Text Analytics Report

Inspecting the Relationship between Film Plots and their Associated Genres

Jacopo Gasparro (620038)
Roberto Cannarella (616400)

Master's Degree in *Digital Humanities* (LM-43)

Contents

List of Figures

1. Introduction

Text analytics tasks typically involve dealing with samples of text produced in natural language, i.e. with unstructured data, and extracting structured, more "controlled" information out of it. Sources and kinds of text may be of different natures, and so the practical applications made possible by the specific task. *Text classification*, for instance, consists in analysing textual data and linking such data to predefined groups (classes). It is an essential task in text analytics, since it can be 'reinterpreted' and readapted for accomplishing domain-specific aims (for instance, emotion detection). In this work, classification algorithms and architectures will be used for predicting a film genre starting from the contents of the film's plot (a sub-task that may be called *plot classification*).

The task can be of interest because it allows to determine whether there is a correlation between the plot of a narrative and the genre, or genres, associated with it. If such a correlation holds, then it can be used for different applications. For instance, on a streaming service it may be of use to determine automatically, and to provide to users, the genre(s) associated to a film starting from the content of its plot. Conversely, the conjunction of different genres may be used for automatically generating new plots, which may be useful to screenwriters.

Moreover, an advantage of using plots as starting textual data is in that their textual features are fairly consistent - for instance, the fact that they *report* some narrative implies that their syntax is as rigid as simple (relying

on parataxis rather than on hypotaxis), and that the vocabulary employed in them mainly relies on lemmas with concrete entities and actions as referents. On the other hand, genres are not always reliable as labels. Any plot-genre association is at some level arbitrary, in the sense that different people may, in some cases, associate different genres to the very same movie. This is true in particular because genre association usually consists in the *combination* of different genres, which augments the internal variability - for instance, the same film may be described solely as "Drama" by some people and as "Drama, Action" by other people. In any case, this variability is a recurring, typical issue when working with any labelled data[1], and the labelled data used in this work can still be considered to be reliable.

This report is structured as follows. Chapter 2 describes the dataset that has been employed for the experiments and summarizes how the data into it have been obtained and preprocessed. In chapter 3, the results of the text classification achieved by using different learning algorithms will be reported. Finally, chapter 4 will report the results of using the dataset for the creation of document embeddings, with the aim of calculating the similarity between the documents.

[1]The multi-annotator agreement being low is so inherently a danger in the process of manual annotation, that it can be a problem even for a fundamental task such as syntactic annotation.

2. Dataset preparation

This section covers the content of the `movies_api_scraping.py` file and of the `Dataset_preprocessing` notebook. The Python script has been created for accessing the YTS database[1] by the interaction with the site APIs. Of each film, the plot, the title and the associated genres have been retrieved and stored.

After dropping the duplicates instances, the so constructed dataset contains information on 37.567 films. A simple preprocessing is applied to each genre for removing the square brackets and the single quote marks among which they were presented. Furthermore, after inspecting the set of the present genres, it has been decided to drop those values not really referring to film genres yet present in the dataset (for instance, "Game-Show"). At this point, the final dataset contains information on 33.812 films, having as genre(s) one or more of the following (22) values: "Action", "Adventure", "Animation", "Biography", "Comedy", "Crime", "Documentary", "Drama", "Family", "Fantasy", "Film-Noir", "History", "Horror", "Music", "Musical", "Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western".

[1] `https://yts.mx`

3. Classification

The classification problem under discussion is a quite complex one, since it is both *multiclass* and *multilabel*. It is multiclass, since the given input (i.e. the given plot) can belong to any of the 22 (after some further preprocessing, 15) classes/genres; it is multilabel, since the given input can belong to one or more of such classes. It is complex also in the sense that it is *concretely* hard to be accomplished: inferring a narrative's genres starting from a (less or more detailed) resume of its content is not always trivial a task, and that is true also for humans. The non-triviality of it is due not only to the fact that one should choose whether to assign one or more genre(s) to the film (in the latter case, one should also decide *how many* of them are to be assigned), but also due to the fact that genres such as "drama" are less distinctive, thus less easily recognizable, than others (such as "western").

3.1 Classification with SVM

Classification was first experimented by using a model trained through an SVM (Support Vector Machine). In particular, the `LinearSVC` and the `MultiOutputClassifier` modules from scikit-learn[1] have been employed for building a multilabel SVC machine. The related notebook is `SVC_multi-label_classification.ipynb`. The notebook opens with some further,

[1] `https://scikit-learn.org/stable/index.html`

basic preprocessing[2]: in order to reduce the number of output classes, nearby/similar genres are grouped together and counted as one (in this way reducing the number of classes from 22 to 15). Some choices (for instance, having decided to consider Sci-Fi and Fantasy as one category) are inspired by the categories used on Netflix[3]. Additionally, the Pandas[4] function `get_dummies` has been used for creating a one-hot representation of the genres, to be used as the label.

The dataset has then been split into two subsets, one for the training process (80% of the original) and one to be used as a test set (20%). Text indexing has been conducted using an English spaCy[5] pretrained model. Only minimal text features have been taken into consideration, i.e. the lemmas and the bigrams of the lemmas. Stopwords have been skipped, and so have those tokens that contain a person's name, this latter choice being led by the fact that character's names are hardly a hint of the film's genre. In general, we have decided to focus only on the *vocabulary* of the documents (ignoring information such as that at the morphosyntactic or at the syntactic level) since the semantic content of the plot can be expected to be the main clue of a film's genre (and not, for instance, its syntax). Intuitively, a western film can be more easily identified thanks to the presence of the word type "cowboy" than for information related to parts of speech. This intuition was confirmed by some trial runs, in which

[2]Note that this phase is repeated at the beginning of the notebooks about LSTM and BERT.

[3]`https://www.whats-on-netflix.com/news/the-netflix-id-every-category-on-netflix/`

[4]`https://pandas.pydata.org`

[5]`https://spacy.io/`

other information such as parts of speech were represented in the feature space, and were found not to be relevant.

Text has been, thus, vectorized according to such features. A grid search has then been conducted in order to identify the optimal model, which has turned out to have the C hyperparamater = 0.1 and to make use of 5000 distinct features. The trained model has finally been tested on the residual data. It has been evaluated by using the F1 score, which is the harmonic mean of precision and recall and is calculated as:

$$F1 = \frac{2 * TP}{2TP + FP + FN}$$

In particular, the F1 score has been computed as micro F1 score and as macro F1 score. The difference is that, in the micro case, the cardinality of the classes is taken in consideration, so that if the examples of a numerous class are misclassified, this has a consequence on the final value. This makes the micro F1 score suitable for cases (such as the one under analysis) where some classes are more represented than others. In the macro case, instead, all classes are treated equally (since the evaluation measures are computed distinctly for each class, and the final evaluation is calculated as their average) and equally contribute to defining the final value. The micro F1 score reached by the model is of 64%, the macro is of 50%. The difference is easily explainable if one considers the confusion matrix of the notebook: it can be noted that a class such as "Action & Adventure", which has a high number of positive values (5090), is correctly classified 93% of the times. Keeping in mind that the number of output classes is quite robust, such results can be considered to be satisfying.

Note that the final cell of the notebook allows to inspect which of the

considered features result to have been more informative/useful for the discriminative capacities of the model (i.e. have higher weights). One can see that lemmas and bigrams of lemmas such as "World War", "coach" and "cowboy" have high weights (since they can be of use for detecting, respectively, war, sports and western films), which is not surprising as a result but confirms that the defined feature space works well.

3.2 Classification with LSTM

Classification was experimented by training a Long short-term memory (LSTM), which, for its being a recurrent neural network, is suitable for dealing with consistent, cohesive texts. The dataset has undergone the same preprocessing defined for SVC. Tokenization has been conducted through the Keras `Tokenizer` object, with each embedding having been set to have a dimension of 300. A `GloVe` embedding (also with dimensionality = 300) containing the representation of 400k different words[6] has been loaded and used for creating an embedding matrix representing the previously tokenized 20k words.

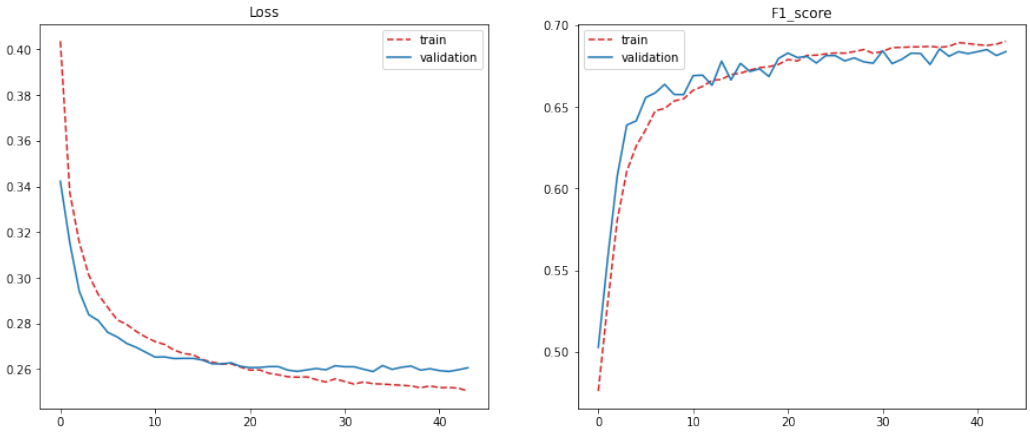
The architecture of the network is thus composed of the embedding layer, a LSTM layer with 40 nodes, a Dropout layer (which regularizes the network and avoids it to overfit) and a dense, final layer with sigmoid as activation function. The employed loss measure is the binary cross entropy/logistic loss, whereas Keras `Adam`[7] has been used as an optimizer. The maximum number of epochs has been set to 50, but an early stopping

[6] `https://huggingface.co/stanfordnlp/glove/resolve/main/glove.6B.zip`.

[7] `https://keras.io/api/optimizers/adam/`

threshold has been introduced on the validation error (again, for avoiding the model to overfit on the training data). Below (3.1) is the plot of the learning curves.

Figure 3.1: Learning curves (loss, f1 score)



The learning curves are generally smooth, and the model reaches F1 scores that are higher than the ones reached by SVM. In fact, when tested on unseen data, the F1 score is around 67%.

3.3 Classification with BERT

Finally, classification was experimented after fine-tuning BERT. The employed model is the one provided by `Simple Transformers`[8]. First, a Scikit-learn[9] object `MultiLabelBinarizer` was used for adapting the output labels to BERT. As usual, the dataset has then been split into two, and the training has lasted for 4 epochs. After the training, the model reached

[8] `https://simpletransformers.ai/docs/usage/`

[9] `https://scikit-learn.org/stable/`

a (micro) F1 score of 72%, which means BERT performs better than both SVM and LSTM.

4. Document embeddings

This section covers the content of the `DocEmbedding.ipynb` notebook. The idea behind this part of the project was to create a vector representation of the plots of the films (i.e. their doc embeddings), so as to detect those films which, given a starting one, are the most similar in terms of content. As such, this is an unsupervised task, for which no labelled data can be used - the quality of the representation will then be tested by direct usage of the system. A representation of this kind might work as the base of a possible recommendation system, which could present to users films that are akin to the ones they have enjoyed. Such a system could, for instance, be enriched by taking into consideration other information, like the director's name or the country in which the film has been produced.

For this second task, the dataset that has been used for classification has been merged with another dataset, available on Kaggle[1], so as to create richer representations[2]. Still, it must be stressed that in our representation *a given ID is supported by one and only one document*. In part, this is inherently

[1] `https://www.kaggle.com/hijest/genre-classification-dataset-imdb`. It is a single label dataset, but that is not a problem since the task under analysis does not involve dealing with target/labelled values. Note also that the employed dataset contained plots in different languages (e.g. in Italian for Italian films). We have partly taken care of this by removing those entries whose plots contained language-specific and frequent words - for instance, "und" was used for detecting German films and "il" was used for detecting Italian and French films.

[2] We first tried to use just the basic dataset, but the comparison with the augmented one showed that more data allow the model to define more reliable embeddings.

an important limit, that has to be taken into consideration when inspecting the capacity of the model.

The document embeddings have been created by mapping the plots of the films to the vector space. After some trials, we have decided to train the `Doc2Vec` model for 20 epochs, limiting the vector size to 200, and considering text windows of 8 words. The `min_count` argument, which is a threshold on words' frequencies, has been set to 5.

As already stated, the analysis of the quality of the vector representation can be conducted only empirically (*ground truth*). Thus, we will now report some examples of the model detecting some films as the most similar to others[3]. A first group of examples that deserve being mentioned are franchises: providing *The Lord of the Rings* as an input film causes the model to return, at the side of some unsatisfying results, films such as *The Lord of the Rings: The Fellowship of the Ring* and *The Hobbit*. Inspecting the words that are the most similar *in the documents that are the most similar* (i.e. in the other LOTR films) returns lemmas such as `frodo`, `hobbit` and `mage`, which are correctly distinctive of such films. The same discriminative capacity goes for the *Star Wars* films. Films that are very distinctive of a genre receive some which share both the genre and the tone as the most similar films. This is the case of *Alien*, which is "correctly" linked to sci-fi horror films (e.g. *Deep Rising*, *Creature*). Other examples of the vector representation being sufficiently good are cases such as *Hitman*: its plot

[3]Inevitably, the films returned by the model depend on the specific training of the model itself. The examples reported, though, are consistent even among different training processes.

contains words such as "crime" and "death", which leads the model to link it to other thriller films and/or to film where such concepts are relevant as well.

Finally, what follows are the visualization of how the document embeddings are distributed in the vector space after two different trainings and displaying partially different genres[4]. In both the figures, one can see that the different genres are quite distinctively distributed, i.e. some clusters are effectively composed. Moreover, note that in both the cases the position of the clusters itself is significant: note that, for instance, in 4.1 "Musical" is in a rather separate from "Western", whereas it is near to "Romance". Several are, indeed, the musical or music-related films with also a focus on romance storylines, whereas westerns and musicals are not the most canonical narrative mix.

[4]The links to the pictures in full resolution are the following: 4.1: `https://ibb.co/TWdwSLJ`; 4.2: `https://ibb.co/VHSPpdc`.

Figure 4.1: Document embeddings visualization

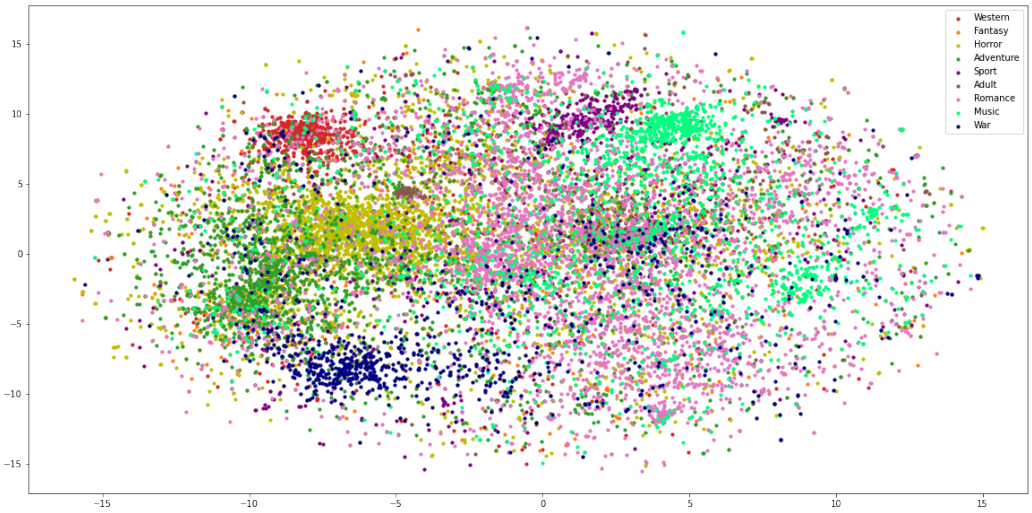
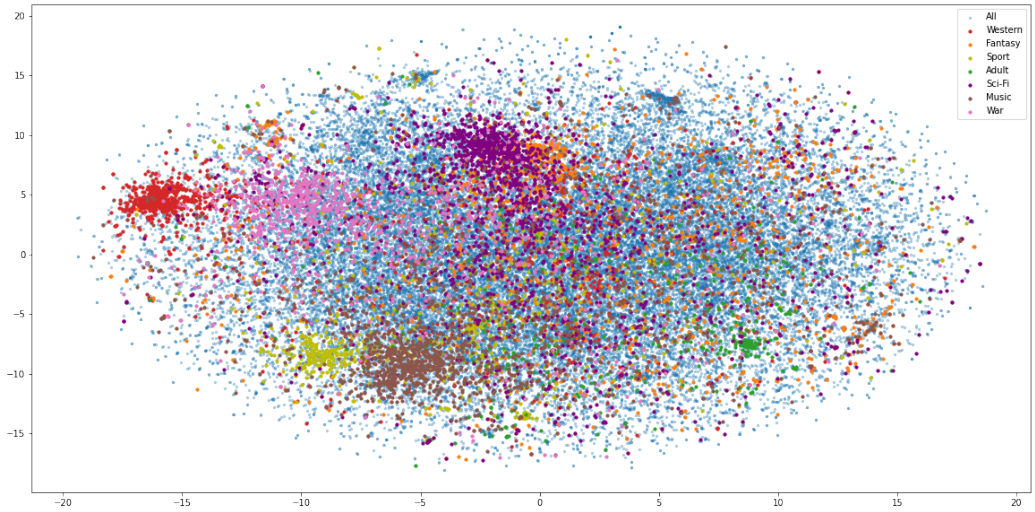


Figure 4.2: Document embeddings visualization



5. Conclusions

The conducted experiments have shown the connection between a film's genre and a description of its plot to be strong. In fact, even though the conditions were not favourable (the labels to be assigned were often many and the boundaries between them were not always evident, plus the data contained some noise - conditions which indicate that the task itself would not be trivial even for human annotators), the classification task has been carried satisfyingly by the various algorithms. As presented in 3, the different models have been employed in an order that matches their increasing complexity - such complexity, it has been shown, has actually led the classification capacities to increase, so that BERT reached higher scores than LSTM, which, in turned, performed better than SVM.

For all the models, in any case, the only textual content that has been employed as features are the lemmas and/or tokens of the plots themselves, which shows this semantic level to be the one truly relevant for such classifications. As discussed in 4, the semantic content of plots was also informative enough for creating basic yet already sufficiently robust vector representations of the plots themselves.