Text as Data Coursework

2577007r

March 2023

Q1 - Dataset

(a) The reasoning behind picking this dataset, and how could automatic labelling of this dataset be used in practice

My dataset consists of movies, along with their genre and description. I picked this dataset because I'm a big cinephile myself and always enjoy finding out new movies.
This model aims to predict a movie's genre based on its description, and could be used in practice as a recommendations system.

(b) Preprocessing of the data and the labels

The labels that are going to be predict are movie genres. The dataset itself contains more than 10 labels, and therefore I picked 9 labels myself and grouped the others into an 'Other' label.
Moreover, because the dataset also contains foreign movies, with descriptions written in other languages, I will use an external library to eliminate non-English movies. This task is performed in a separate Jupyter Notebook named 'get_Eng_movies' which will create a new file named 'Eng_movies.txt'.
The text used for classification will only be the movies' descriptions, because as stated previously, some titles are not in English.

(c) Splitting the data

The dataset on Kaggle is already split into train/test, but these splits are way bigger than the 10000 limit. Therefore, I only use one file from those provided, namely the train split, because it also has genres in it, and create my own 60/20/20 splits.

Split	Thriller	Drama	Horror	Western	Comedy	Action	Sci-fi	Romance	Adventure	Other
Train	598	596	582	611	624	600	389	387	484	580
Validation	187	208	217	193	177	209	134	138	139	215
Test	215	196	201	196	199	191	123	144	147	205

Table 1: Label counts for data splits

The distribution seems to be quite uniform within each split, with some slight exceptions. The three genres that did not have at least 1000 movies (Romance, Adventure and Sci-fi) maintain the somewhat uniform distribution within the validation and test split, whereas they fail to do so within the train split.
Finally, the splitting has ensured that the 'other' genre does not overtake the other genres.

Q2 - K-means

(a) Few documents example for each cluster, along with top 5 tokens with the highest magnitude in each centroid

The examples I will provide are based on the current clusters displayed in the notebook.

Cluster 1: Movie 1 - Gallagher's Travels (action). Movie 2 - Rodina Zhydot (action).
Top 5 tokens with highest magnitude: 'world', 'earth', 'team', 'alien', 'human'.

Cluster 2: Movie 1 - White Cargo (comedy). Movie 2 - Paranorma There is Always a Dark Side (horror).
Top 5 tokens with highest magnitude: 'film', 'story', 'movie', 'world', 'life'.

Cluster 3: Movie 1 - Khatra (thriller). Movie 2 - Path of Egress (thriller).
Top 5 tokens with highest magnitude: 'man', 'murder', 'kill', 'find', 'wife'.

Cluster 4: Movie 1 - The Fargo Kid (western). Movie 2 - Taiyo no hakaba (drama).
Top 5 tokens with highest magnitude: 'gang', 'ranch', 'town', 'sheriff', 'kill'.

Cluster 5: Movie 1 - Second Best (drama). Movie 2 - The Demons Within (horror).
Top 5 tokens with highest magnitude: 'love', 'life', 'girl', 'friend', 'family'.

(b) Cluster topics

The clusters from part (a) seem to make sense to some extent. The 2nd, 3rd and 5th clusters pick up on general words that are present in many movie descriptions, be it drama, romance or thriller. What's interesting is that the 1st cluster has words that are present mainly in Sci-fi movies (alien, human, earth), and that the 4th cluster has western-related tokens with highest magnitude (ranch, town, sheriff).

(c) Confusion matrix

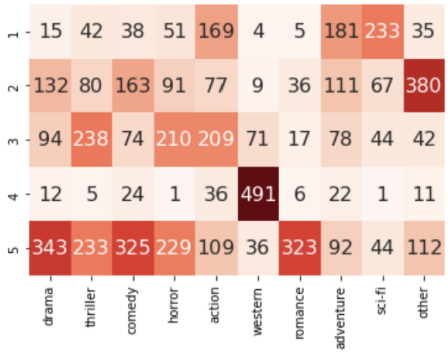

Figure 1: Confusion matrix for K-means clustering

(d) Trends that arise from the confusion matrix

As anticipated in part b), the confusion matrix shows that Cluster 4 is able to pick up very well on Western movies, and Cluster 1 picks up most of the Sci-fi movies. However, Cluster 5 picks up on many labels, such as drama, thriller, comedy, horror and romance, but this is not completely unexpected because these genres could have similar descriptions. What's interesting is that Cluster 3 also picks up many of the thriller, horror and action movies, which is as expected because these genres have similar plots usually.

Q3 - Comparing Classifiers

(a) Five baseline classifiers and their metrics

Model	Accuracy	Macro precision	Macro recall	Macro F1
Dummy Classifier with strategy most_frequent	0.097	0.01	0.1	0.018
Dummy Classifier with strategy stratified	0.088	0.085	0.085	0.085
Logistic Regression with One-hot vectorization	0.534	0.542	0.53	0.535
Logistic Regression with TF-IDF vectorization	**0.561**	**0.574**	**0.55**	**0.555**
SVC with One-hot vectorization (SVM with RBF kernel)	0.511	0.541	0.5	0.507

Table 2: Classifiers' metrics on validation data

Model	Accuracy	Macro precision	Macro recall	Macro F1
Dummy Classifier with strategy most_frequent	0.114	0.011	0.1	0.021
Dummy Classifier with strategy stratified	0.106	0.104	0.104	0.104
Logistic Regression with One-hot vectorization	1.0	1.0	1.0	1.0
Logistic Regression with TF-IDF vectorization	0.912	0.914	0.907	0.91
SVC with One-hot vectorization (SVM with RBF kernel)	0.953	0.954	0.951	0.952

Table 3: Classifiers' metrics on training data

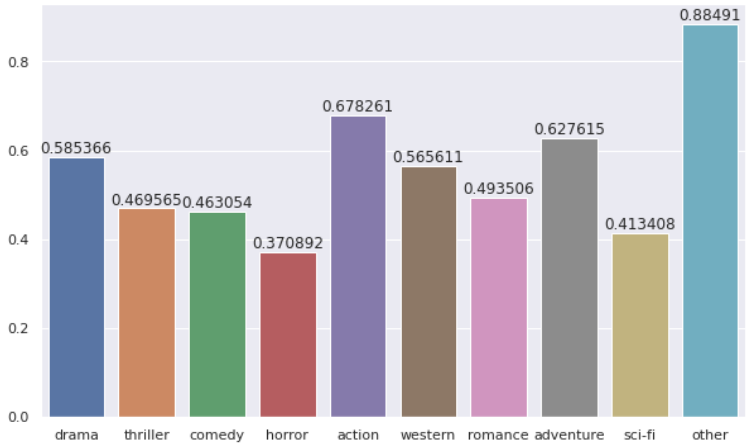

Figure 2: Bar plot of F1 scores from Logistic Regression with TF-IDF

The best classifier is Logistic Regression with TF-IDF vectorization. The dummy classifiers perform quite poor, but this is expected because their predictions are either all 'comedy' (for the most_frequent dummy classifier), or stochastic (for the stratified dummy classifier). The SVC performs quite close to both Logistic Regression implementations (both with One-hot and TF-IDF). All models, excluding the dummy ones, overfit on the training data (judging by their close to, or even equal to 1 F1 scores), which is expected, but it's good that they don't underfit. However, I was expecting better performance from the SVM since these are usually the way to go for text data.
Finally, as it can be seen from the bar plot, Logistic Regression with TF-IDF does pretty well for most of the labels, but tends to underfit for the horror and sci-fi labels. These F1 scores are generally good because when talking about movies, usually one movie falls within multiple genres, and therefore there is not one exact correct genre for each movie. With that being said, there are also some flaws in the splits, because many labels are all grouped into the 'other' label, thus the 0.88 F1 score for this label. Further improvement could be done by using a bigger vocabulary when vectorizing, and also by tuning the parameters of the Logistic Regression model.

(b) Building my own classifier

For this part I firstly implemented a new tokenizer. I experimented to see whether proper nouns have negative impact on the results, and if numbers (that could in turn denote years) would give better metrics. Out of these two features, removing proper nouns decreased the F1 scores, whereas keeping the tokens that contain digits increased the performance both for Logistic Regression and SVC.

As for the different classifiers, I've experimented with multiple classifiers, such as: Logistic Regression, SVC with both RBF and sigmoid kernels, Naive Bayes and Random Forests.

The classifier I will pick is SVC with sigmoid kernel with TF-IDF vectorization, where tokens containing digits are kept this time. I made this decision because the metrics are quite close to the best performing classifiers from part a) - **0.562 accuracy, 0.578 precision, 0.556 recall and 0.563 F1** - way better than the Dummy Classifiers, and comparable to the previous SVC and Logistic Regression implementations, but mainly because the F1 score is greater by close to 1%, which may not seem much but after many tries to improve the metrics, this yields the best result. However, one downside might be that SVC is slower than Logistic Regression when it comes to larger datasets, since the optimization problem is not linear anymore.

The final point that made me pick SVC is researching about different models and seeing that Support Vector Machines (SVM) algorithms are usually preferred when dealing with text data.

Q4 - Parameter Tuning

The parameters I will tune are **sublinear_tf and max_features of the vectorizer, and the solver and C value used by the classifier.**
For the **C value**, I tried using all powers of 10 ranging from 10^{-3} to 10^{5}. If the C value was lower than 1, the metrics decreased significantly, whereas for values greater than 1 the F1 scores increases for C=10, but not considerable. I believe that this happens because when C value is less than 1, the model is too constrained by the regularization, resulting in underfitting. When the C value is greater than 10, this usually leads to overfitting on the training data, thus the poor results on the validation data.

For the **sublinear_tf**, setting this to True yields slightly better metrics. This probably happens because the texts have many words that are frequent but not that important. By setting this parameter to True, the weight given to frequent tokens within a document becomes lower.

For the solvers, I did an exhaustive search trying out every solver along with every different C value listed above. The best solver was **Newton-CG with C value = 10**, yielding an increase of about 1% in the F1 score. The best solver usually comes down to many factors, such as sample size, regularization and the features. What's interesting is that when tweaking the solver, the best one is Saga, but when searching through all the combinations of solvers and C values, the best one becomes Newton-CG with C-value=10.

For the max_features parameter, since my corpus has about 27k tokens, I have tried several values such as 25k, 20k, 26k and 18k. The parameters for the Logistic Regression were C-value=1 and solver=Newton-CG, because these performed the best from previous experiments. However, none of the limits for the vocabulary size increased the F1 score, but didn't decrease it by a large amount either, about 0.5%. This shows that there are not many "stopwords" in the documents - words that occur often but don't have huge significance.

The table below only shows some of the results obtained by the exhaustive search I have conducted. This is because some tweaks did not yield interesting results. All the results are present in the notebook.

Parameters	Accuracy	Macro precision	Macro recall	Macro F1
sublinear_tf = True	0.577	0.586	0.565	0.569
sublinear_tf = True, C = 0.01	0.216	0.273	0.21	0.130
sublinear_tf = True, C = 10, solver='newton-cg'	**0.575**	**0.581**	**0.570**	**0.575**
same as above & max_features = 25k	0.570	0.577	0.566	0.570
same as above & max_features = 20k	0.572	0.578	0.567	0.571

Table 4: Parameter tuning for Logistic Regression with TF-IDF

As a conclusion, the increase in performance is not significant probably due to the inconsistency of the dataset.

Q5 - Context vectors using BERT

(a) Using first context vectors

Although this implementation takes quite a long time to run (approx. 30 minutes), it offers a boost of roughly 5% in the F1 score, which is quite significant considering how tuning the parameters in Q4 only gave 2% extra increase in F1 score.
The evaluation metrics are: **accuracy = 0.634, precision = 0.632, recall = 0.628, and F1 score = 0.629.**

(b) Training an end-to-end classifier

For this part, the classifier performed decently compared to the others.
The evaluation metrics are: **accuracy = 0.622, precision = 0.623, recall = 0.623, F1 score = 0.622.**

(c) Tuning hyperparameters for the end-to-end classifier

For this part, because I was running out of GPU memory frequently, I chose to tune one hyperparameter for the first two modified trainers, and then for the last one I modified two hyperparameters.
Due to the dataset being quite large, I was unable to modify the parameters to see whether the F1 score could become very good (mainly because Google only gives limited amount of computational units).

Hyperparameters	Accuracy	Macro precision	Macro recall	Macro F1
default	0.622	0.623	0.623	0.622
epochs = 4	0.622	0.623	0.623	0.622
batch_size = 6	0.114	0.011	0.1	0.021
batch_size = 10, learning_rate = 1e-2	0.103	0.010	0.1	0.019

Table 5: Hyperparameter tuning for roberta-base classifier

(d) Findings

The classifier that performed the best, from part Q5, is the one in part a) that uses the pipeline. With that being said, the difference between the classifier in part a) and the one in part b) is not significant, about 0.007 difference in the F1 scores.
I believe that this happens because the train set is not that big, and the model does not have enough data to train in order to achieve better performance, compared to the one in part a) which is pre-trained on a large dataset.

What's also interesting is that the classifier in part b), and the first classifier in part c) - the one with epoch = 4, have the exact same metrics. This could happen, in my opinion, for two reasons:
Reason 1: The model converges after one epoch, so after that any other epochs don't improve the classifier at all.
Reason 2: Maybe by increasing the epochs the model starts overfitting on the train data, and then when used on other unseen data it does not show any improvements.

Finally, I believe that the model from parts b) and c) could yield better results than the one in part a) by choosing optimal hyperparameters. However, this is very computationally expensive (could not even run the default model on my 4GB of VRAM GPu), and the increase in the F1 score might be minimal.

Q6 - Conclusions and Future Work

(a) Evaluation on the test set

For this part I will use the classifier that only uses the context vectors of the starting token for each movie.
The metrics are: **accuracy = 0.620, precision = 0.617, recall = 0.614, F1 score = 0.614**

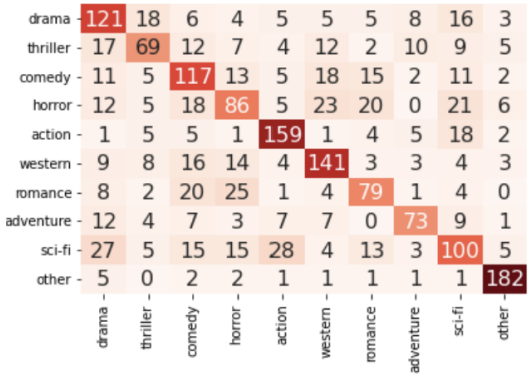

Figure 3: Confusion matrix for the test dataset

(b) Common classification errors

Overall, the classifier predicts pretty well for each genre. However, the first classification errors that can be seen is slight class imbalance, judging by the high false negatives and false positives in the following genres: Sci-fi, romance, and horror.

Secondly, after analyzing the data set, I saw some spelling mistakes, some foreign names and also some special characters that the tokenizers cannot really pick up. Also, if the entire data set was used (approx. 52,000 movies), the model would have probably been better trained.

Finally, due to the constraints on the label count, many movie genres have all been grouped within the 'other' category, which makes the model overfit on the 'other' label.

(c) Overall performance

I believe that the final performance is good enough for the stated purpose of this classifier because it predicts pretty well movies that the train set had more of, and only fails to predict with high accuracy some genres that were sparser, i.e. horror, sci-fi, thriller. With that being said, if this model were to be deployed, it would probably be a good idea to train it on a larger dataset and also transform it into a multilabel classifier.

Both false positives (predicting a movie is a specific genre when it is actually not), and false negatives (fail to predict the genre of a movie), if this system was deployed as a form of recommendation system, would probably determine the user to watch a movie he will not enjoy. For example, since the Sci-fi F1 score is the lowest (Figure 4), if a user would watch many Sci-fi movies, then they might be recommended (with good probability) to watch a horror, action or drama movie, which has nothing to do with the sci-fi genre.
This could also happen when it comes to a horror movie by displaying a romance movie instead.

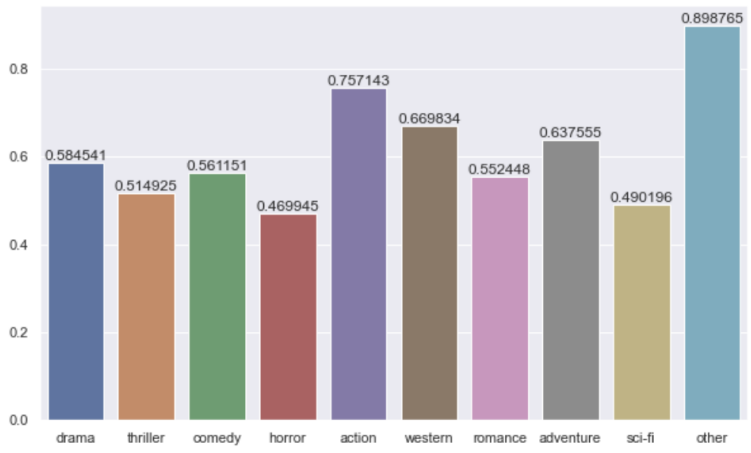

Figure 4: Bar plot of F1 scores on the test set

(d) Negative societal effects

I don't think that the deployment of this system would have tremendous negative societal effects. This is mainly because this type of model is used often nowadays in recommendation systems. For example, Netflix probably uses a similar model to recommend movies to people.
However, there are some negative societal effects worth mentioning.

The first one would be that this type of system probably uses some type of cookies, or shared preferences in order to collect data from users, and most of the time users do not bother to read the GDPR policies and might be unaware of the usage of such systems.

The second one would arise from the dataset itself, by adding biases and maybe picking up some patterns that would marginalize groups of people, etc, and even add bias.

Although there are these negative effects, I believe that a well-trained system's benefits would outweight the negative effects.

(e) Further improvement

Since the model used in part a) is based on the roberta-base model, a larger vocabulary is not needed. However, the first step would be adding more data to the train set, and also checking for any words that appear too often in the documents but don't hold that much information.

Furthermore, instead of using only the first context vectors, maybe the metrics of the model would improve if all context vectors were used - this is usually the case, but not always.

Finally, this problem could also be further developed into a multilabel classification problem, rather than a multi-class problem, because, as said earlier, one movie falls within multiple genres. This would transform some wrong predictions into slightly correct or even correct predictions.

(f) How much time was spent on this assessment

This assignment has taken me approximately 35 hours. However, this was mainly because I had to re-run all the cells when starting to work on the project again. Furthermore, because of my 4GB of VRAM GPU, I had to switch between Google Colab and Jupyter Notebook to run different parts of the assessment.
Overall this was a very insightful assignment, especially for those interested in data science.