Genre-ous: The Movie Genre Detector

Amr Shahin and Adam Krzyżak(✉)

Department of Computer Science and Software Engineering, Concordia University, 1455 Boulevard de Maisonneuve West, Montréal, QC H3G 1M8, Canada
amrnablus@gmail.com, krzyzak@cs.concordia.ca

Abstract. The advent of Natural Language Processing (NLP) and deep learning allows us to achieve tasks that sounded impossible 10 years ago. One of those tasks is text genre classification such as movies, books, novels, and various other texts, which, more often than not, belong to one or more genres. The purpose of this research is to classify those texts into their genres while also calculating the weighted presence of this genre in the aforementioned texts. Movies in particular are classified into genres mostly for marketing purposes, and with no indication on which genre predominates. In this paper, we explore the possibility of using deep neural networks and NLP to classify movies using the contents of the movie script. We follow the philosophy that scenes make movies and we generate the final result based on the classification of each individual scene.

Keywords: NLP · HAN · Genre classification

1 Related Work

Brezeale and Cook [1] used combination of closed captions[1] and visual features to detect the genre of a video using a support vector machine (SVM). The authors used 81 movies from the MovieLens project[2] and were able to achieve 89.71% accuracy when using closed captions as the feature vector inputted to the SVM. It however is not clear, how the authors calculated the accuracy when the movie belongs to multiple genres. Moreover, the usage of classical machine learning algorithms like SVM and bag of words does not scale well when working with a larger dataset. The authors also pointed out that the closed captions "typically won't include references to non-dialog sounds", we found that using the original script of the movie circumvents this problem as not only it includes the actual speech, but also the "general feeling" of the scene.

Similar to our approach, Tsaptsinos [2] used a hierarchical attention networks to classify songs into genres based on their lyrics. The author tested the model with over 117-genre dataset and a reduced 20-genre dataset and concluded that the HAN outperforms both non-neural models and simpler neural models whilst

[1] https://en.wikipedia.org/wiki/Closed_captioning.
[2] https://grouplens.org/datasets/movielens/.

© Springer Nature Singapore Pte Ltd. 2020

also classifying over a higher number of genres than previous research. Moreover, the HAN model has the ability to visualize the attention layers of the HAN model. While on the surface, it may seem that this problem is identical to ours, however, a deeper look debunks this assumption. It is very easy to notice that the genres differ greatly between songs and movies, additionally, a song belongs to a single genre which makes dataset gathering simpler, and finally, a trained model can be used directly to predict unseen data unlike our case, where the model is used to classify scenes that are used to classify the full script.

Aside from the aforementioned papers to our best knowledge there was no other research that directly works with text genres using the movie script. In the rest of this paper we will be presenting research done on text classification in general.

2 Dataset

One of the biggest challenges we were faced with while writing this paper was finding a suitable dataset that contains proper training data. The form of data we were looking for is a sentence mapped to a single genre (i.e: [But you step aside for the good of the party; people won't forget. The President and I won't let them] belongs to the genre 'politics', [He's in love with you. I've only ever seen him look at one other girl the way he looks at you] is 'romance', etc ...). Unfortunately, such dataset does not exist.

2.1 Available Datasets and Their Issues

We considered using The Internet Movie Script Database which contains the full script of most movies and parsing its contents, however, this leaves us with a large text corpus that belongs to multiple genres, which presents two issues: 1. The resources required to run a Recurrent Neural Network (RNN) to handle a corpus of this size are enormous and 2. Training a NN on a text that belongs to multiple genres will lead to the network learning the joint probability of the genres which is not desired in our experiment. Another option was using a news dataset which contains texts mapped to a certain news category such as https://www.kaggle.com/crawford/20-newsgroups or https://www.kaggle.com/therohk/india-headlines-news-dataset, we found that there is no clear one-to-one mapping between a news class and a movie genre aside from politics and sports, which will cause our model not to scale well should we decide to add more genres to our work.

2.2 Building a Custom Dataset

Due to the above-mentioned reasons, we decided to build our own dataset consisting of five genres: action, comedy, drama, politics, and romance to collect data for. We experimented with two types of textual contents:

Movie Quotes. Our process starts with scraping Google search using the queries: "Top <genre> movies" and "Top <genre> series"[3]. Out of the search result, we manually selected a collection of movies and series that we felt represent the selected genre best for the list of movies and series. Using this set of movies and series, we built a tool that scrapes the API of https://en.wikiquote.org/ collecting quotes belonging to this set.

This method provided us with the data we needed properly formatted. As the number of samples per genre varied greatly, we additionally scraped https://www.goodreads.com/quotes in order to have matching numbers in our five genres. We managed to collect 3000 samples for each genre which we have used both for training and testing.

Movie Plots. Using the same set of movies and series in the aforementioned method, we scraped https://www.imdb.com/ collecting plot synopsis of the movies and series we selected, an example of a synopsis can be found at https://www.imdb.com/title/tt0068646/plotsummary. Unlike the previous method, we were able to assemble a balanced dataset, complementing was unnecessary.

2.3 Data Preparation

In order to convert words into a form the network can understand, we first tokenize the sentences into individual words, which in turn got converted into integers each representing a unique index corresponding to each word present in the corpus. The resulting tokenized sequences were padded to be all of the same lengths, the sequence length is a parameter that will be specified for each model in the corresponding section.

The embedding layer is the other part of the input, which consists of the word embeddings of the form <unique word index>:<word embedding vector>. This is used as a look-up table for the neural network in order to map integers to vectors. We chose GloVe embedding of size 100 for in our experiments.

3 Methods

3.1 Attention

Self Attention (AKA Bahdanau or Intra Attention). Self-attention proposed by Bahdanau et al. [3] works by assigning an alignment score between the input as position i and the output based on how much the input affects the output. Self attention works by training a feed-forward networks with a single hidden layer along side the main network, thus, the loss function for attention neural network will be:

[3] Thus, looking up the genre "Romance" will result in queries: "Top romance movies" and "Top romance series".

$$score(s_t, h_i) = v_a^T \tanh(Wa[s_t; h_i])$$

where W_a is the weight of the attention layer.

Luong Attention. Luong et al. [4] proposed the idea of global and local attention, the global attention is similar the aforementioned Bahdanau attention where the attention vector moves freely over the input, while the hard attention is a mix of soft attention and hard attention where only part of the inputs can have the attention at a given time step, the model is preferred over hard attention as it's differentiable.

3.2 Word Representation

Machine Learning and Deep Learning architectures are incapable of processing text directly as input. In the case where the input is a text, pre-processing is needed in order to convert the text into numbers. In this section, we present the various methods to do so.

One-Hot Encoding. A one-hot encoding, in general, is a representation of categorical variables as vectors. Each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1, for example, applying One-Hot to an input vector consisting of: ['drama', 'comedy', 'sports'] will result in the following encoding [[1 0 0], [0 1 0], [0 0 1]].

While One-Hot encoding does work in the sense that they convert words to numbers, however, they fail to capture the relations between various words, the word "Ottawa" could be represented using to the 1000th column, while the word "Amman" could be the 1st column, although both words represent capitals and should have smaller distance. For this reason, One-Hot encoding is not widely used in NLP.

3.3 Word Embedding

A Word Embedding format generally maps a word to a vector. One important property of the vector representation of a single word is that its distance from other similar words[4] is less than that of less similar words. The resulting vector has the property that cosine similarity between words is higher for more similar words. This property is very important when training a neural network as it helps the network determine the nature of words it did not see during training.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \|\boldsymbol{b}\| \cos\theta$$

$$\cos\theta = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|}$$

[4] Similar here means the words appear in the same context.

Word2Vec. Mikolov et al. [5] used skip-gram model to generate embeddings and circumvented some of the training challenges using negative sampling [6]. The main idea behind this method is that you train a model on the context of each word, so similar words will have similar numerical representations.

Word2vec model learns the weights by feeding a pair of input word and a target word to a neural network with one hidden layer of size [embedding dimension, vocabulary size] and an output layer of dimension [vocab size] consisting of softmax units. The hidden layer represents the probability that the word it represents will appear in the same context as the target word.

When the network training is done the output layer is dropped and the hidden layer will be used as the word vector.

GloVe: Global Vectors for Word Representation. Pennington et al. [7] presented the idea of learning embeddings by constructing a co-occurrence matrix (words X context) that counts how frequently a word appears in a context[5].

3.4 Hierarchical Attention Networks (HANs)

HANs consist of stacked recurrent neural networks on word level followed by an attention model to extract important to the classification of the sentence and aggregate the representation of those informative words to form a sentence vector. Then the same procedure is applied to the derived sentence vectors which then generate a vector that carries the meaning of the given document and that vector can be passed further for text classification as shown in Fig. 1

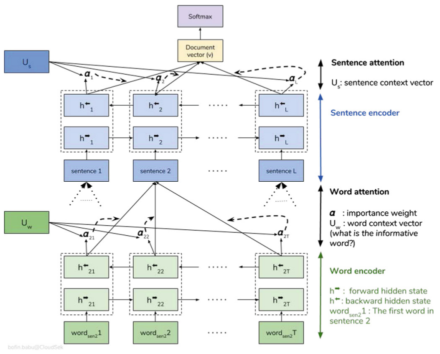

Fig. 1. HAN structure (image source: Fig. 1 Yang et al. [8])

[5] Context here is user-defined, in the literature it is usually chosen to be whether or not a word appears within X number of words of the target word.

4 Experiments

4.1 Model Architecture and Configuration

The HAN model is easier to be explained when thought of as two separate models, the first part (the encoder) consists of conventional recurrent neural network (RNN) built with a bi-directional long short-term memory (BLSTM) layer followed by an attention layer of the same size, this model is responsible for encoding the input data, it takes the word embeddings as input, and outputs an encoded representation of the words based on the hidden states of the BLSTM cells. The output of the encoder is then fed to the second part of the model, which in turn starts off with a time-distributed layer, this layer is the core of the HAN network, the purpose of the time-distributed layer is to run a copy of the encoder on each input sentence, the time-distributed layer is followed by a BLSTM layer of and an attention layer of the same size. Finally, the model adds a softmax fully connected layer as the output layer.

5 Discussion

5.1 Results

The HAN model achieved a test accuracy of 90% after training for 100 epochs on the quotes dataset and a testing accuracy of 93% on the plots dataset. The model out-performs, in terms of accuracy, both the traditional bi-directional LSTM with an accuracy of 50% and the one-dimensional CNN model with an accuracy of 75% as summarized in Table 1. It is worth mentioning, however, that the model is 2x times slower to train than the CNN model and about 1.5x times slower than the bi-directional LSTM. Another interesting feat to this model is that splitting the quotes in fewer sentences with higher maximum sentence length leads to better results compared to using more sentences with shorter sentence length. Hyper-parameter tuning was done on the validation dataset, we found that using Adam optimizer [9], a dropout [10] rate of 0.5, and an LSTM l2 regularizer with rate $1e^{-5}$ yield the best results.

Table 1. Models' testing accuracies

Model	Dataset	Accuracy
HAN	Quotes	90%
CNN	Quotes	75%
BLSTM	Quotes	51%
HAN	Plots	93%
CNN	Plots	86%
BLSTM	Plots	N/A

Looking at the confusion matrix exhibited in Table 1, we can see that the confusion is distributed evenly across genres. The only exception being comedy and romance, which is understandable since the romance movies tend to have the comedy genre as well, this example taken from the movie "The little mermaid" shows a potential source of confusion: "Scuttle: This, I haven't seen this in years. This is wonderful! A banded, bulbous snarfblatt.". The quote is more likely to be classified to the comedy genre by a human than it is likely to be classified as romance although it belongs to the romance genre, keeping in mind that the data was not manually labelled and that the HAN model can mitigate this issue as we explain the following section, we decided that this sort of data is an acceptable noise and does not affect the final results of this research.

Table 2. Confusion matrix of the HAN model on quotes

Genre	Action	Comedy	Drama	Politics	Romance
Action	370	7	6	12	3
Comedy	5	313	8	15	**38**
Drama	6	10	399	14	5
Politics	4	7	9	352	15
Romance	3	15	6	9	369

5.2 Rationale

To understand why HAN performs better than the other models, let us take a look at the following examples taken from our dataset:

Example 1: A text from Dr. House

Dr. House: [to Dr. Cameron] Is he Canadian?
Dr. Cameron: He's a low priority.
Dr. House: Is that a yes?

Example 2: A text from Godfather

Don Vito Corleone: [Sobs for a moment before he regains his composure] I want no inquiries made. I want no acts of vengeance. I want you to arrange a meeting with the Heads of the Five Families. This war stops now.

Looking closely at the above examples, it is clear that, not only, the second sentence contributes much more to the drama genre than the first one, but also, there are particular keywords like "sobs", "vengeance" and "war" in the second example that give clearer clues to the drama genre. The Hierarchical attention networks excel in such cases as they utilize two attention vector, the first works as a conventional attention vector over single words as described in Sect. 3.1 and the other works as an attention vector for the whole sentence.

5.3 Plots VS. Quotes

At the first glance, it looks that using the trained model to do the final classification of the full movie script makes more sense as all the three models performed better when using this dataset. However, when we tested both models, it turns out that the model trained on quotes was much more successful classifying movies correctly, the reason being is that the script text is more similar in structure to the quotes than it is for plots. We discuss the classification of full movie scripts in the following sections.

5.4 Attention Visualization

Another important feature of the attention networks in general is that the values of the attention can be visualized, we selected two examples from our dataset to show the attention for, the example, taken from the movie "The Rosa Parks Story" of the drama genre, is a very good example of attention as we can see that words that carry dramatic features such as "civil rights", "leaders" and "unmarried" got high attention values as shown in Fig. 2[6].

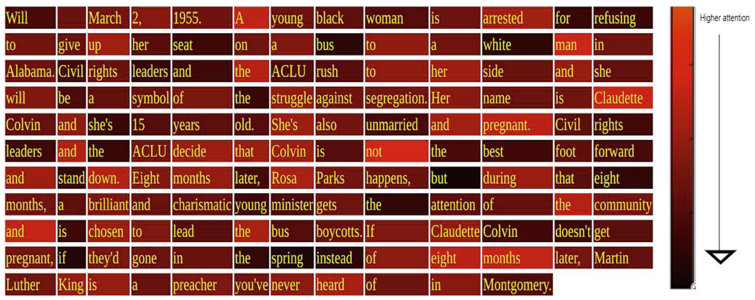

Fig. 2. Attention values for a drama genre example

Using the Trained Model for Genre Prediction. While building this model, we theorized that if the model can classify a single scene correctly, it would be able to be able to detect the overall genre, the testing process goes as follows: the movie script is downloaded from "The internet movie script database", the script gets parsed and broken into individual scenes. Next, the words are converted into their corresponding indexes, it's important to preserve the indexes to match the previously used indexes so that the embedding layer maps the correct vector to the word. The scenes are then padded to match the training data, and finally, the scenes are classified individually and the weighed sum of the probabilities is considered to be the full corpus' genre probability.

[6] It is also worth mentioning that although the attention is visualized on words, in reality the attention is applied to the RNN states which carry the meaning of the current word along with the surrounding words.

5.5 The Problem of Unseen Words

One challenging issue we faced is that a lot of words that appear in the testing data do not actually appear while training, which causes them to be dropped and could dramatically affect the predictions, to circumvent this, the model needs to be altered as follows[7]:

1. Remove Kera's embedding layer.
2. Add a Kera's input layer of type float, the shape of the model should be (max sentence length x embedding dimension).
3. The input to the trained model should be changed to be word embeddings rather than word ids.

6 Testing

Testing the model is particularly challenging due to non-existent data to evaluate with, we decided to rely in our experience with well-know movies as well as comparing the results with IMDB's genre classification, following are some of the predictions we made using the trained model:

1. "**Godfather**" (full script: http://www.dailyscript.com/scripts/The_Godfather.html), the two most dominating genres were: action with 24% of the total scenes and drama with 32% of the total scenes, we feel it matches the movie genre and it matches IMDB's genre tags.
2. "**Dumb and Dumber**" (full script: https://www.imsdb.com/scripts/Dumb-and-Dumber.html) had 63% of the scenes belonging to the comedy genre and about 10% for all other genres, which is pretty accurate depiction of the movie, it matches IMDB.
3. movie "**Lincoln**" (full script: https://www.imsdb.com/scripts/Lincoln.html) had 64% scenes classified as politics and 17% as action, we feel this is an accurate classification, we were unable to compare to IMDB as they don't have the politics genre.

7 Future Work

First and foremost, we would like to collect a more empirical testing data to evaluate our model, against, while we did put a lot of effort into manual testing, having an empirically labeled dataset would be very helpful especially in highly-subjective matter such as genres.

We also would like to predict the movie genre using an LSTM network, where the spacial input is the scene genre classification, the output would be similar to our existing input.

And finally, we predict that integrating the model's predictions into a movie recommendation engine is expected to lead to higher accuracy, which an experiment we are working on.

[7] For more details: https://amrnablus.github.io/blog/word_embedding.html.

8 Conclusions

In this research, we explored the possibility of using hierarchical attention network (HAN) architectures targeting of building a model capable of predicting a genre of a text corpus. Our dataset consists of quotes from famous movies from each genre taken mainly from wikiquote.com website and complemented from goodreads.com.

Upon looking at the confusion matrix in Table 2, we see that "romance" and "comedy" have high confusion, leading us to conjecture that generally the romance movies have comedy as a sub-genre, keeping in mind that the dataset we used consists of quotes picked up from movies without any sort of manual correction. In general, the comedy genre is difficult to detect by machines as it can be very context-dependent and thus often being a source of confusion.

Acknowledgement. We would like to thank Prof. Charalambos Poullis of the Department of Computer Science and Software Engineering, Concordia University for suggesting movie plots as an input to our model which gave us a good basis for comparing the results of two independent models.

References

1. Brezeale, D., Cook, D.J.: Using closed captions and visual features to classify movies by genre. In: Poster Session of the Seventh International Workshop on Multi-media Data Mining (MDM/KDD2006) (2006)
2. Tsaptsinos, A.: Lyrics-based music genre classification using a hierarchical attention network. In: Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, 23–27 October 2017, pp. 694–701 (2017)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXive-prints, abs/1409.0473, September 2014
4. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421. Association for Computational Linguistics (2015)
5. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. Computing Research Repository, abs/1301.3781 (2013)
6. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 26, pp. 3111–3119. Curran Associates Inc. (2013)
7. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543. Association for Computational Linguistics (2014)

8. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489. Association for Computational Linguistics (2016)
9. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)