# A FILM SYNOPSIS GENRE CLASSIFIER BASED ON MAJORITY VOTE

Roger Booto Tokime[1] and Eric Hervet[1]

[1]Perception, Robotics, and Intelligent Machines Research Group (PRIME), Computer Science, Universite´ de Moncton, Moncton, NB, Canada

## *ABSTRACT*

*We propose an automatic classification system of movie genres based on different features from their textual synopsis. Our system is first trained on thousands of movie synopsis from online open databases, by learning relationships between textual signatures and movie genres. Then it is tested on other movie synopsis, and its results are compared to the true genres obtained from the Wikipedia and the Open Movie Database (OMDB) databases. The results show that our algorithm achieves a classification accuracy exceeding 75%.*

## *KEYWORDS*

*Movie Synopsis, Text Classifier, Information Retrieval, Text Mining.*

## 1. INTRODUCTION

Automatic classification of text is known to be a difficult task for a computer. Usually, algorithms in the field try to regroup a given set of texts into genres or categories that can contain subcategories (e.g. News articles can be separated in sports, news, weather, etc.), or even regroup them together to form a new one. This can be accomplished by the use of several methods and algorithms that are usually based on text analysis, word grouping, sentence segmentation, etc. Automatic text classification is widely tackled in the field of classifying textual data from the web [6] [9] [8], and some research has been done on classifying other textual data such as books summaries [2]. Though there are many algorithms and strategies already in place, with some using advanced approaches like machine learning [3], it is still difficult for algorithms to classify correctly texts into genres. The main reason for this has been explained by Chandler in his *Introduction to Genre Theory*: ”There are no rigid rules of inclusion or exclusion. Genres are not discrete systems consisting of a fixed number of listable items”. While working on a computer vision homework, we had to implement an algorithm for the feature extraction task. We used the Hough transform to do so, and we got to study the purpose of this technique. Wikipedia gives a very good description of its purpose: ”The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space, that is explicitly constructed by the algorithm for computing the Hough transform.” [12]. What is important to keep in mind is that in computer vision, we are able to find features in an image by letting multiples algorithms vote to what they consider a good feature, and at the end keeping only the most rated. We then thought to apply this principle to classify text into genres, which is the main matter of this work. This paper shows how to make use of a combination of features in order to build a majority voting system that finds automatically the category(ies) which a text belongs to. Our work focuses on the classification of movie synopsis into genres. First, we are going to introduce a new concept, the movie signature. A movie signature is defined as the list of categories found by the system after voting. Now, let's have an overview of the struc- ture of this paper, which is organized as follows: First, we describe how the data collection has been retrieved and processed before being presented to the system. Second, we present the different algorithms compared to compute the similarity between movie synopsis. Then we present the results that show how the majority voting system reaches a precision exceed- ing 75%. Finally, we conclude with a discussion about possible improvements and other application do- mains where our system could be used to classify other types of documents than movie synopsis

## 2. Related Work

This paper is about automatic classification of movie synopsis into genres. To do so, we use a novel approach that combines the strengths of several similarity algorithms (that could be referred as voters) from the domain of Natural Language Processing (NLP) in order to obtain a better classification. Each of these algorithms taken individually performs great on some specific data, but is not able to process efficiently other various kind of data. The goal of our method is to generalize the classification of textual data into pre-labeled categories. To do so, we explore different NLP methods that classify textual data based on specific features. The methods chosen as eligible to vote are the ones that perform the best in their category. We have the following feature comparison themes: term frequencies, sets, vector-based and strings. Those features are the most used in the field of information retrieval and text classification.

### 2.1 Term frequencies comparison

In order to know if a textual query is a member of a category, the first thing we can think of is to count the number of words from the textual query and then sum it up. This sum can be used as a measure of the similarity between the textual query and the pre-labeled database.

#### 2.1.1 Term frequency

Term frequency is at first glance a good idea because we would expect that the words representing a target category appear more often than the others, but it is not that simple. Term frequency has a big defect due to its definition: Indeed, by counting only the number of words from the textual query to a category, e.g., Action, the frequent words in the English language which bring no relevant information such as "the", "a", "about", etc. are the words that come up most often. To avoid this problem we could consider only the words repeating with a certain frequency, but then it rises the question of the right threshold to use. To bypass this issue, we use a list of stop-words from the English language [7]. A stop-word is defined as a word that comes up most often in various texts of a given language and provides no relevant information to categorize the text. Now that we are able to filer stop-words, here is the mathematical expression that we use to calculate the similarity between two texts, based on the term frequency. Let us assume that *w* is a word from the query that occurs in a target category, cat (reset at each run of an algorithm) is the list of categories, and *x* is a given target category, e.g., Action. We have that:

$$\mathbf{cat}(x) \mathrel{+}= \begin{cases} 1, \forall w \notin \{\text{stop-words}\} \\ 0, \forall w \in \{\text{stop-words}\} \end{cases} \tag{1}$$

This means that we use a binary classification for each word, according it is a stop-word or not. Since in our case, a text query is a movie synopsis, *cat* represents a similarity measure between a movie plot and all the different known genres in which we would like to know to what extent a given synopsis applies to. This method is simple to implement and give good results. The main concern about this method is that the quality of the results is really dependent on the stop-word list, so in practice a prior knowledge and pre-processing of the genres, especially about what a stop-word is in the field of movie synopsis, is necessary to achieve accurate classification results.

### 2.1.2 Term frequency inverse document frequency

Text classification by simply counting the frequencies of words, minus the stop-words, is then very limited when previous knowledge of the classifying genres is not available. Since we are trying to build an accurate voting system, we use the same basis as this type of algorithm, but we introduce a much more robust term frequency (TF) similarity algorithm. Here, the approach is different in a way that we consider the words of each kind of our genre list as independent entities, and their influence during the classification is represented by some weight *wt*. This way of doing things is called feature selection. Feature selection studies how to select a subset or list of attributes or variables that are used to construct models describing data. Its purposes include dimensions reduction, removing irrelevant and redundant features, reducing the amount of data needed for learning, improving algorithm's predictive accuracy, and increasing the constructed model's comprehensibility [14]. Our goal here is to reduce the vector dimension by automatically removing irrelevant words. An irrelevant word is defined as a word that appears in at least 80% of the categories. To do so, we use the inverse document frequency (IDF) measure. The IDF measure is based on the document frequency (DF), which is simply the number of documents in which a word *w* appears. The IDF can be computed as follows:

$$IDF(w) = log\left(\frac{D}{1 + DF(w)}\right) \tag{2}$$

where *D* is the total number of document (genres), and the plus one at denominator is used in case a word appears in none of the datasets, to avoid dividing by zero [5]. With this measurement, we can simply remove irrelevant words that appear in at least 80% of the categories by checking if the result of the ratio *(DF/D)* is less than 0.8. The calculation of the term frequency inverse document frequency (TFIDF) is performed as follows:

$$TFIDF(w) = TF \times IDF(w) \tag{3}$$

TFIDF is a measure of the importance of a word *w* in a document or a set of documents. Let us assume that *w* is a word from the query that occurs in a category, cat is the list of categories, and *x* is a target category, e.g. Action. We have that

$$\mathbf{cat}(x) \mathrel{+}= \begin{cases} TFIDF(w) \text{ if } (DF/D) \leq 0.8 \\ 0 \text{ if } (DF/D) > 0.8 \end{cases} \tag{4}$$

We have now a robust term frequency similarity measure that eliminates non relevant words to our purpose from the entire vocabulary (over all documents).

## 2.2 Sets comparison

In this section, we present a new approach which does not compare the frequencies of words when calculating the similarity, but simply checks the presence or the absence of a word. To do so, we use the Jaccard coefficient.

#### 2.2.1Jaccard similarity

The Jaccard coefficient, also known as the Tanimoto coefficient, is a measure of similarity based on datasets. In the article ”Text Categorization using Jaccard Coefficient for Text Messages” [4], it is shown that this kind of similarity is extremely simple and fast to compute, since it comes down to the simple calculation of the division of the intersection by the union of two sets of data The values of the Jaccard coefficient range from [0, 1], with 0 meaning that the two datasets are not similar, and 1 that they are exactly the same. Let us assume that *pq* is the plot query, cat is the list of categories and *x* is any target category, e.g., Action. We have that

$$J(x) = \frac{pq \cup \mathbf{cat}(x)}{pq \cap \mathbf{cat}(x)} \tag{5}$$

J (x) measures the similarity between the set of categories and the set of words of the plot query. Since our case, the cat sets are much larger than the *pq* set, the Jaccard coefficient can never reach the value 1. But it still provides a good order of similarity between the different genres. Therefore, the Jaccard similarity measure is one of the method we use in our combined voting system.

## 2.3 Comparison of the angle between vectors

At this stage, we change the way we look at our documents, and rather represent them as vectors of words, also called bags of words. A bag of words representation allows us to calculate the correlation between two bags, which is used as the similarity measure between the corresponding documents. Calculating the correlation between bags or vectors of words is the same as calculating the angle between two vectors, and is known as cosine similarity.

#### 2.3.1 Cosine similarity

The cosine similarity measures the cosine of the angle between two vectors of words. Each vector is an n-dimensional vector over the words $W = \{w_1, w_2, \ldots, w_n\}$, and each dimension represents a word with its weight (usually TF) in the document. Also, the word vector similarity coefficient does not vary if the vectors do not have the same lengths. This feature is especially convenient for us since our datasets vocabulary is usually much bigger than the synopsis query. Because the weight of each word is its TF, the angle of similarity between two vectors is non-negative and bounded in [0, 1] [1]. Let us assume that *v* is the plot query vector, cat is the list of categories and cat(*x*) is any specific category vector, e.g., Action. We have that:

$$Sim_{cos} = \frac{\vec{v} \cdot \mathbf{cat}\vec{(x)}}{|\vec{v}| \times |\mathbf{cat}\vec{(x)}|} \tag{6}$$

This means that we consider the information as a vector of words rather than a set of words, and the angle between the two vectors is used as a similarity metric.

## 2.4 String comparison

Here, we get deeper into our comparison scheme by going as low as comparing the strings from a given plot query to the ones in the dataset. This level is the lowest we can go in order to compare two sets of textual data. The idea here is to have an algorithm that performs well under the following conditions: strings (words) with a small distance must be similar, and the order in which the words are distributed in the text must not affect the similarity coefficient. The reason behind these conditions is to give the algorithm the possibility to be used at large scale.

### 2.4.1 Dice similarity

Well-known algorithms, such as Edit Distance, Longest Common Substring and others, do not perform well against these requirements. The Edit distance does respect the first requirement, which states “words with a small distance must be similar”, as it rates “FRANCE” and “QUEBEC” as being similar with a distance of 6, compared to “FRANCE” and “REPUBLIC OF FRANCE” with a distance of 12. The Longest Common Substring would give “FRANCE” and “REPUBLIC OF FRANCE” quite a good rating of similarity with a distance of 6, but would fail at giving the same similarity value for “FRENCH REPUBLIC”, “REPUBLIC OF FRANCE” and “REPUBLIC OF CUBA” [10]. For these reasons, we decided to use the Dice coefficient as it does respect both conditions. The Dice coefficient is defined as twice the number of common terms in the compared strings, divided by the total number of terms in both strings [11]. Let us assume that pairs($x$) is function that generates the pairs of adjacent letters in the plot query, cat($x$) is the same function for the list of categories, and $s_1$and $s_2$ are respectively a plot query and a specific dataset genre

text, e.g., Action. We have that

$$\text{similarity}(s_1, s_2) = \frac{2 \times |\text{pairs}(s_1) \cap \text{cat}(s_2)|}{|\text{pairs}(s_1)| + |\text{cat}(s_2)|} \tag{7}$$

Here, each plot and categories are decomposed in pairs of words, and the calculation works as follows: it computes the percentage of pairs of words that can be found both in the plot $s_1$and in the category $s_2$, then multiplies it by a factor of pairs that belongs to both $s_1$ and $s_2$.

## 3. DATASETS

The data used in this work are movie synopsis from Wikipedia [13]. The training data is a selection of American, British, and French cinematographic works from 1920 to 2016. The collection of data is made once and requires four steps: The first step consists in retrieving the information from the Wikipedia website; The second step consists in filtering only the cinematographic movies, of which at least one of the genres belongs to the following list of 19 genres: Action, Adventure, Animation, Comedy, Crime, Drama, Fantasy, Family, Fiction, Historical, Horror, Mystery, Noir, Romance, Science, Short, Thriller, War, and Western. The third step consists in repeating steps 1 and 2 with the data from the OMDB database, with the difference that this time the requests are limited to the movie titles previously filtered. At the end of the third step, the number of movies goes from 25561 to 9937, and after eliminating duplicates there is a total of 8492 movie titles remaining. Those are the ones used to build the training dataset used in the fourth step. The fourth step consists in the creation of 19-word banks corresponding to the 19 aforementioned genres. The word banks are created as follows: Let $T$ be a movie object, and cat a word bank. We have that:

$$\forall \textbf{cat} \in T_{\text{genre}} : \textbf{cat} \mathrel{+}= T_{\text{title}} + T_{\text{plot}} \tag{8}$$

The means we add the movie titles to the word bank to help the algorithms that calculate the similarity between two documents. The same steps are performed when creating the test data. The test data contains Canadian, Spanish, Australian, Japanese, and Chinese cinematographic works from 2000 to 2015. A total of 793 different titles are used as test data. We would like to precise that for this experiment, all textual information were taken from the english version of Wikipedia, therefore the only language conflict we may have could be the translations of the synopsis into english.

## 4. Similarity Measures

To allow various algorithms to vote, a measure of similarity must be defined, in order to compare two objects and measure their level of disparity or similarity. Usually, this measure depends on characteristic features upon which we want to make a comparison. In many cases, these characteristics are dependent on the data, or on the problem context at hand, and there is no measure that is universally best for all kinds of clustering problems [1]. This is the reason we made the hypothesis that a combination of different features would perform a better classification of textual data. This approach is similar to the ones used in supervised machine learning since it is based on previously created pre-labeled databases. The percentage of belonging of a text to each category is computed by different similarity calculation algorithms. However, unlike supervised machine learning algorithms, the classification of the test data is performed by choosing the most significant categories obtained by each of the similarity algorithms. More precisely, the algorithms used in this work compare similar features between two text documents based on: term frequencies, sets, bi-grams, and angles between vectors of words. The following sections of this paper give some details about the algorithms used and compared in this work.

## 5. Results

Our method is a majority voting system, which means that each algorithm has the chance to vote for its top candidates. The final election is conducted as follows: First, each algorithm receives the same synopsis query, and classifies the query over all genres. Second, the top candidates from each algorithm are selected to create the movie signature. Third, the movie signature is compared

to the true genres, and the average accuracy is computed in the following way

$$\text{accuracy} = \frac{len_{Sum}}{Sum_{GT}} \tag{9}$$

$$\text{with: } len_{Sum} = \sum_{i=1}^{N} len(|MS_i \cap GT_i|) \tag{10}$$

$$\text{and: } Sum_{GT} = \sum_{i=1}^{N} len(|GT_i|) \tag{11}$$

where *MS* represents the movie signature, *GT* the ground truth, *N* the number of test synopsis, $len_{Sum}$ the true positive predictions for a given synopsis, and $Sum_{ttT}$ the total amount of genres expected across the test set. We conducted three elections where we selected the top 1, 2 and 3 candidates from each voter. The reason we limited our vote to the top 3 best candidates is purely mathematical: Indeed, we have 5 voters and 19 genres, and: $^{19}$ = 3. This means that in the worst

case, if each voter elects different candidates, we get a movie signature length that is less than the total number of genres, which means that there is a long lis$\overline{5}$t of elected candidates. Therefore, it reduces the precision, and enhances the probability to find a genre in which a film synopsis belongs to. Table 1 shows the results ordered from top 1 to 3, while table 2 shows the effect of the majority voting system on the precision of the algorithm. This result is completely expected since the measure of precision in the context of classification is computed as follows:

$$\text{precision} = \frac{|\text{truePositive}|}{|\text{truePositive} + \text{falsePositive}|} \tag{12}$$

It is the ratio between the ground truth (truePositive), and the sum of our signature and the ground truth. Our signature (predicted genres) length is significantly bigger than the ground truth when considering the top choices, as shown in table 3, and because a false positive result is the set of propositions that are in our signature, but not in the ground truth, we end up with a low precision. This is a trade-off we had to make because finding the genre(s) which a synopsis truly belongs to is more important in our case. So here is how we compute the value of falsePositive based on *MS* (Movie Signature) and *GT* (Ground Truth):

$$\text{falsePositive} = |MS| - |MS \cap GT| \tag{13}$$

Table 1 presents the best accuracies obtained when considering the top 1, 2, and 3 movie genres.

Table 2 compares our algorithm, Majority Vote, to the five other existing algorithms described earlier, in terms of accuracy, precision, recall, and F-measure, also called F1-score. The F-measure is defined as the harmonic mean of precision and recall :

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

Finally, Table 3 presents the average signature lengths for top 1, 2, and 3 movie genres based on our algorithm.

**Table 1. Top 1 to 3 best classification accuracies**

| Top | Accuracy | lenSum | sumGT |
|---|---|---|---|
| 1 | 35.41% | 624 | 1766 |
| 2 | 60.27% | 1062 | 1766 |
| 3 | 77.53% | 1366 | 1766 |

**Table 2.Performance of our proposed algorithm vs. the existing ones**

| Algorithm | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| Majority vote | **77.53%** | 16.76% | **79.62%** | 27.69% |
| Cosine | 13.56% | 9.92% | 10.79% | 10.34% |
| TF-IDF | 30.93% | 22.82% | 30.78% | 26.21% |
| Term Frequency | 52.95% | **39.18%** | 59.09% | **47.10%** |
| Jaccard | 5.22% | 3.74% | 4.51% | 4.10% |
| Dice | 3.35% | 2.35% | 2.85% | 2.58% |

**Table 3 . Increase of the signature length with our method**

| Top | Avg. Signature | Avg. Ground Truth |
|---|---|---|
| 1 | 3.77 | 2.23 |
| 2 | 6.87 | 2.23 |
| 3 | 10.26 | 2.23 |

## 6. CONCLUSION

In summary, we tackled a challenging problem in the field of text classification into multiple genres. More specifically, we developed a new method that automatically assigns genres to a movie synopsis. As explained in the introduction, the difficulty of this task comes from the fact that there is no rigid boundaries between the genres. To solve this issue, we proposed a majority voting system inspired by the Hough transform which is usually used in computer vision. Our method is able to generalize text classification into a category with an accuracy exceeding 75%. To obtain those results, we had to make some trade-off to the detriment of the precision of prediction. Due to the absence of rigid rules to discriminate between genres, we put the focus on the accuracy results, and outperformed the existing algorithms in the field. In addition, our system is language-invariant up to the extent of the stop-word list used by the term frequency voting method. We tested our method on movie synopsis with good results, and we believe that it could be used to classify books, journals, and even websites.

The next step after this work is to use deep learning, especially long short-term memory (LSTM) neural networks, to compare and classify movie synopsis.

## REFERENCES

[1] ANNA, H. Similarity-measures for text document clustering.

[2] EMILY, J. Automated genre classification in literature, 2014.

[3] IKONOMAKIS, E., KOTSIANTIS, S., AND TAMPAKAS, V. Text classification using machine learning techniques. WSEAS transactions on computers 4 (08 2005), 966–974.

[4] JADHAO, D. A. J. A. A. Text categorization using jaccard coefficient for text messages. International Journal of Science and Research (IJSR) 5, 5 (May 2016), 2047–2050.

[5] KA-WING, H. Movies' genres classification by synopsis.

[6] PHILIPP, P. Assessing approaches to genre classification, 2009.

[7] RANKS, N. L. Stopwords, visited 2018.

[8] S., Y., G., W., Y., Q., AND W., Z. Research and implement of classification algorithm on web text mining. In Third International Conference on Semantics, Knowledge and Grid (SKG 2007) (Oct 2007), pp. 446–449.

[9] SANTINI, M. Automatic genre identification: Towards a flexible classification scheme.

[10] SIMON, W. How to strike a match, 1992.

[11] WAEL, H. G., AND ALY, A. F. Article: A survey of text similarity approaches. International Journal of Computer Applications 68, 13 (April 2013), 13–18.

[12] WKIPEDIA. Hough transform, visited 2018.

[13] WKIPEDIA. https://www.wikipedia.org/, visited 2018.

[14] YANG, J. W. C. Text categorization based on a similarity approach.