Multi-genre Classification on Literary Books

ARISTOTLE UNIVERSITY OF THESSALONIKI

George, Dialektakis
gdialekt@csd.auth.gr

George, Georgiou
georgiougk@csd.auth.gr

Zisis, Flokas
flokaszisis@csd.auth.gr

June 2021

Contents

1 Overview **3**
- 1.1 Data . . . 3
- 1.2 Problems Addressed . . . 3
- 1.3 Evaluation . . . 4

2 Dataset **5**
- 2.1 Requirements . . . 5
- 2.2 Overview . . . 5
- 2.3 Pre processing . . . 5

3 Multi-label Classification **8**
- 3.1 Approach . . . 8
  - 3.1.1 Problem transformation methods . . . 8
  - 3.1.2 Neural network methods . . . 9
  - 3.1.3 Base estimators . . . 10
  - 3.1.4 Text representations . . . 10
  - 3.1.5 Evaluation . . . 10
- 3.2 Experiments . . . 10

4 Class Imbalance **12**
- 4.1 Multiclass Transformation & Approach . . . 12
- 4.2 Techniques . . . 13
  - 4.2.1 SMOTE . . . 13
  - 4.2.2 ADASYN . . . 13
  - 4.2.3 Text augmentation . . . 14
  - 4.2.4 Easy Ensemble . . . 15
- 4.3 Evaluation . . . 15
  - 4.3.1 Metrics . . . 16
  - 4.3.2 Results . . . 17

5 Active Learning **19**
5.1 Query By Committee 20
5.2 Information Density 20
5.3 Ranked batch-mode Sampling 20
5.4 Experiments 21

6 Conclusion & Future Work **24**

Chapter 1

Overview

In the work presented in this report, we will attempt to perform genre prediction of literary books. The problem addressed, is described as predicting the relevant genre a book should belong to, given a short description of its contents.

Genre/label prediction is a classical classification task that is a prime target to be solved with machine learning models. It presents various challenges that must be addressed to efficiently solve and can be approached from various standpoints. It is inherently multilabel, as rarely a single genre fully describes literary work. The labels acquired can be used in various ways, i.e search and recommendation systems.

The task thus presents us with interesting problems and ample space for experimentation on different machine learning techniques and their results.

1.1 Data

For the task at hand we have utilized a dataset retrieved from Kaggle that consists of a number of science fictions books and a number of features, most notably their descriptions and the most voted reader labels. The textual nature of the dataset, alongside the high cardinality of the labels presents interesting challenges to solve. We approached the data in 2 ways. First in its initial multilabel form. Second in a simplified multiclass form that was created by selecting the most popular genre of each sample, after some simple pre-processing.

1.2 Problems Addressed

The selected dataset and task presented us with 3 distinct problems which needed to be addressed. Starting was the core **Multilabel Classification** problem, of predicting the set of correct labels for each sample, based on its description. Following, the dataset after performing the transformation of the target from multilabel to multiclass, presented **Class Imbalances** as is common with genres, especially user provided ones. This problem proved

challenging to solve, as typical oversampling methods that are based on datapoint distances do not translate well to textual features. Finally, we experimented on the performance of various **Active Learning** techniques. Text is one of the most prominent targets for the use of active learning methods, as it is costly, tiresome and inconsistent to label large quantities of text by hand.

1.3 Evaluation

For the evaluation of the techniques employed, we decided on utilizing the same base algorithms for consistency among the different techniques. These were Logistic Regression as a discriminative model, a generative model in Naive Bayes and an ensemble method in Random Forests. Some additional experiments were conducted with more complex boosting methods like the popular XGBoost, but the execution was highly resource intensive and the results were rather disappointing. Consequently these experiments will not be reported analytically.

In the general case, to transform text into a numerical feature space, the 2 most popular sparse vectors representations were utilized. These were the Bag of Words and Term Frequency / Inverse Document Frequency methods.

Chapter 2

Dataset

2.1 Requirements

To support our study we needed to find a dataset with certain characteristics. Most importantly we needed a labeled dataset that would contain multiple labels for each of its samples. This requirement provides the necessary experimentation base both for the multi-label learning and the active learning models that we developed. Also, a dataset with high cardinality regarding labels provides an interesting case to study for the class imbalance techniques that we implemented.

2.2 Overview

Taking into account the factors mentioned in section 2.1 we singled out a dataset of approximately 14k books about Science Fiction we found on Kaggle. Those data are originally coming from Goodreads social platform. For each book we have information regarding author name, book title, book description, average rating, year of release and genres that Goodreads' users have tagged each book with, along with the number of votes on each label. The majority of the books is written in the English language and they come from approximately 3.6k authors.

2.3 Pre processing

In order to make the most out of our dataset in the applications that we wanted to use it for, we followed some preprocessing steps regarding genres and book description. In our book collection there are 506 unique genre tags which we reduce to 214 "primary" ones by removing sub-genres contained in them. Those sub-genres are keywords inside parenthesis next to genre name. Over 85% of books are tagged with 10 different genres before this process. After the extraction of the "primary" genres the distribution of genres per books changed as it is shown in the figure bellow.

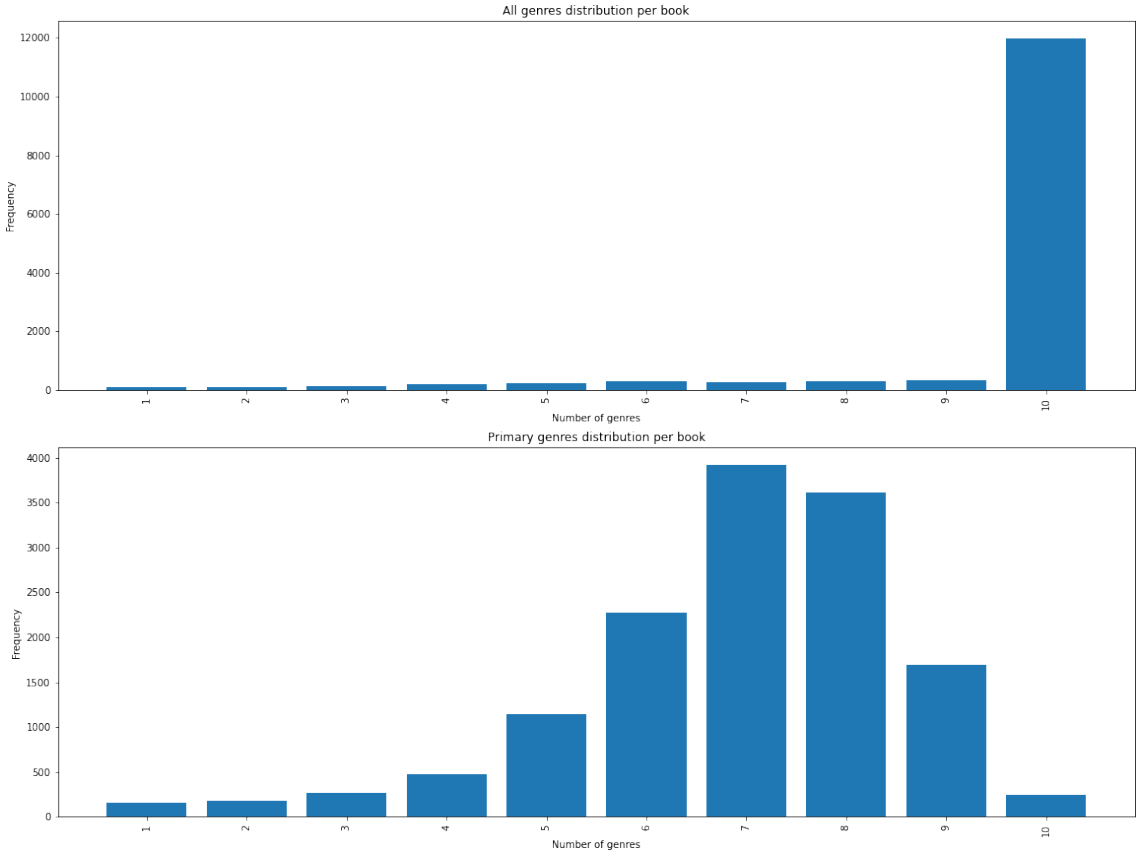


Figure 2.1: Number of genres per book distribution

Next we filtered out some of the books as there were duplicates, we kept only books with descriptions written in English language and books that had at least 10 tokens in the description. As it is demonstrated in figure 2.2 the most popular genres found in this dataset are those that are closely related to Science Fiction, which for our study are redundant as we already know that all of our books come from this specific genre. For this reason we ignored those labels and made a manual sub-selection of nine genres that would be used for the applications to come. Any books that didn't contain any of those nine genres were removed. The genres we handpicked are: Romance, Adventure, Young Adult, Space, Historical, Adult, Speculative Fiction, War, Apocalyptic

Furthermore, we made some text processing for the book description that would be used for the classification of books in our different applications. This process included, removing stop-words and punctuation, expanding contractions e.g. I'm to I am, making text lowercase and removing any character that doesn't belong to the latin ones.

For the sake of the Class Imbalance and Active Learning applications, we also generated

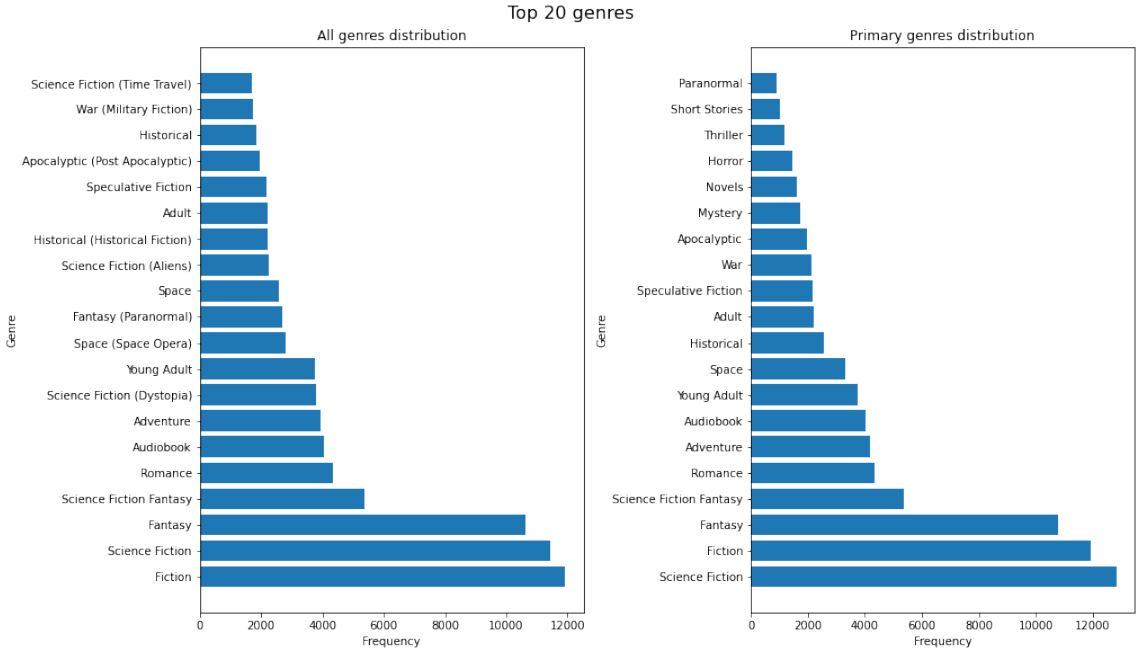


Figure 2.2: Genres distribution

a single label for each book in order to address a multi-class classification problem. This label called "major" genre represented the most voted genre (from the nine we kept) by the Goodreads community as mentioned in section 2.2. Finally, we applied one hot encoding for the multi-label genres and label encoding for the multi-class genres.

Chapter 3

Multi-label Classification

Classification of multi-genre literary books is a multi-label classification problem where the desired output of the model is a bi-partition, because a book may or may not belong to any of the target labels we have picked (section 2.3). For this task we employed four different techniques, for which we compare performance of the best model after a series of experiments.

3.1 Approach

To tackle a multi-label classification problem there are generally two categories of methods [11]. There are the Problem transformation methods, where the original problem is transformed into one or more single-label classification tasks, therefore traditional algorithms can be applied. But also, Algorithm adaptation methods, where existing algorithms are being extended in order to handle multi-label problems, such as MLkNN [15]. On the other hand, Neural networks deal with multi-label problems natively as they can provide multiple outputs for each sample provided. To enhance performance of learning from multiple labels with NNs there have been proposed adaptation methods such as BP-MLL loss function [14]. In our study we experimented with three problem transformation methods and a Bi-LSTM Neural Network.

3.1.1 Problem transformation methods

One Vs Rest

This method is the most naive one regarding multi-label learning tasks. It is a binary relevance method as it trains one classifier per target label to make a binary decision, if the a given sample belongs or not to the class this classifier represents. The main drawback of this method is that it learns labels individually ignoring label relations.

Classifier Chains

Classifier Chains is also composed by as many classifiers as the labels to be predicted. The predictions of a classifier in the chain are turned into features for the next one. This way label relation is achieved but this method is strongly depended to the order of labels in the chain.

Random k-Labelsets

This classification method breaks the set of labels in n subsets of size (k), called k-label sets [12]. For each label set a multi-label classifier is trained using the Label Powerset method. Label Powerset treats each unique set of labels as a different class in order to transform problem to a single-label classification one and for each sample tries to predict the most probable class. Rakel unlike LP achieves training classifiers over more balanced sets and can predict unseen labelsets.

3.1.2 Neural network methods

Bi-LSTM

Finally we experimented with Bi-LSTM neural networks as it is a popular NN technique when it comes to text classification. This popularity is drawn due to the fact that LSTM NNs have the ability to connect previous information to the current task, along sequences, which is really important when it comes to text data where there are long-term dependencies between words.

In order to address a multi-label task with NNs, Dense output layer must output as many values as the labels that exist and then round probabilities to get a vector of zeros and ones. We didn't use any pretrained word embeddings but we used a dropout layer to prevent overfitting. Regarding this technique we also tried using an existing python implementation of BP-MLL [14] loss function with no success.

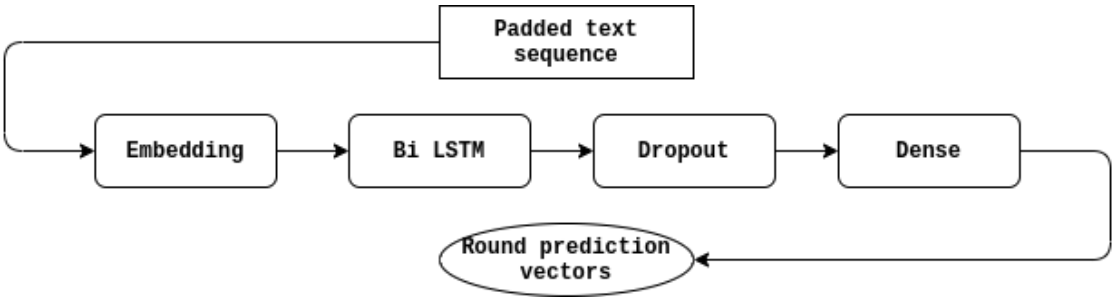


Figure 3.1: Bi-LSTM architecture

3.1.3 Base estimators

For the transformation methods that we implemented, we used three different estimators: Logistic Regression, Multinomial Naive Bayes and Random Forest.

3.1.4 Text representations

Our base estimators required transforming book description text input to vector representations. We decided to do that by using two standard word occurrence frequency based approaches, Bag of Words and TF-IDF, which generate sparse vector representations.

Bag of Words

The BoW model generates document embeddings disregarding grammar or word order but only by taking into consideration occurrence count of tokens.

TF-IDF

Similarly to BoW this model counts word occurrence in a document but in addition multiples it by the inverse frequency of occurrence of a word in all documents. This way it penalizes words that appear with high frequency in all documents such as articles.

3.1.5 Evaluation

In order to evaluate our models we measured their performance both in sample and label level. For the sample based evaluation we used micro F1 score and Hamming loss [11]. For the label based evaluation we used Accuracy, Precision, Recall and F1 score as the values in this case are binary.

3.2 Experiments

Based on the approach described in the previous section we conducted multiple experiments for our models. We fine-tuned models' parameters through this process in order to achieve best performance. Regarding the overall performance as shown in Table 3.1, One Vs Rest using Multinomial Naive Bayes as an estimator and BoW for vectorization of book descriptions was the best model. This result could mean that label relation isn't very strong in the labels we have sub selected. Classifier chains ran for different label orders without improving performance, which reinforces our previous claim. In the case of Ra*k*el we only used 10k features of BoW vectorization as the volume of data created by the labelsets could not fit in our machine's memory, so this method could potentially perform better. Bi-LSTM disregarding the Dropout layer used, overfitted after 8th epoch and performed the poorest overall.

Model	F1 score(micro)	Hamming Loss
One Vs Rest, Multinomial NB [BoW]	**0.8436**	**0.0707**
Classifier chains, Multinomial NB [BoW]	0.843	0.071
Ra*k*el, Logistic Regression [BoW 10k]	0.8037	0.089
Bi-LSTM	0.7965	0.0972

Table 3.1: Best models

In table 3.2 we present the label based evaluation results of the One Vs Rest, Multinomial NB model. As it is shown the model performed quite well in all genres. The results suggest that Adult and Speculative Fiction genres must be mostly dominated by the existence of absence of the label, due to the low recall/high accuracy.

Genre	F1 score	Accuracy	Precision	Recall
Romance	0.883	0.916	0.873	**0.893**
Adventure	0.795	0.87	0.86	0.738
Young Adult	0.872	0.925	0.9	0.844
Space	**0.914**	0.956	0.95	0.88
Historical	0.847	0.941	0.898	0.8
Adult	0.704	0.91	0.855	0.598
Speculative Fiction	0.789	0.933	0.904	0.701
War	0.847	0.949	0.887	0.811
Apocalyptic	0.863	**0.96**	**0.951**	0.79

Table 3.2: One Vs Rest label based results

Chapter 4

Class Imbalance

Class imbalance as a characteristic of a dataset, affects not only the efficiency and performance of most machine learning methods but can also easily lead to misinterpreted evaluation experiments, if not accounted for, resulting in erroneous decisions about classification performance. In the case of our dataset and classification problem, we decided to simplify the task at hand when researching different class imbalance techniques, so that we could more easily and clearly assess the experimentation results. All experimentation, as stated in 1.3, were performed with 3 different classifiers (Logistic Regression, Multinomial NaiveBayes and Random Forest) and two different vectorized representations, Bag-of-Words (Bow) and Term frequency, inverse document frequency (tf-idf).

4.1 Multiclass Transformation & Approach

The multilabel classification problem was transformed to a multiclass one, by selecting the most voted genre of each sample and dropping the rest. The labels were further processed by dropping dominant, but redundant, labels such as "science" and "fiction" since the total of the dataset refers to science fiction books. The remaining samples demonstrated the class distribution illustrated at 4.1. It is easily observable that few labels dominate the data, while others are underrepresented. As expected, in the baseline approach a number of class prediction errors can be seen in 4.2.

In order to remedy the class imbalance, the main approach followed was that of oversampling of the various minority classes. This was dictated both by the low number of initial samples that would further be diminished should we have chosen undersampling techniques, as well as the multiclass nature of the transformed problem. Several synthetic sample methods were employed with varying results. In addition a bagging method was attempted, sadly with sub par classification performance.

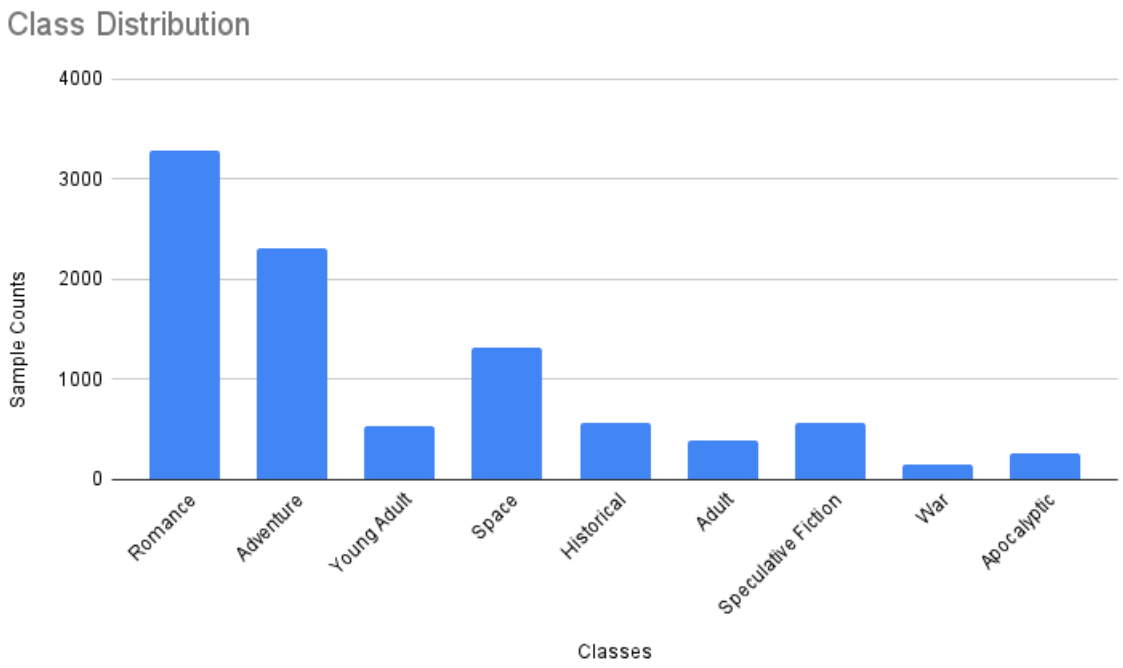


Figure 4.1: Initial Class Distribution

4.2 Techniques

4.2.1 SMOTE

The first attempt to create synthetic samples, was through the use of Synthetic Minority Over sampling Technique (SMOTE)[3]. The algorithm selects the k nearest examples, based on the KNN algorithm, randomly selects one of them and creates a synthetic sample based on the distance between the initial sample and the selected neighbour. SMOTE is expected to result in classes with the exact same number of examples, as demonstrated in 4.3. The algorithm does not take in account the properties of each neighbourhood and creates the same amount of synthetic examples for each minority sample. In our case the best results were obtained when selecting an initial count of k=5 neighbours

4.2.2 ADASYN

A slightly more sophisticated approach to synthetic sample creation is ADAptive SYNthetic (ADASYN)[6]. ADASYN operates much like SMOTE, but also calculates the ratio of all neighbours to those of the majority class in each neighbourhood. This results in the algo-

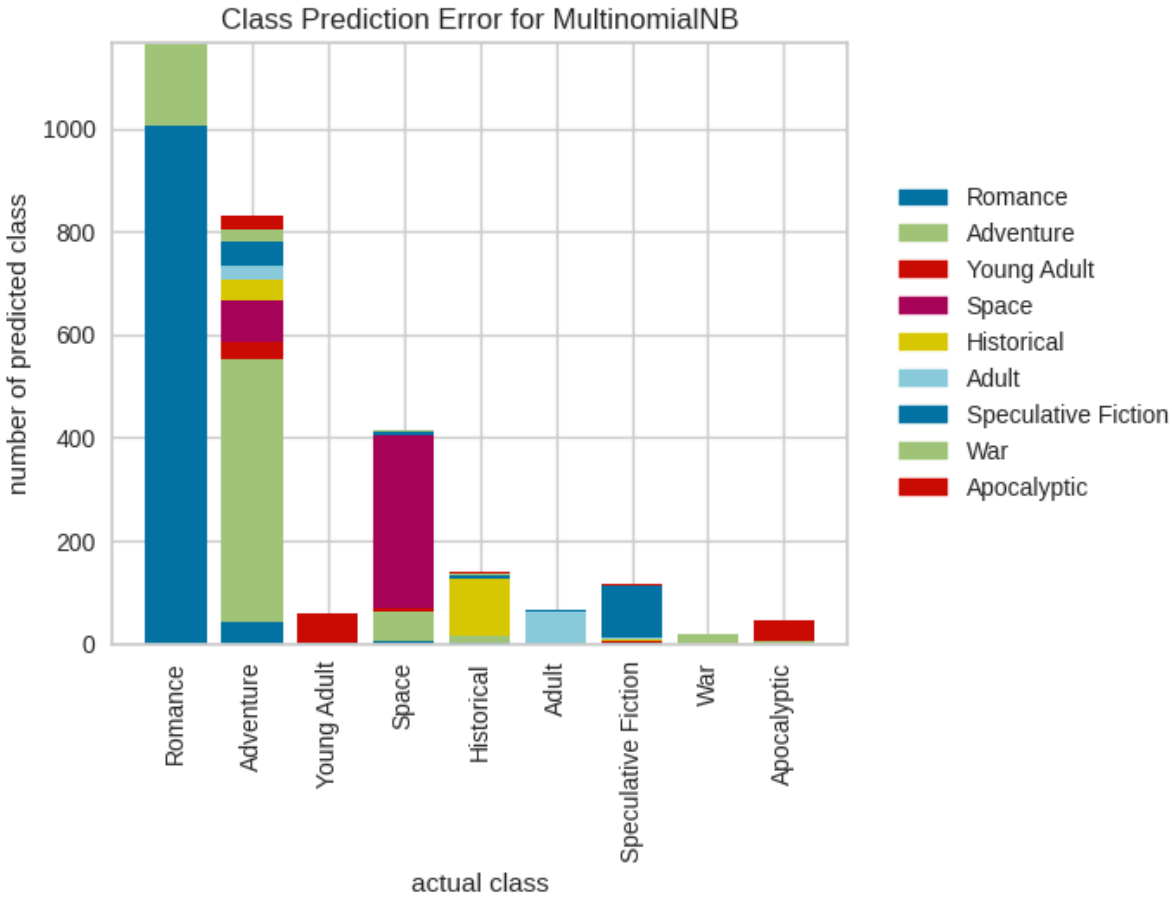


Figure 4.2: Initial Classification Error

rithm creating more examples for harder to learn classes, based on the density distribution of majority examples of each neighbourhood. There lies the "adaptive" nature of the algorithm. As such we expect ADASYN to not create the exact same amount of samples, a result that can indeed be observed in 4.3

4.2.3 Text augmentation

Traditional class imbalance techniques that transform samples in the feature space may not always be useful and lead to creating samples that do not make sense, therefore are not interpretable. For this reason research is being conducted [13] to develop techniques that balance class support based on text processing. A standard way of generating new text samples is by

replacing tokens with synonyms in existing samples, using lexical databases such as WordNet [5]. Also, randomly removing and adding (synonym) tokens can improve performance as shown in [13]. More complex methods involve translating text to another language and then back to original to create a paraphrased/similar text samples [1]. The aforementioned techniques are accepted as practices for text augmentation, but can be considered heuristic techniques to oversample minority classes. Due to this fact these techniques are heavily depended to the environment of the problem they try to solve and cannot generalize easily. After some experimentation with different techniques, we employed a strategy similar to the one shown in [10]. For each one of the classes expect the dominant one we created new samples based on the following formula.

newsamples = min([samples(majority) - samples(minority)], samples(minority)) * 3

Samples are multiplied by three because for each original one we create three new ones by applying the following techniques.

- **Synonym replacement:** Replace each token in the original sample by a random one using WordNet
- **Unique words:** Create a sample by keeping only the unique words found in it
- **Random mask:** Randomly remove 0-20% of tokens in a sample

The resulting distribution can be seen in 4.3

4.2.4 Easy Ensemble

The final method attempted was that of Easy Ensemble[7]. It involves randomly selecting the same number of samples from all of the classes, regardless of their initial distribution, thus creating balanced sub-datasets. Afterwards, boosting models are trained on the different sub-datasets and then their classification results are combined through voting. In our case, 10 Adaboost classifiers were fit on 10 resamplings of our classes, where data were randomly sampled with replacement.

4.3 Evaluation

For the evaluation of the differents class balancing methods, the 3 base classifiers mentioned were used with each class representation, excluding ofcourse Easy Ensemble where seperate Adaboost classifiers were utilized on the subproblems. The experiments were run on both Bow and tf-idf feature representations of our samples.

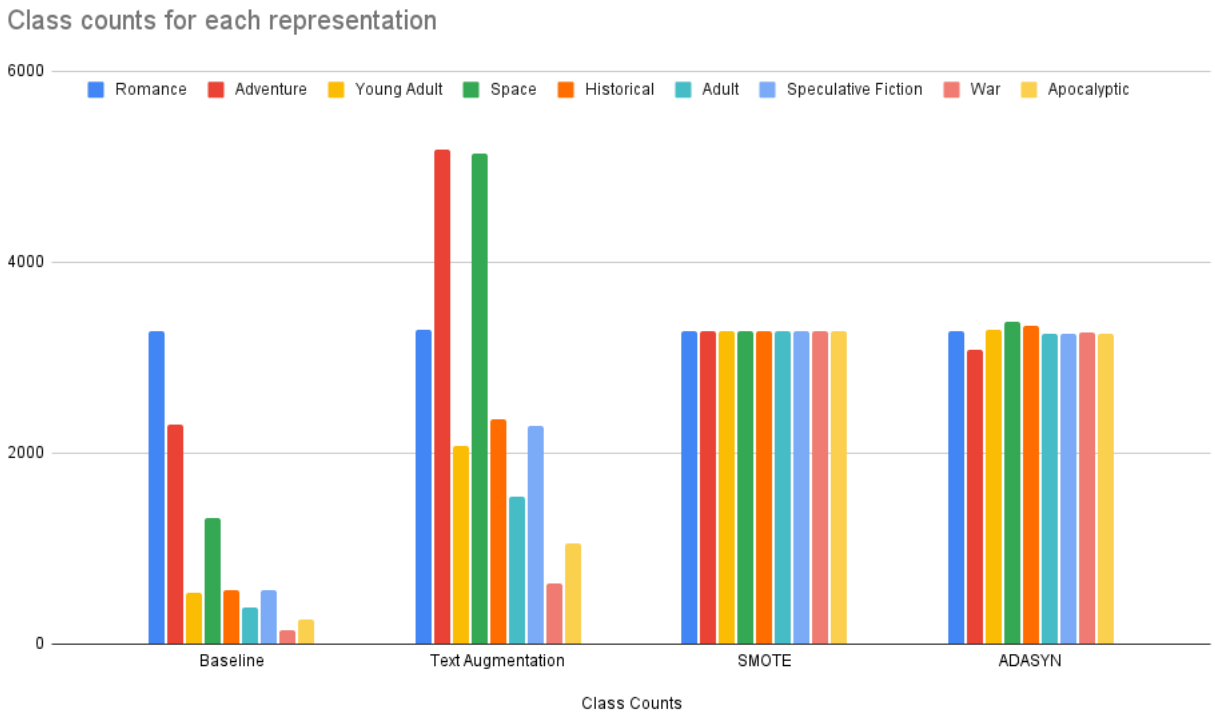


Figure 4.3: Altered class counts

4.3.1 Metrics

Accuracy is more often than not, an ill-advised metric when evaluating class imbalance problems. A completely naive classifier, that consistently predicts the majority class could yield high accuracy results, due to the dominance of said class. This is only somewhat remedied in multiclass problems where more than one class are the majority ones. In any case, more appropriate metrics need to be employed in problems that demonstrate class imbalance, such as the F1-score, Precision-Recall Curves and Receiver Operating Characteristic (ROC) curves that can more accurately depict the performance of our classifiers in predicting each class.

We considered a number of different metrics to utilise, such as Balanced Accuracy, G-Mean or the Brier Score loss but decided against them either because they were not suited for the multiclass nature of our problem, or did not fit in the physical interpretation of our problem. Classifiers that predict the genre of books, are more useful when predicting the correct genres for a majority of the samples, while mispredictions are not as costly. As a result we decided on utilizing the harmonic mean of precision and recall, namely the F1-score. Additionally, although Precision - Recall curves would be a useful metric, since this information is already

somewhat conveyed by the F1-score, we opted to show the ROC curves of the performance of our classifiers together with Area Under Curve (AUC) score, that portrays the relation between the TruePositive and FalsePositive rates of our models and gives the same weight across classes.

A final point to consider in evaluation is the means that the results of each class are averaged. Macro-average calculates the metric of each class and then averages those, resulting in a more insensitive to imbalanced datasets average. On the contrary micro-average first calculates all individual metric numbers and then averages those, thus giving weight to each class according to its respective size. Micro average however can be useful in cases where predicting correctly the positive class is preferable, as is the case in our approach.

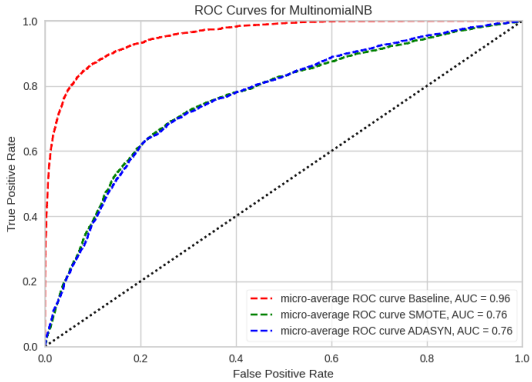
4.3.2 Results

As is easily apparent in the corresponding graphs, the typical synthetic sample creation methods yielded worse results than the baseline approach, in all cases. This can be seen both in F1-score as well as the ROC curves. Synthetic examples that stem from vectorized feature representations of text, will create samples that are devoid of meaning and may very well not convey the correct information to the classifiers. Although we can see that ADASYN did perform slightly better that SMOTE, they both were inadequate in modeling the nature of our problem.

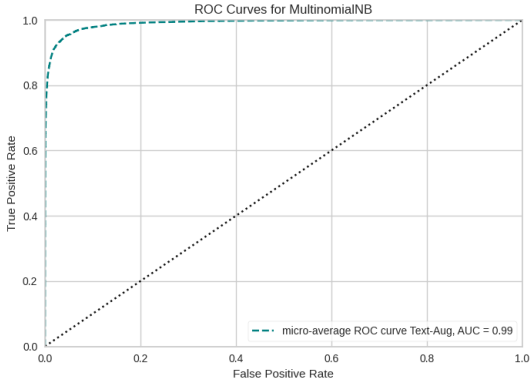
On the contrary, as illustrated in 4.3.2, 4.3.2 and 4.4 the text augmentation technique employed did in fact significantly improve our f1-score as well as ROC AUC score. Since the technique employed more accurately creates synthetic examples based on the actual text data rather than its vectorized representation, it would also prove to be more interpretable even though this was not examined in depth in our efforts.

Finally Easy Ensemble seems to perform significantly worse than all of its counterparts. This is attributed to the boosting nature of the classifiers, in particular Adaboost. This class of estimators did not perform adequately in any of our experiments, as was also the case with the use of the XGBoost classifier early in our experiments.

Bag of words - F1	Logistic Regression	Naive Bayes	Random Forest
Baseline	0.79	0.79	0.72
Text Augmentation	0.86	**0.90**	0.83
SMOTE	0.74	0.78	0.72
ADASYN	0.74	0.79	0.73
Easy Ensemble	0.36	0.36	0.36



(a) ROC AUC



(b) ROC AUC Text Augmentation

Figure 4.4: ROC Curves

tf-idf - F1	Logistic Regression	Naive Bayes	Random Forest
Baseline	0.70	0.78	0.73
Text Augmentation	0.77	**0.89**	0.81
SMOTE	0.78	0.79	0.74
ADASYN	0.79	0.80	0.74
Easy Ensemble	0.37	0.37	0.37

Chapter 5

Active Learning

In this section, we discuss the task of Active Learning and the methods we implemented to deal with the lack of labeled data, as it commonly happens in real-world problems.

First of all, Active Learning (AL) is a special case of machine learning in which a learning algorithm can interactively query a user (or some other information source) to label new data points with the desired outputs [8, 4]. The information source is often called teacher or oracle. AL is very important as there are situations in which unlabeled data is abundant but manual labeling is expensive. In such a scenario, learning algorithms can actively query the user/teacher for labels. Since the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required by selecting samples randomly.

The most naive method that can be used to query the oracle is **random sampling**, where samples are chosen completely randomly and each one of them has an equal probability to be picked. This approach is quite simple, however, it is very likely to provide the user with samples that may not be important for the learning algorithm or almost identical to samples the algorithm has already seen before. To tackle this difficulty, we can use **Uncertainty Sampling** which is a well-known AL technique that detects the most useful example in the data and presents it to the user to be labeled. Uncertainty Sampling works by first calculating the **usefulness** of prediction for each example and selects an instance based on the usefulness. The most common method to measure the usefulness of an example is the uncertainty of classification defined by:

$$U(x) = 1 - P(\hat{x} \mid x) \tag{5.1}$$

where x is the instance to be predicted and $\hat{x}$ is the most likely prediction.

Although Uncertainty Sampling works well in common situations, in some scenarios it tends to be biased towards the actual learner and it may miss important examples which are not in the sight of the estimator. To address this obstacle, three different Active Learning techniques are presented below.

5.1 Query By Committee

Query by committee is a popular active learning strategy, which alleviates many disadvantages of uncertainty sampling. For instance, uncertainty sampling tends to be biased towards the actual learner and it may miss important examples which are not in the sight of the estimator. Query by committee tackles this problem by keeping **several hypotheses** at the same time (i.e. trained classifiers) about the data, selecting queries where **disagreement** occurs between them. In our case, we selected to have three committee members.

Measuring the disagreement between the hypotheses can be done using uncertainty sampling. However, due to the drawbacks already mentioned about uncertainty sampling we have chosen to use the max disagreement sampling. Max disagreement sampling measures each learner's disagreement with the consensus probabilities and queries the instance where the disagreement is largest for some learner. For example, if the vote probabilities for each learner and the consensus probabilities are given, we can calculate the **Kullback-Leibler divergence** of each learner to the consensus prediction and then for each instance, select the largest value.

5.2 Information Density

When using uncertainty sampling (or other similar strategies), we are unable to take the structure of the data into account. This can lead us to suboptimal queries. To alleviate this, one method is to use information density measures to help us guide our queries. For an unlabeled dataset X_u, the information density of an instance x can be calculated as:

$$I(x) = \frac{1}{|X_u|} \sum_{x' \in X} sim\left(x, x'\right) \tag{5.2}$$

where $sim(x, x')$ is a similarity function, in our case cosine similarity. The higher the information density, the more similar the given instance is to the rest of the data. In this way, Information Density tries to return those samples that are as least similar to others as possible [9].

5.3 Ranked batch-mode Sampling

One of the drawbacks of standard pool-based sampling is their interactive nature: these sampling methods are best fit for cases where we have an attentive human-in-the-loop for maximizing the model's performance. Although these sampling methods do support returning multiple samples per query, they tend to return redundant/sub-optimal queries if we return more than one instance from the unlabeled set. This is a bit prohibitive in settings where we would like to ask an active learner to return multiple examples from the unlabeled set. Ranked batch-mode sampling [2] not only supports batch-mode sampling but also establishes

a ranking among the batch for end-users to prioritize which records to label from the unlabeled set. In this method, each example x is scored using the following formula:

$$score = \alpha \left(1 - \Phi \left(x, X_{labeled}\right)\right) + (1 - \alpha)U(x), \tag{5.3}$$

where $\alpha = \frac{|X_{unlabeled}|}{|X_{unlabeled}|+|X_{labeled}|}$, $X_{labeled}$ is the labeled dataset, $U(x)$ is the uncertainty of predictions for x and Φ is a so-called similarity function, in our case the **cosine similarity**. This latter function measures how well the feature space is explored near x. (The lower the better.) After scoring, the highest scored instance is put at the top of a list. The instance is removed from the pool and the score is recalculated until the desired amount of instances are selected.

5.4 Experiments

To evaluate our active learning methods we performed several experiments. First of all, we chose four different estimators [Logistic Regression, Naive Bayes, Random Forest, XGBoost], and initially trained them on the whole set of data, treating it as labeled, to estimate their full potential. The results of this analysis are presented in Table 5.1, along with the time they took to train. As can be seen, Random Forest shows the best performance among the four classifiers, however, due to its high time complexity we choose to consider Naive Bayes as the best option, since it is slightly worse than Random Forest but it is about 2 orders of magnitude faster.

Next, we conducted three different experiments where we examined the performance of our techniques in terms of the number of labeled samples they need to reach 65% of F1_score. The first and the most important experiment compares our three active learning techniques to each other and to the random sampling approach, to study the benefit (if there is any) of using such sophisticated methods in Text Classification. The results are shown in figure 5.1, with Table 5.2 presenting the running time of each method. We observe that Information Density requires the least amount of labeled examples to reach 65% of F1_score among the four different techniques, while random sampling performs the worst as expected. In specific, Information Density needs 1400 labeled samples, about 300 less than random sampling. Moreover, Ranked batch-mode sampling and Query by Committee perform roughly the same, the latter being

Estimator	**F1_score**	**Running time (sec)**
logreg	0.7047	6.95
bayes	0.7274	**0.012**
rand_forest	**0.7445**	4.77
xgbootst	0.5711	94.70

Table 5.1: Different classifiers comparison trained on the whole data.

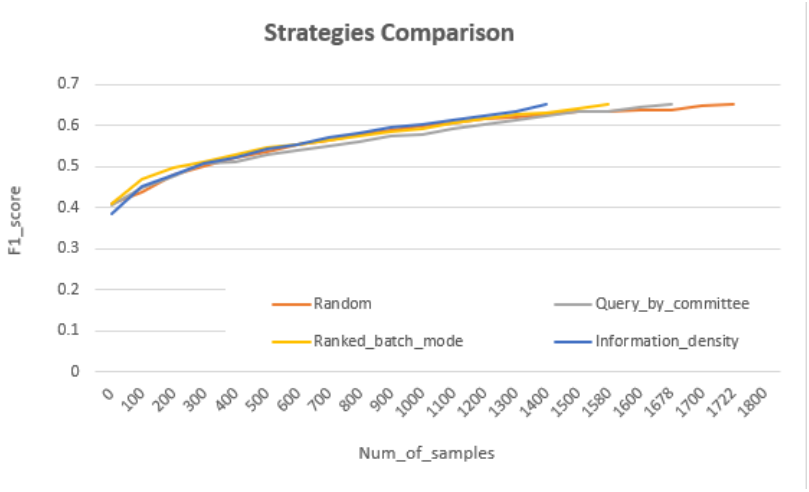


Figure 5.1: Different Active Learning techniques comparison.

Method	Running Time (sec)
Random	55.38
Query_by_committee	256.56
Ranked_batch_mode	679.63
Information_density	1707.09

Table 5.2: Running time comparison to reach 65% of F1_score.

slightly worse. Finally, as can be seen from Table 5.2, our best technique takes the most time to run, while random takes the least, indicating that there is a trade-off between performance and time complexity.

The second experiment we performed was to compare the different classifiers using the Information Density technique which proved to be the best among the ones we considered. Figure 5.2 presents the number of annotated samples Information Density requires when trained with Logistic Regression, Naive Bayes, and Random Forest. We observe that Naive Bayes manages to reach 65% F1_score with only 1400 labels samples, while it needs 1835 with Random Forest and about 2000 with Logistic Regression. This outcome comes to contradiction with the performance of the classifiers when trained on the whole set of data, where Naive Bayes is slightly outperformed by Random Forest in terms of final F1_score. However, as seen from the last experiment, Naive Bayes converges much faster to 65% F1_score.

Finally, we experimented on the initial batch size of labeled examples we provide our learners with. In specific, we considered the following batch sizes: 100, 250, and 500. The results are illustrated in figure 5.3. As expected, the more label data we provide our learners with, the less they need to label subsequently to reach 65% F1_score. In addition, it is worth

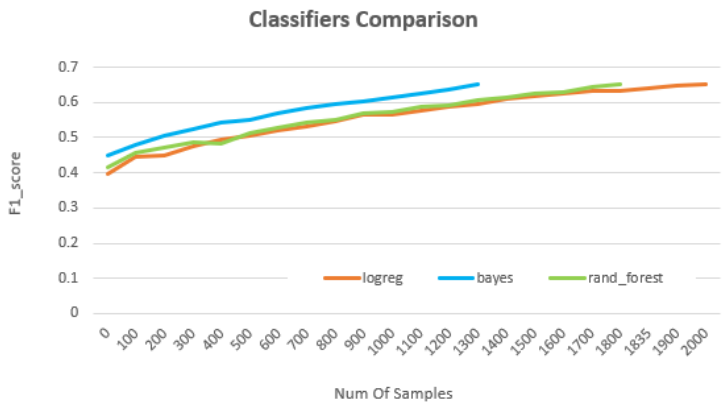


Figure 5.2: Information Density with different Classifiers.

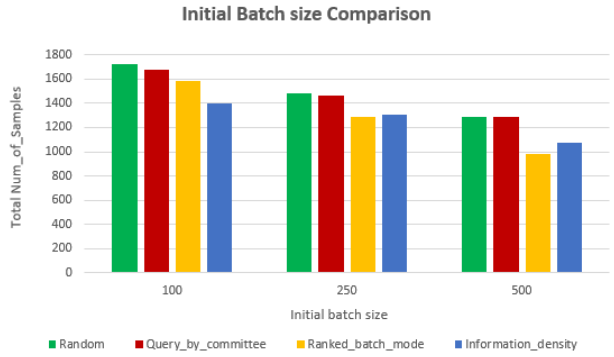


Figure 5.3: Performance comparison with different initial batch sizes.

noting that even though Information Density is the best performing method when it is given 100 initial labeled samples, Ranked batch-mode Sampling outperforms it in the case of 250 and 500 initial batches. This is important, as we notice that the initial batch size can highly affect the performance of an active learning technique, and constitutes a parameter worth optimizing and experimenting on.

Chapter 6

Conclusion & Future Work

In this work, we examined three different tasks for dealing with Book genre classification. At first, we considered the problem of Multi-Genre Classification, as each book can be classified into more than one genre, and we implemented five different methods. After several experiments, One vs Rest proved to be the best approach among the ones we studied. Next, we transformed the problem/data to multi-class as we wanted to experiment on class imbalance. For this task, we implemented four different techniques, SMOTE, ADASYN, Easy Ensemble, and a Custom Text Augmentation approach. Through various experiments, we observed that Custom Text Augmentation showed the best performance. This is mainly due to the nature of the problem we studied (Text Classification) since most class imbalance techniques are not able to create realistic synthetic data. Finally, we examined the task of Active Learning, where we implemented three different techniques and compared them to each other and the completely random approach. As expected, the random approach was inferior to the other methods. In addition, Information Density proved to require the least number of labeled examples to reach a certain performance score, indicating that data structure is worth considering when designing Active Learning techniques.

So far, we have considered class imbalance and active learning on a single-label level. In the future, we plan on experimenting with these two tasks on a **multi-label level**, which is a more challenging task but also quite more interesting for real-world applications. For this reason, after a short data analysis we performed, we observed that there are roughly 190 label permutations in our Books dataset.

*The source code of this project can be found at: https://github.com/georgSquared/Multi-genre-Classification-on-Literary-Books

Bibliography

[1] Suphamongkol Akkaradamrongrat, Pornpimon Kachamas, and Sukree Sinthupinyo. Text generation for imbalanced text classification. pages 181–186, 07 2019.

[2] Thiago NC Cardoso, Rodrigo M Silva, Sérgio Canuto, Mirella M Moro, and Marcos A Gonçalves. Ranked batch-mode active learning. *Information Sciences*, 379:313–337, 2017.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002.

[4] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. Incorporating expert feedback into active anomaly discovery. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 853–858. IEEE, 2016.

[5] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[6] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, 2008.

[7] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[8] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer, 2015.

[9] Burr Settles. *Synthesis Lectures on Artificial Intelligence and Machine Learning*. 2012.

[10] Marwan Torki, Mai Ibrahim, and Nagwa El-Makky. Imbalanced toxic comments classification using data augmentation and deep learning. 12 2018.

[11] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[12] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011.

[13] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics.

[14] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 18:1338– 1351, 11 2006.

[15] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.