.538	0.591	0.563
Logistic regression	metadata	0.548	0.609	0.578
SVM (Gaussian Kernel)	metadata	0.501	0.672	0.574
Bagging	metadata	0.604	0.662	0.631
Random Forest	metadata	0.559	0.593	0.576
AdaBoost	metadata	0.511	0.563	0.536
Gradient Boosting Tree	metadata	0.544	0.596	0.569
Naive Bayes	textual	0.545	0.987	0.702
K-Nearest neighbor	textual	0.549	0.844	0.666
SVM (Gaussian Kernel)	textual	0.547	1	0.707
K-Nearest neighbor	visual	0.582	0.636	0.608
Decision tree	visual	0.521	0.55	0.535
Logistic regression	visual	0.616	0.6	0.608
SVM (Gaussian Kernel)	visual	0.511	0.67	0.58
Random Forest	visual	0.614	0.664	0.638
AdaBoost	visual	0.601	0.717	0.654
Gradient Boosting Tree	visual	0.561	0.616	0.587
Logistic regression	audio	0.507	0.597	0.546
Gradient Boosting Tree	audio	0.56	0.617	0.58

Table 3 from the paper

Stacking Strategy	Precision	Recall	F1
Voting (cv)	0.94	0.57	0.71
Label Stacking (cv)	0.72	0.86	0.78
Label Attribute Stacking (cv)	0.71	0.79	0.75
Voting (test)	0.62	0.8	0.7
Label Stacking (test)	0.62	0.9	0.73

B REPRODUCED TABLES

Table 1 reproduction

Features used	Precision	Recall	F1
User rating	very low	0.610	very low
Visual	0.545	0.520	0.526
Metadata	0.462	0.448	0.454
Metadata + user rating	0.478	0.462	0.468
Metadata + visual	0.528	0.511	0.517

Table 2 reproduction

Classifier	Modality	Precision	Recall	F1	BestF
k-Nearest neighbor	audio	0.506	0.617	0.55	10
Decision tree	audio	0.62	0.657	0.622	4
SVM (Gaussian Kernel)	audio	0.51	0.737	0.543	13
Random Forest	audio	0.526	0.583	0.545	13
AdaBoost	audio	0.602	0.6	0.591	4
Gradient Boosting Tree	audio	0.595	0.657	0.619	6
k-Nearest neighbor	textual	0.523	0.77	0.619	1543
Decision tree	textual	0.547	0.823	0.653	116
Logistic regression	textual	0.533	0.617	0.558	3282
SVM (Gaussian Kernel)	textual	0.527	0.903	0.664	3282
Bagging	textual	0.615	0.753	0.658	3282
Random Forest	textual	0.559	0.82	0.654	3282
AdaBoost	textual	0.677	0.887	0.756	2375
Gradient Boosting Tree	textual	0.669	0.863	0.744	1007
Naive Bayes	textual	0.591	0.73	0.645	3211
k-Nearest neighbor	visual	0.614	0.677	0.641	757
Decision tree	visual	0.6	0.553	0.555	16
Logistic regression	visual	0.617	0.697	0.633	826
SVM (Gaussian Kernel)	visual	0.577	0.92	0.708	568
Bagging	visual	0.677	0.58	0.614	826
Random Forest	visual	0.561	0.607	0.563	198
AdaBoost	visual	0.606	0.6	0.592	826
Gradient Boosting Tree	visual	0.605	0.617	0.602	826
Naive Bayes	visual	0.548	0.687	0.588	826
k-Nearest neighbor	metadata	0.579	0.557	0.56	75
Decision tree	metadata	0.558	0.517	0.533	39
Logistic regression	metadata	0.569	0.54	0.536	22
SVM (Gaussian Kernel)	metadata	0.548	1.0	0.707	75
Random Forest	metadata	0.527	0.553	0.526	75
AdaBoost	metadata	0.664	0.54	0.576	75
Gradient Boosting Tree	metadata	0.541	0.567	0.548	75

Table 3 reproduction

Stacking Strategy	Precision	Recall	F1
Voting (cv)	0.58	0.87	0.69
Label Stacking (cv)	0.61	0.6	0.565
Voting (test)	0.61	0.81	0.69
Label Stacking (test)	0.6	0.74	0.66