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]. We present our approach to the task, the steps we took as well as challenges we encountered on the way. Finally we present, that we were able to reproduce the tables of the original paper quite well (no significant differences), but also argue that the conclusions drawn in the original paper don't perfectly match up with ours.

1 DATA PREPROCESSING

The dataset we used was taken from a Dropbox folder found at [3]. From this folder only the content of the zip-file was used. The training and test data split was already done by the dataset providers, as the corresponding files were in separate folders. 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 left 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 (audio, visual, metadata, user rating) 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 trailer. This one was probably the most well described part of the data sets in the paper, so the preprocessing was pretty staright forward. The paper mentions 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 as the audio. Again one file was provided per movie, but this time with a constant dimension, 2 rows and 816 columns. The paper mentioned these being JCD, or Joint Composite Descriptors, which are used to describe and compare images. However, despite reading some papers on JCD, 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 features, so essentially appending the second row to the first one, or taking the average per column, which left us with 1652 or 826 visual features, respectively.

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 words (terms) in texts. There were thousands of different keywords in the headers, varying in number between training and test set. The values stopped after the 95th and the 223rd column for training and test set respectively, making it quite obvious that the matrix was supposed to be transposed and the movie titles added as row names. We decided to add them in alphabetical order, since that was the most obvious choice. Since the total values per movie varied a lot we decided to rather encode them in a binary fashion, replacing all non-zero values with 1. That way, any keyword important to a movie was represented as a 1, while all unimportant ones were as a 0. Finally we removed all columns (keywords) from the test data that were not contained in the training data and filled the test data with zeros for missing columns. This left us with 3284 textual features.

1.4 Metadata & user ratings

The provided metadata was contained in an XML file for each movie. These files contained a multitude of attributes of which only some were used by the original paper. For the metadata, on the one hand, those were 'language ', 'year ', 'genre ', 'country ', 'runtime ', 'rated '. The language, genre and country columns contained multiple values for several movies so we decided to split them up and multi-hot-encode them for all movies. Similarly, the year and rated columns were encoded, while the runtime was kept as an integer value. This resulted in a total of 77 columns for the metadata tables. On the other hand, only three attributes were picked from the XML file for the user ratings table. These three attributes ('metascore ', 'imdbRating 'and 'tomatoUserRating ') were chosen since they were mentioned in the paper accompaning the dataset [2]. All three of these values were kept as decimal values.

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 data, metadata, metadata + user ratings and metadata + visual data. The process to create this table was not described in the paper given to us [1], but was rather copied one-to-one from the dataset's paper [2]. In there however, it was outlined quite clearly how it was created. We wrote a python script to create the neccessary value combinations and export them to `.arff`-files to be loaded into WEKA. Compared to that, running the PART classifier in WEKA was very straightforward. Our results are shown in comparison to the original ones in appendix B and A, respectively. Our numbers vary only slightly from the table in the paper, especially in the case of user ratings, where we couldn't even compute the precision and F1 score, as the number of true positives was 0.

3 REPRODUCING TABLE 2

To reproduce table 2 we used `scikit-learn` 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 and loaded the four training datasets of different modalities (audio, textual, visual and metadata). Since the paper states, that a Las Vegas Wrapper (LVW) was employed, we implemented one based on pseudocode and a github repository we found on the web. The LVW basically chooses the best performing subset of features based on F1 score in a chosen number of iterations n. The number of iterations was not stated in paper, so we decided to try with different options (50, 100, 200, 1000, 2000, 10000) of which 100 iterations turned out to yield the best results in the end.

We built a pipeline, where we load the 10 classifiers and run them on the four datasets with 10-fold cross-validation as stated in paper and save their scores of precision, recall and F1. Afterwards we ran the LVW to try and find the best feature subset for each of the combinations based on the F1 score. The results were quite different on each run, as the LVW introduces randomness. In the end, only those classifiers and modalities were selected which had precision, recall and F1-scores higher than 0.5. The results are over 30 different combinations, which are shown in comparison to the original ones in appendix A and B.

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 was to employ three different stacking methods: Majority Voting, Label Stacking and Label-Feature Stacking. It was not clearly stated in the paper if `scikit-learn` was again used for this step, but we assumed so. Scikit's version 0.18.2 has a VotingClassifier, while scikit version 0.22.1 also has a 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 therefore 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 the training set and on the test set, we created two functions: `cv_estimator` and `test_estimator`. Both of them return an array of predictions on all previously selected classifiers. We then employ majority voting on the predictions based on the majority voting function from scikit's 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 then training and test sets. 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 opted to use LogisticRegression.

For label attribute stacking, the authors of the original paper state that they combined the label matrix with features, but they do not state with which features did they used. This lack of information made it impossible for us to finish this part of the reproduction.

The results of all methods (except Label Attribute Stacking) are compared to the original ones in appendix A and B.

5 CONCLUSION

Just looking at the tables we created and comparing them to the original ones gave us the impression that our efforts weren't completely pointless after all. The numbers weren't perfect, but still seemed somehow in the range of the original ones. To confirm that suspicion, we compared our F1 scores to the original with two-sided independent t-tests (taken from the `statsmodels` package). For table 1 the test resulted in a p-value of 0.24, which confirmed that we shouldn't reject the H_0 ($\mu_0 = \mu_1$). This was also true for table 2 where the test resulted in a p-value of 0.67. In table 3 the p-value of 0.07 turned out to be just shy of a significant difference at level 0.95, which we attributed mostly to our voting classifier performing poorly and out label attribute stacking approach missing entirely (this approach was left out of the significance testing completely).

We can conclude, that we were able to generally reproduce the results of table 1 and 2 and the conclusions drawn from them (which modalities and classifiers to use), but didn't get very satisfing results for table 3. We attribute this mostly to the amount of information missing from the paper, especially concerning this table.

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.

[3] https://www.dropbox.com/sh/j7nunecnzfjrp2r/AAC1BAf5JEv-rGUW9h02L2X2a?dl=0

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	-	0.610	-
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
k-Nearest neighbor	audio	0.506	0.617	0.55
Decision tree	audio	0.62	0.657	0.622
SVM (Gaussian Kernel)	audio	0.51	0.737	0.543
Random Forest	audio	0.526	0.583	0.545
AdaBoost	audio	0.602	0.6	0.591
Gradient Boosting Tree	audio	0.595	0.657	0.619
k-Nearest neighbor	textual	0.523	0.77	0.619
Decision tree	textual	0.547	0.823	0.653
Logistic regression	textual	0.533	0.617	0.558
SVM (Gaussian Kernel)	textual	0.527	0.903	0.664
Bagging	textual	0.615	0.753	0.658
Random Forest	textual	0.559	0.82	0.654
AdaBoost	textual	0.677	0.887	0.756
Gradient Boosting Tree	textual	0.669	0.863	0.744
Naive Bayes	textual	0.591	0.73	0.645
k-Nearest neighbor	visual	0.614	0.677	0.641
Decision tree	visual	0.6	0.553	0.555
Logistic regression	visual	0.617	0.697	0.633
SVM (Gaussian Kernel)	visual	0.577	0.92	0.708
Bagging	visual	0.677	0.58	0.614
Random Forest	visual	0.561	0.607	0.563
AdaBoost	visual	0.606	0.6	0.592
Gradient Boosting Tree	visual	0.605	0.617	0.602
Naive Bayes	visual	0.548	0.687	0.588
k-Nearest neighbor	metadata	0.579	0.557	0.56
Decision tree	metadata	0.558	0.517	0.533
Logistic regression	metadata	0.569	0.54	0.536
SVM (Gaussian Kernel)	metadata	0.548	1.0	0.707
Random Forest	metadata	0.527	0.553	0.526
AdaBoost	metadata	0.664	0.54	0.576
Gradient Boosting Tree	metadata	0.541	0.567	0.548

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