TECHNICAL UNIVERSITY OF CRETE
ELECTRICAL AND COMPUTER ENGINEERING

DIPLOMA THESIS

A Novel Social Choice Mechanisms-Based Recommender System for the Movies Domain

---

Author:
GEORGIOS KLIOUMIS

Committee:
GEORGIOS CHALKIADAKIS, PROFESSOR - SUPERVISOR
MICHAIL G. LAGOUDAKIS, PROFESSOR
VASILIS SAMOLADAS, ASSOCIATE PROFESSOR

A thesis submitted in fulfilment of the requirements for the Integrated Master of Engineering degree in Electrical and Computer Engineering awarded by the School of Electrical and Computer Engineering at the Technical University of Crete

Chania, February, 2024

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Ανάπτυξη Καινοτόμου Συστήματος Συστάσεων για Ταινίες Βασισμένο σε Μηχανισμούς Κοινωνικής Επιλογής

Συγγραφέας:
Γεώργιος Κλιούμης

Εξεταστική επιτροπή:
Γεώργιος Χαλκιαδάκης, Καθηγητής - Επιβλέπων
Μιχαήλ Γ. Λαγουδάκης, Καθηγητής
Βασίλειος Σαμολαδάς, Αναπληρωτής Καθηγητής

Χανιά, Φεβρουάριος, 2024

Abstract

GEORGIOS KLIOUMIS

A Novel Social Choice Mechanisms-Based Recommender System for the Movies Domain

Recommender systems are software tools that assist users on selecting items from a large set. In this work, we propose a personalised recommender system for the movies domain that uses novel modelling techniques, as well as a social choice theory-driven recommendation process.

On the item modelling domain, our approach builds a probabilistic movie model, based on information about its genres, acquired through the use of the movie's summary along with its consensus rating. Specifically, movies in our approach are modelled as multivariate Gaussian distributions over a number of movie features defining its dimensions. These features are selected as a result of a classification stage that employs various classifiers over vectors, characterising each movie; and which themselves are acquired via a set of text vectorization techniques.

Specifically, a set of Natural Language Processing techniques are used to transform summary texts to vector representations, using (i) Term Frequency Inverse Document Frequency (TFIDF), (ii) Class Label Frequency Distance (CLFD), and (iii) Count vectorizers. Following that, we employ the Classifier Chain architecture, based on the (i) Naive Bayes, (ii) Logistic Regression, (iii) Random Forest classification algorithms, as well as a (iv) Long Short Term Memory (LSTM) neural network, to solve the multi-label classification problem of extracting the genre set. After evaluating the above techniques, we reach the decision of selecting the Logistic Regression with CLFD transformed data and the LSTM approach, as our ultimate information extraction sources.

We follow the "You Are What You Consume" Bayesian recommender approach put forward by Babas et al[1], thus also modelling users as multivariate Gaussians with the same features as the movies. The user model's updating process utilises an efficient Bayesian Learning technique, through the use of the Normal Inverse Wishart Distribution. Additionally, we use the user's appeal to popular movies, as a way of enhancing our beliefs about their less evident preferences.

Our final recommendation process uses a social choice theory mechanism based on multi-winner elections. We use two sets of voters, that assign their votes based on the probabilistic divergence of the user and the item model, on the popularity and the movie genre domain.

The experimental phase of this work was carried out among two different types of real-world users. Our results on the first set of users, characterised by their mainstream movie culture, indicate that our recommender system suggests movies that

are rated, on average, 3.4/5 by the user. Our suggestions on the second user set, characterised by their niche movie culture, achieve a mean rating of 3.5/5. Our results indicate that (i) movie summaries are indeed a valuable tool for movie classification, (ii) recommender systems can benefit from the use of probabilistic modelling, (iii) different user types can benefit from different recommendation approaches, (iv) our movie recommendation approach using social choice theory mechanisms is effective on real-world data.

Περίληψη

Γεώργιος Κλιούμης

Ανάπτυξη Καινοτόμου Συστήματος Συστάσεων για Ταινίες
Βασισμένο σε Μηχανισμούς Κοινωνικής Επιλογής

Τα συστήματα συστάσεων είναι εργαλεία λογισμικού που βοηθούν τους χρήστες στην επιλογή αντικειμένων ενδιαφέροντος (λ.χ, ταινίες, εστιατόρια, κ.ο.κ) από ένα μεγάλο σύνολο. Σε αυτήν την εργασία, προτείνουμε ένα εξατομικευμένο σύστημα σύστασεων για τον τομέα των ταινιών, το οποίο χρησιμοποιεί καινοτόμες τεχνικές μοντελοποίησης, καθώς και μία διαδικασία σύστασης βασισμένη σε μηχανισμούς κοινωνικής επιλογής.

Στον τομέα της μοντελοποίησης των αντικειμένων, η προσέγγισή μας δημιουργεί ένα πιθανοτικό μοντέλο ταινίας, βασισμένο σε πληροφορίες σχετικά με τα είδη της, που αποκτήθηκαν μέσω της περίληψης της ταινίας μαζί με τη γενική βαθμολογία της. Συγκεκριμένα, στην προσέγγισή μας, οι ταινίες προσδιορίζονται ως πολυδιάστατες κανονικές κατανομές πάνω σε μια σειρά από χαρακτηριστικά που καθορίζουν τις διαστάσεις της κατανομής. Αυτά τα χαρακτηριστικά επιλέγονται ως αποτέλεσμα μιας κατηγοριοποίησης που χρησιμοποιεί ταξινομητές πάνω σε διανύσματα που χαρακτηρίζουν την κάθε ταινία, τα οποία αποκτώνται μέσω μιας σειράς διανυσματικών μετασχηματισμών κειμένου.

Πιο συγκεκριμένα, αξιοποιούμε μια σειρά από τεχνικές επεξεργασίας φυσικής γλώσσας για τη μετατροπή των κειμένων περίληψης σε διανυσματικές αναπαραστάσεις, χρησιμοποιώντας τους διανυσματικούς μετασχηματισμούς (i) Term Frequency Inverse Document Frequency (TFIDF), (ii) Class Label Frequency Distance (CLFD) και (iii) Count Vectorizer. Έπειτα, χρησιμοποιούμε (a) την αρχιτεκτονική Classifier Chain, βασισμένη στους (i) Naive Bayes, (ii) Logistic Regression, (iii) Random Forest αλγορίθμους ταξινόμησης - καθώς και (b) ένα Long Short Term Memory (LSTM) νευρωνικό δίκτυο, για την επίλυση του προβλήματος ταξινόμησης πολλαπλών ετικετών της εξαγωγής του συνόλου των ειδών της εκάστοτε ταινίας. Μετά από την αξιολόγηση των παραπάνω τεχνικών, επιλέγουμε να χρησιμοποιήσουμε (a) τον αλγόριθμο Logistic Regression με δεδομένα που έχουν μετασχηματιστεί από το CLFD, και (b) τη μέθοδο LSTM, ως τις τελικές πηγές εξαγωγής των πληροφοριών μας.

Ακολουθούμε την Bayesian προσέγγιση συστάσεων "You Are What You Consume" των (Babas et. al., 2013), μοντελοποιώντας τους χρήστες ως πολυδιάστατες κανονικές κατανομές με τα ίδια χαρακτηριστικά όπως και οι ταινίες. Η διαδικασία ενημέρωσης του μοντέλου του χρήστη χρησιμοποιεί μια αποδοτική τεχνική Bayesian Learning, μέσω της χρήσης της Normal Inverse Wishart κατανομής. Επιπλέον, χρησιμοποιούμε το ενδιαφέρον του χρήστη για δημοφιλείς ταινίες ως έναν τρόπο ενίσχυσης των πεποιθήσεών μας σχετικά με τις λιγότερο εμφανείς προτιμήσεις του.

Η τελική διαδικασία συστάσεών μας, χρησιμοποιεί έναν μηχανισμό κοινωνικής επιλογής βασισμένο σε εκλογές πολλαπλών νικητών. Χρησιμοποιούμε δύο σύνολα ψηφοφόρων που

αναθέτουν τις ψήφους τους, βασισμένοι στην πιθανοτική απόκλιση του μοντέλου χρήστη και του αντικειμένου, στους τομείς της δημοφιλίας και των ειδών ταινίας.

Η πειραματική αξιολόγηση αυτής της εργασίας πραγματοποιήθηκε με χρήση δύο διαφορετικών τύπων χρηστών του πραγματικού κόσμου. Τα αποτελέσματά μας για το πρώτο σύνολο χρηστών, οι οποίοι χαρακτηρίζονται από την τάση να παρακολουθούν ταινίες που προσελκύουν ευρύ ενδιαφέρον, δείχνουν ότι το σύστημα συστάσεών μας προτείνει ταινίες που βαθμολογούνται με μέσο όρο, 3.4/5 από τον χρήστη. Οι συστάσεις μας για το δεύτερο σύνολο χρηστών, οι οποίοι χαρακτηρίζονται από την τάση τους να παρακολουθούν ταινίες που δεν προσελκύουν ευρύ ενδιαφέρον, επιτυγχάνουν μέση βαθμολογία 3.5/5. Τα αποτελέσματά μας δείχνουν: (i) ότι οι περιλήψεις των ταινιών αποτελούν πράγματι ένα χρήσιμο εργαλείο για την ταξινόμηση ταινιών, (ii) ότι τα συστήματα συστάσεων μπορούν να επωφεληθούν από τη χρήση πιθανοτικής μοντελοποίησης, (iii) ότι διαφορετικοί τύποι χρηστών μπορούν να επωφεληθούν από διάφορετικες προσεγγίσεις συστάσεων, και ότι (iv) η προσέγγισή μας για συστάσεις ταινίων με χρήση μηχανισμών κοινωνικής επιλογής είναι αποτελεσματική σε πραγματικά δεδομένα.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Georgios Chalkiadakis, for his guidance and dedication throughout the course of my thesis. I would also like to thank him for the knowledge I gained during my studies and the motivation that he gave me to pursue Artificial Intelligence. I would also like to thank the members of the thesis committee, Professor Michail G. Lagoudakis and Professor Vasilis Samoladas, for their helpful commenting, time and the wealth of knowledge I acquired from them during my studies. I would also like to thank my friend, Errikos Streviniotis, for his guidance and support in overcoming the challenges encountered during this research endeavour. Secondly, I would like to thank Maria Nikoloudi for the fruitful discussions and support during the development of this thesis. Finally, I would like to deeply thank my fellow dear friends and family for their love and support throughout my academic journey. Their views and ideas have been invaluably helpful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our world is marked by the swift evolution of consumer goods, requiring the modern consumer to compare and assess them based on their individual needs. The world wide web has played a significant role in this advancement, providing customers with the ease of comparison, purchase and consumption of items, through the comfort of their homes. On the other hand, the aforementioned advancement poses a great challenge. Consumers are often overwhelmed by the item choices of the modern market, resulting in an increase of impulse purchases, poor comparison judgement and a sense of chaos. Consumers often avoid thorough investigation of products, instead opting to follow trends established either naturally, through genuine product superiority or, in some cases, artificially through strategic marketing techniques [15].

Recommender systems are tools that aim at suggesting items from a large set. Their goal is to find the most similar or appealing items and present their user with a limited recommendation set, which can be either generic or tailored to the user's satisfaction. Such systems help users identify their preferences and reduce their item investigation complexity, therefore, addressing the aforementioned challenge. Their effectiveness can be seen through their common usage by large corporations, present on the world wide web. Many companies, nowadays, employ recommender systems to help their customers find suitable items, and therefore, increase their overall profit.

During the last decade, video streaming services have brought movies on the market foreground, letting people enjoy video content, through the comfort of their home [1]. Contrary to the past, consumers nowadays watch multiple minutes, or even hours, of video content every single day. Consequently, movie producers have started to rapidly film movies, in order to match the growing market demand. This approach, resulted in the huge movie selection that is available to every consumer today. Another observation that should be made is that audience, often get herded into huge blockbusters, absorbing their preferences and making the search for movies tailored to the their real preferences hard [6].

1.1 Our Contributions

This work proposes a personalised recommender system for the movies domain, based on novel modelling techniques and social-choice driven recommendation approaches. Our approach uses movie models, relying solely on movie summaries, capturing the essence of a movie's content. Through the use of Natural Language

[1] https://explodingtopics.com/blog/video-streaming-stats

Processing and classification techniques, movie summaries are utilised to define the genres that characterise a movie and ultimately, construct an accurate movie model. The users of our system are modelled using their rating feedback, resulting in a shared model structure among movies and users. Moreover, the techniques that are used on movie and user modelling are based on uncertainty, aiding in the generalisation of our approach. Lastly, the recommendation mechanism that is used for movie suggestions, uses social choice driven techniques, that take into account various aspects of a user's character, in order to ultimately present a recommendation set, tailored to their preferences.

More specifically, our item modelling approach employs the use of (i) Term Frequency Inverse Document Frequency (TFIDF), (ii) Class Label Frequency Distance (CLFD) and (iii) Count vectorizer, to transform text summaries to vector representations. Following that, with the use of the Classifier Chain architecture, we evaluate the (i) Naive Bayes, (ii) Logistic Regression, (iii) Random Forest, as well as the (iv) Long Short Term Memory (LSTM) neural network classification techniques, on the task movie genre discovery. User preference modelling, uses the aforementioned movie models and the user's feedback in the form of rating, to efficiently update our user preference beliefs. Additionally, we employ the user's appeal to popular movies, by defining the *Niche User Index (NUI)* metric, tailoring our recommendation approach to their needs. Our recommendation process uses a social choice theory mechanism based on multi-winner elections with two sets of voters, assigning their votes based on the probabilistic divergence of the user and the item model, as well as the user's appeal to popular movies, respectively.

1.2 Thesis Structure

The necessary background knowledge of this thesis, as well as the related work used, is provided in Chapter 2. That Chapter includes basic information about recommender systems, Natural Language Processing and text vectorization techniques, as well as Random Normal variables and Bayesian Learning. The outline of our recommender system modules is provided in Chapter 3. This Chapter includes the implementation strategies that were followed in order to apply the techniques, mentioned in Chapter 2, to our recommender system. Chapter 4 includes the experiments conducted to evaluate our recommendation approach, with their respective results. Finally, Chapter 5 provides an overall summary of this research, along with a set of points for future work.

Chapter 2

Background and Related Work

2.1 Recommender Systems

Many movie recommender systems use a blend of information about their topics and users, in order to succeed in the task of recommending relevant, diverse, trustworthy and to their user's liking items [15]. A set of decision processes and evaluation techniques handle the data, that the system gathers about its users and items, with the ultimate goal of suggesting e.g., what product the user should buy, what song to listen to, which movie to watch etc. A Recommender System can also have learning and adaptation abilities, which can be applied on both the user and the item domain. What makes the recommendation process a challenging task, though, is the vast majority of parameters that define users and items.

The industry uses recommender systems in various tasks. On the consumer market, recommender systems are used to display ads, create playlists, recommend movies, propose trip schedules etc. As the number of their users increases, the recommender system's models become increasingly aware of attributes, regarding their data, formerly unknown or neglected.

2.1.1 Personalised Recommendations

Personalised Recommender Systems are a specific set of Recommender Systems, which focus on suggesting items that are tailored to their user's preferences. Contrary to personalised approaches, some recommender systems can suggest items without taking into account a specific user. These systems aim at suggesting popular and generally liked items. Each technique has their own advantages and should be selected carefully, taking into account the environment on which it will be deployed on. Certain item domains, such as music or movies, propose a wide user preference space. Each person develops a certain taste for movie genres, which depends heavily on life background, psychological state etc. Thus, the movies domain can benefit from a personalised recommendation process.

2.1.2 Non Content-Based techniques

Recommender system can use non content-based and content based filtering techniques. Collaborative filtering, which is a non content-based filtering technique, takes advantage of the fact that similar users, who "consume" items with related satisfaction, will find each other's suggestions useful [15]. These techniques search for similar, to the current subject's history, users and recommend items that the peer

set of users have enjoyed. Collaborative filtering possesses the advantage of not requiring item modelling and analysing. This advantage eliminates a large set of the aforementioned unknown or neglected parameters, reducing the system's complexity. Disadvantages of collaborative filtering include cold start issues, as the system cannot perform, if adequate user history is not present. Another disadvantage is the lack of scalability and sparsity, becoming noticeable as more users utilise the system. Large item sets, found in real world examples, cannot be evaluated extensively by the user set, thus resulting in a constantly limited recommendation spectrum.

Many proposed movie recommender systems use collaborative filtering techniques to some extent. The use of collaborative filtering in the movies domain can eliminate potential risks during modelling, by entailing movie features to the user herself. As noted above, though, collaborative filtering has some serious drawbacks that influence the quality of recommendations. Use of collaborative filtering can result in a situation where the recommendation space is restricted. Consequently, the user may not be able to explore the whole item set, due to restrictions introduced by their peer users.

2.1.3 Content-Based techniques

Another filtering technique is content-based filtering [15]. This technique relies on accurate modelling of items and user's profiles, through feature analysis. Content-based filtering is used when a large amount of data is known about the items and thus, modelling can exploit this knowledge to build accurate and efficient representations. The user model can be generated through past system interactions (ratings) and can be adapted through user feedback. The recommendation process relies on the user's past ratings and current exploration area to find items that are relevant to the user's history. Content-based techniques tackle the cold start challenge, as item models are only based on already known data, and scalability, as the number of the users in the system has no impact on the recommendation process.However, Such techniques suffer from relying heavily on item feature extraction. Modelling items can be hard task, as processing every available item feature, can have a huge cost, while feature elimination can lead to information loss. Moreover, extensive analysis of an item, with the goal of obtaining its features, cannot always be successful. A hybrid approach, regarding the above filtering techniques and others, is often the sweet spot that many commercial recommender systems use. In the movies domain Netflix is a prime example, as it uses both collaborative filtering and content-based filtering [4]

Content-based filtering, proposes the challenge of accurate movie modelling. Many implemented content-based recommender systems, on the movies domain, use a set of easily sourced features to model a movie. These sets of features include movie characteristics that potential users may already base their movie selection process upon or , on the contrary, completely ignore. Examples of those features include movie budget, cast, director, language, poster, sold tickets, genres, expert ratings etc. A movie is certainly defined, to some relative extend, by the above characteristics, but the great question quickly arises: Are they enough?

2.1.4 Modelling movies through summaries

In the movies domain, recommender systems can use a variety of information. User's and expert's ratings on movies, movie genres, budget, actors, directors, language,

length are only some of the features that can be used to model a movie. Our recommender system uses movie ratings and the movie's summary as the main information extraction sources.

A written text, describing a movie, can provide the consumer with a large amount of data. A source of the aforementioned written text, proposed in past work, is the dialog parts of a movie's script. These texts can easily be obtained from the subtitle (*.srt*) file. Cast interaction is one of the key parts that define a movie's characteristics. Thus, it can be used to infer the majority of the movie's features stated above. Examining this source of data, though, can quickly prove to be a particularly challenging computer task, as movie scripts can get quite large and their information density quite sparse.

A movie's summary is probably the most informative source that a user can examine before purchasing a theatre ticket. The summary contains data that describe many key features of the movie, that cannot be easily listed and understood through reviews, scoreboards or genre tags. Extraction of genre-related information, from a movie's summary, is a promising idea. Various works have approached item modelling on the movies domain, but our work does employ explicit genre classification techniques [1]. As stated above, a movie's summary is a dense pool of information regarding the respective movie. Therefore, using information extracted from summaries can aid movie modelling algorithms to an unknown extend. The main idea of this recommender system is to model the movie's genres through their summary, using natural language processing.

2.2 Natural Language Processing

Natural Language Processing (NLP) is a branch of computer science that handles the ability of computers to understand human generated text. NLP uses techniques that model text through a combination of language rules, statistics, machine learning and deep learning in order to understand the full meaning, as well as the author's intends. Understanding the meaning of a document can be used for its classification or to find similar documents that share certain characteristics.

Our recommender system focuses on statistical and neural NLP on movie summaries, to obtain the information needed to classify movies to certain genres. A set of NLP techniques is used to preprocess summary texts, understand the importance and relevance of certain words to specific genres and provide input to the movie classification process. The preparation of text data is done through frequency base vectorizers. Frequency based vectorizers transform text to a vector representation and give an insight to the relevance of a word to a document or a set. There are many frequency base vectorizers found in literature, but the vectorizers used in our recommender system are the *Term Frequency-Inverse Document Frequency (TF-IDF)* [3], the *Class Label Frequency Distance (CLFD)* vectorizers [10] and the *Count Vectorizer* [13].

2.2.1 Frequency Based Vectorizers-TFIDF

The TF-IDF vectorizer is based on the *Term Frequency* (*TF*) as well as the *Inverse Document Frequency* (*IDF*) component. The TF-IDF value increases proportionally to the frequency of a word's appearance and is offset by the number of documents in the corpus, that contain that word. More specifically, The *TF* component describes the

occurrence frequency of a word in a set. Regarding the document domain, the TF component is calculated as:

$$TF(t,d) = \frac{\text{occurrences of term } t \text{ in document } d}{\text{total number of terms in } d} \tag{2.1}$$

The TF component measures the importance of a word based on its frequency. Frequent words, though, such as pronouns and common verbs (e.g. like), have a high frequency in almost every document, but provide little, to no, information about its contents. This is the reason the IDF component is utilised, in order to compare term frequencies between documents of the corpus, and extract further information about a term's importance. The IDF component is calculated as:

$$IDF(t,D) = \log\left(\frac{\text{total number of documents in corpus } D}{\text{total number of documents containing term } t}\right) \tag{2.2}$$

As the number of documents containing term t increases, the value inside the log approaches 1, and therefore the IDF component approaches 0. The TF-IDF metric is the product of the above statistics stated in (2.1) and (2.2). It is calculated as:

$$TFIDF(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{2.3}$$

where t is the term in document d, whose metric is calculated, and D is the corpus of documents. The TFIDF value represents the proportional importance of a term t, on a certain document $d \in D$. A high TFIDF value is obtained through a high TF and a low IDF, or document frequency, in the corpus. Consequently, a low value can be obtained by a low TF or a very high IDF component, which bring the TF-IDF value, closer to 0.

2.2.2 Frequency Based Vectorizers-CLFD

The *Class Label Frequency Distance (CLFD)*, proposed by [10], is text vectorization technique that produces a term-based metric, that describes the importance of terms in different classes of documents. It was initially used for the binary text classification task of detecting fake news.

In order to define the *CLFD* metric we first need to define *class label frequency ratio (clfr)*. In a corpus C, which includes a set of documents D, clustered in i classes, the *clfr* metric determines the importance of a term t in a particular class of documents. A formal definition of *clfr* regarding a term t, in a class of documents i, as stated in [10], is:

$$clfr_i(t) = log_e\left(\frac{1 + occurrences(t,i) \cdot total(\bar{i})}{1 + occurrences(t,\bar{i}) \cdot total(i)} + 1\right), \forall t \tag{2.4}$$

where *occurrences(t,i)* are the occurrences of term t in the set of documents D_i, of class i, *total(i)* and $total(\bar{i})$ are the total occurrences of term t in documents D_i, of class i, and documents $D - D_i$ of every class other than i, respectively. The $clfr_i(t)$ vectors calculated above are used to calculate the *clfd* metric, which is a class-wide metric and is relevant only to the term t:

$$clfd(t) = max_{i \in C}(clfr_i(t)) - min_{i \in C}(clfr_i(t)) \tag{2.5}$$

The *clfd* value is essentially the maximum distance of the *clfr* values, found in the corpus C. The *clfd* weights for each term of the corpus can be used as an insight to

the importance of each term. In order to achieve vector transformation of text data using *clfd* weights, we take the Hadamard product between *clfd* weights, for every term t found in the corpus D, and a frequency based vectorizer, which will be the $TF - IDF$ vectorizer in our case, as follows:

$$clfd^d(t) = clfd(t) \circ tfidf(t, d), \forall t \tag{2.6}$$

With the above product, we insert the document-based component to the *clfd* vectors, generating a universal vocabulary of the terms.

Compared to term frequency (TF) vectorization, CLFD contains a natural filter to stopwords noise, due to the use of the class label frequency ratio (clfr) component.

2.2.3 Frequency Based Vectorizers-Count Vectorizer

The Count Vectorizer is the simplest transformation used in this work. Count Vectorizer has no natural stop-word filtering and cannot carry information about the importance of a document's terms, contrary to the aforementioned two techniques. Count Vectorizer transforms a text to a vector, using the frequency of occurrence for each term called *document frequency - df*. The output of this technique is a matrix of size corpus_length x terms_set_length, with each cell containing the frequency of a certain term in the respective document.

2.3 Multi-label Classification

Classification is the process of finding the categories or classes an observation belongs to [11]. In machine learning, classification is used to infer the classes an object belongs to. Classifying objects, introduces a level of abstraction on data, which can aid in modelling, problem solving, algorithm adaptation and many other tasks.

Data cannot always get classified with the same technique. Due to their different nature, several classification techniques exist. *Binary Classification* is the task of finding, among 2 classes, the one that an object belongs to, through the assignment of a binary tag to each class. This technique is used in many data domains such as spam or no-spam emails, fake or real news, e.t.c.. Another classification technique is *Multi-class Classification*. This technique assigns binary tags, among more than 2 classes, with only a unique tag, to which the object belongs to, being true. An example use of this technique is the classification of the IRIS Dataset that includes information about flowers belonging to one of 3 possible species. The classification of the IRIS Dataset cannot be accomplished through the use of classic Binary Classification, which shows that classification techniques should be used with respect to the data. Another classification technique is *Multi-label classification*. This technique is used to find the categories that an observation belongs to. The difference between Multi-class Classification and Multi-label classification is that in the later, an observation can belong to multiple classes, thus the binary tag output vector of an observation can be true, on various classes.

Our work focuses on the movies domain using classification in order to find the genres that a movie belongs to. Many movies cannot be adequately characterised by only one genre. A great example are dramedy movies which belong to the drama

and comedy genre. As a movie can belong to one or multiple genres, the use of Multi-label Classification techniques is necessary.

2.3.1 Classifiers

This section gives an oversight on the classification techniques that are used in this work. Classification is used to find the relevant movie genres based on vector transformed summaries. All the classifiers used, will be adapted to our multi-label classification domain.

The *Naive Bayes Classifier* is a simple machine learning classification algorithm that is based on Bayes' theorem [11]. This classifier assumes that the features describing a data point are conditionally independent, with respect to class labels. This classification technique calculates the probability of data point belonging to a class, based on the observed values of its features. Despite being easy to implement, Naive Bayes Classifier is a powerful tool when it comes to text classification and is commonly used in spam filtering and sentiment analysis.

Logistic Regression, despite its name, can also be used as a classification algorithm [11]. It models the probability that a given data point, defined by its features, belongs to a certain class, using the logistic function. In the context of text classification, the features often represent various aspects of the text, in our case these features are word frequencies. Logistic Regression analyses the aforementioned features and makes predictions about the likelihood of a text belonging to a particular class.

The *Random Forest Classifier* is an ensemble learning method, which constructs multiple decision trees during training and combines their outputs for superior classification performance [9]. Injecting randomness during both training and prediction ensures diversity among decision trees, preventing overfitting and promoting generalisation to unseen data. Its effectiveness can be noticed in handling high-dimensional datasets and providing insightful feature importance rankings.

Long Short-Term Memory-LSTM is a type of Recurrent Neural Network (RNN) architecture designed to address the challenges of learning long-term dependencies in sequential data [17]. LSTMs are particularly effective for tasks involving sequences, like natural language processing. Unlike traditional RNNs, LSTMs incorporate a memory cell and a set of gates that control the flow of information, allowing them to capture and retain relevant information over extended sequences. In our case, the first layer of the LSTM neural network, directly impacted by the Count Vectorizer transformation of text summaries, is a *Word Embedding* layer. Word embedding is a technique that transforms words into dense vector representations, enabling the capture of word dependencies, meaning and context. More specifically, the representation of a term is a real-valued vector that captures the semantic meaning of it, in such a way that term vectors, with similar meaning, appear to be closer in the vector space.

2.3.2 Classifier Chain

The Naive Bayes Classifier and the Logistic Regression cannot be directly applied on the multi-label classification domain, due to their binary nature. LSTMs, on the other hand, use the neural network architecture, which can produce multi-label classification results. A famous technique to tackle multi-label classification problems, using binary classifiers, is the *Binary Relevance Method-BM*. BM uses $|L|$ classifiers, where L

is the set of discrete labels, associating each one to a label. Each classifier, associated with a label, and is able to predict if the data point corresponds to the respective label or not. This technique assumes that labels are not correlated, which in many cases does not apply to the data. In our movies domain, labels or genres are indeed correlated to some extend, e.g horror and thriller movies. This lead to the decision of using a technique based on BM that tackles the label correlation problem.

The technique that is used on our multi-label classification problem, is the *Classifier Chain (CC)* architecture [14]. This technique uses the low computational complexity of BM, while respecting label dependencies. In the CC architecture, $|L|$ classifiers are used in a chain, connected one after the other, with each classifier associated with the classification problem of a discrete label $l_i \in L$. The feature space of each classifier, in the chain, is extended from the initial features of the data point, by the output of the previous classification links. This means that classifier j has a feature space $(X, CC_0, ..., CC_{j-1})$, where X is the feature vector of the data point and CC_k is the output of the k^{th} link. Associating every classifier with the extra feature set, which includes the outputs of the previous classifiers, respects label dependencies during the training of the CC model. The order in which the $|L|$ classifiers are linked in the chain plays a significant role in the performance of classification, as the order can reveal, or shadow, certain label correlations.

2.4 You Are What You Consume

The item and user models, used in this work, are based on the *You are what you consume* idea, that was mentioned in the work "A Bayesian Method for Personalised Recommendations"[1] and describes the close correlation between user habits and their preferred items. As mentioned in the above research, the famous Greek philosopher Aristotle quoted that *"We are what we repeatedly do. Excellence, then, is not an act, but a habit"*. This idea describes that humans tend to mainly focus on areas within the realm of their background, with the ultimate goal to excel. In the consumer domain, this idea describes the tendency of the customer to consume items that have previously lead to satisfaction. Steve Jobs used this notion on the following quote, *It's not the customers' job to know what they want*. The items that we consume impact our personality and our future choices. This leads to the notion that our character is comprised from the items we use, thus testing new items certainly leads to a change of ourselves. The above ideas can have a great impact on the recommendation domain and the general understanding of user preferences.

2.4.1 Gaussian or Normal Distribution

The core behind 'You Are What You Consume' is the Gaussian or Normal Distribution, which is the continuous probability distribution of normal random variable. This work uses the normal distribution for the probabilistic modelling of the items and users of the system. A normal distribution is defined by the value μ, which is the mean or expected value of the random variable and σ, which is the standard deviation. The variance of the distribution is σ^2. The formal representation of a normal distribution is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

This Figure 2.1 represents a set of univariate normal distributions with various μ and σ^2 As we can see, the variance σ^2 controls the spread of the distribution. The mean

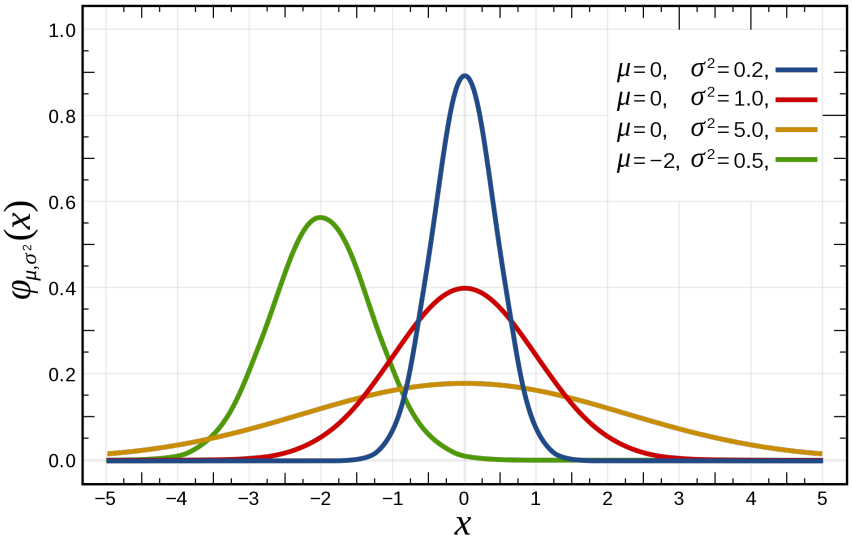


FIGURE 2.1: A set of univariate normal distributions.

μ is able to shift the distribution and represents the most probable outcome.

The *multivariate normal distribution* is the generalisation of the the univariate normal distribution in the multi-dimensional space. The formal representation in a multidimensional space of D dimensions is:

$$f(x) = (2\pi)^{-k/2} \det(\mathbf{\Sigma})^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

where $\in D$ is the mean vector and $\Sigma \in DxD$ is the covariance matrix. Figure 2.3 represents a *multivariate normal distribution* in 2-dimensional space.

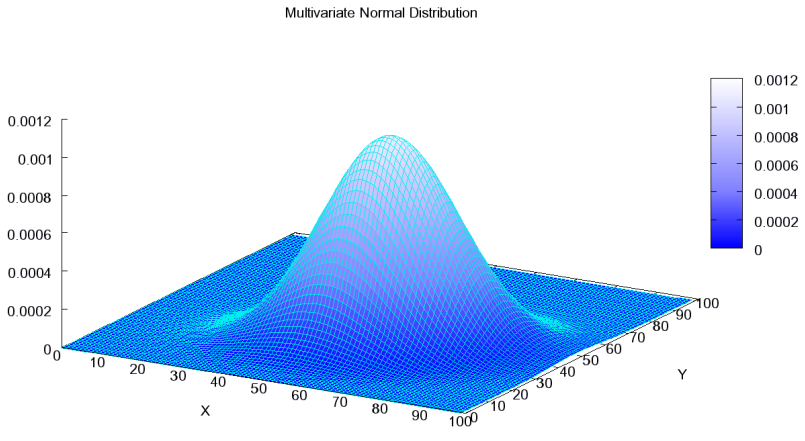


FIGURE 2.2: A multivariate (bivariate) normal distribution.

2.4.2 Bayesian Learning

In a vast number of situations a researcher needs to infer an unknown probability distribution. A method that can be used to infer an unknown distribution is *Bayesian*

Learning. This method derives a posterior probability through an updating process, using a prior probability representing the model's beliefs, and a likelihood function obtained from the observed data. The update of the prior probability from the posterior is done based on the Bayes' Theorem:

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

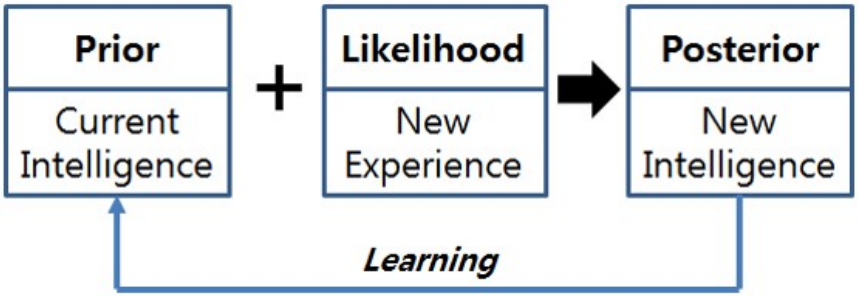


FIGURE 2.3: Bayesian Learning Process.

When the prior and posterior probabilities belong to the same type of distribution, they are called conjugate. *Normal Inverse Wishart Distribution - NIWD* [16] is 4-parameter family of probability distributions that is a conjugate prior of a *multivariate normal distribution*, with unknown mean and covariance matrix. The 4 parameters of a Normal Inverse Wishart Distribution - NIWD distribution are $\boldsymbol{\mu}_0 \in \mathbb{R}^D, \lambda > 0, \boldsymbol{\Psi} \in \mathbb{R}^{D \times D}, \nu > D - 1$ NIWD offers an efficient form of Bayesian updating procedure, as stated in the work of *"You Are What You Consume"* [1], through the update of the 4 hyperparameters based on the observed data. The use of NIWD for efficient Bayesian Updating procedures is present on various works found in literature [18], [2]. The updating process is:

$$\begin{aligned} \boldsymbol{\mu}_n &= \frac{\kappa_0}{\kappa_0 + n} \cdot \boldsymbol{\mu}_0 + \frac{n}{\kappa_0 + n} \cdot \bar{x} \\ \kappa_n &= \kappa_0 + n \\ \nu_n &= \nu_0 + n \\ \boldsymbol{\Psi}_n &= \boldsymbol{\Psi}_0 + \boldsymbol{S} + \frac{\kappa_0 \cdot n}{\kappa_0 + n} \cdot (\bar{x} - \boldsymbol{\mu}_0) \cdot (\bar{x} - \boldsymbol{\mu}_0)^T \\ S &= \sum_{i=1}^{n} (x_i - \bar{x}) \cdot (x_i - \bar{x})^T \end{aligned} \tag{2.7}$$

where x_i are the observed samples, $\bar{x}$ is the mean of the samples and n is the number of the samples. When we need to derive a multivariate normal distribution's parameters from the NIWD then the covariance matrix and mean vector, of the updated beliefs, are:

$$\boldsymbol{\Sigma} \sim \mathcal{IW}(\boldsymbol{\Psi}_n, \nu_n)$$

$$\boldsymbol{\mu} | \boldsymbol{\Sigma} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma} / \kappa_n)$$

In this work the NIWD and its updating process are used to model and update the system's users beliefs. A similar way of item and user modelling is used on the work of [21]. The proposed recommender system of the aforementioned work uses Bayesian Learning as well as Reinforcement Learning for the user's model updating

procedure. The user model updates are done using the *Dirichlet* distribution, which is a conjugate prior of the Multinomial distribution.

2.5 Social Choice Theory

Social Choice Theory is a theoretical framework that analyses ways of combining user preferences to achieve the maximisation of their utility or social welfare. Social Choice Theory applies to various domains such as recommender systems, political science, economics, e.t.c. In general, Social Choice Theory applies aggregation mechanisms on defined user preferences, in order to maximise their collective satisfaction (utility). More specifically, each user, or voter, u can rank a set of items, or alternatives, I in the order of their satisfaction. Social Choice Theory mechanisms can be applied in order to elect a single item, or a committee, with respect to user preferences.

A voting rule that is used in this work is *Borda Count*. This voting rule uses the positional ranking of items, created by the user's preferences. Each item is voted n times, where n is the position in the ascending user's preference ranking, so that the most preferred item receives k votes, where k is the size of the item set I.

Another social choice mechanism that is used in this work is *Multi-winner elections*, which is a set of election mechanisms that are able to elect a set of items, instead of a single winner. Various domains can benefit from multiple winners, including the movie domain. In our work, Social Choice Theory is used to select the movies that the system will recommend to the user. Multi-winner election mechanisms give us the opportunity to recommend multiple items to the user, thus representing, to a greater extend, their non-trivial preferences. These election mechanisms have been used extensively in literature, regarding personalised recommendations [18], [19].

2.6 Dataset

As stated in IMDB's guide for plot submission [8], a plot describes the sequence of events in a movie. There are three types of plots. The outline is a brief explanation of events taking place in the movie and is the shortest, in terms of word count, type of plot. The summary provides an in-depth overview of the movie's events, aiming to inform viewers about the story-line of the movie, before they watch it, leaving spoilers and cliff-hangers out. A synopsis, has the same characteristics as the summary, but should be able to inform a viewer about a movie's story-line, even if the viewer does not, ultimately, watch the movie. Using a movie's plot, inserts the NLP idea of summation in the data itself. A movie's summary is a text, quite carefully composed by humans, with the goal of describing the movie plot, to certain types of audience.

The Movies Dataset [20] is an ensemble of movie data, collected from the Full MovieLens Dataset [5] and The Movies Database (TMDB) API. This dataset consists of 45.000 movies released before July 2017 and 270.000 unique real users, with their respective movie ratings. For each movie, a large set of metadata, including the movie's summary and genres, is provided. Many people disagree on the number and types of genres that a movie can belong to. This dataset defines a list of 18 movie genres:

Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western

The dataset labels movie summaries as overviews, a term not officially used by IMDB. However, upon examining samples, it becomes evident that these overviews function as summaries. Ratings are given by a user to a certain movie, in a scale of 0.5-5. We should note that users have rated a variable number of movies. The Movies Dataset [20] was selected as it combined both the data, regarding movies and users, of the MovieLens dataset and the summary corpus needed to extract further information. We will provide all the details regarding dataset exploration in Chapter 4, specifically in Section 4.1.

Chapter 3

Our Approach

Our recommender system's architecture is summarised on Figure 3.1. Through the aforementioned figure, we observe the main modules that are used on our system: *i*) Preprocessing, *ii*) Movie Modelling, *iii*) User Modelling, *iv*) Recommendation Process. The Preprocessing module prepares movie summaries for the classification and modelling phase. The Movie Modelling module produces movie models, based on the preprocessed summaries and the information found on our dataset. The User Modelling module uses the aforementioned movie models, along with user generated data, to build an accurate user model. Lastly, the Recommendation Process module uses the generated user model, of a certain user, to recommend a set of movie models that are close to her taste. Each module will be elaborately analysed on the following sections.

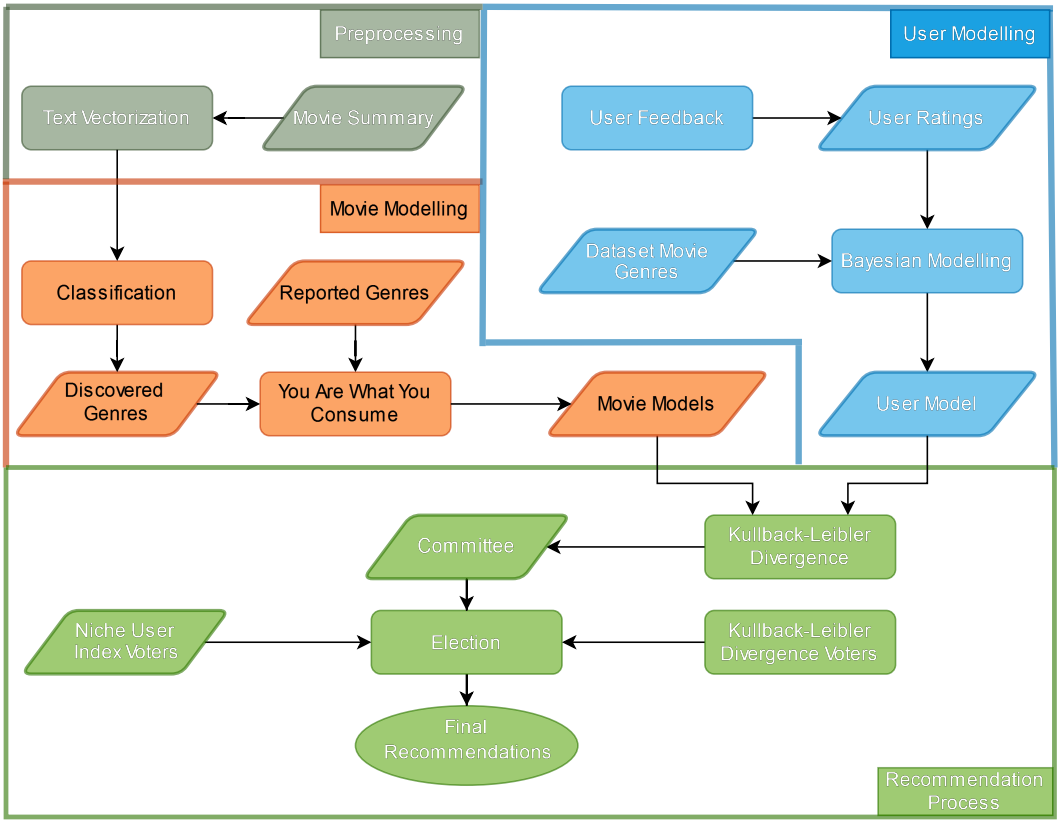


FIGURE 3.1: Outline of the Recommender System.

3.1 Preprocessing

The Movies Dataset contains various kinds of data that we need to manipulate in order to extract useful information, that will be used for accurate movie and user modelling. The data that will be used in the modelling process are the movie's reported genres, summaries and user ratings.

3.1.1 Movie Ratings and Users

A generally accepted movie quality metric, is the mean rating that of viewer's feedback. Many well-known sites providing movie information, argue that viewer's ratings are not that reliable regarding the quality of a movie, and thus, use ratings provided by expert movie critics to define the aforementioned metric. Using expert ratings can introduce a bias on the recommendation process, as the goal is to suggest movies to a simple user and not to an expert. This is what leads to the decision of defining a movie's score as the mean user rating value, calculated over the large number of user feedback provided in the dataset. This movie score, also called movie rating, is defined in a scale of $[0.5 - 5]$ points, with half-point increments.

3.1.2 Summary Preprocessing

Preprocessing is the preparation and transformation of data with the ultimate goal of feeding them into a model. In our recommender system, preprocessing should be applied to the summaries, provided for every movie, in text form. Text data is a quite unstructured source of information and skipping this step, could lead to the famous phrase "GIGO-Garbage In Garbage Out". When data contain noise or high levels of unnecessary information, the training process can struggle in knowledge discovery.

A text is defined by words, punctuation, length, grammar, syntax, etc. These features contain various degrees of information, therefore requiring a set of techniques that get rid of features providing little to no information. The use of text filtering techniques is able to relieve the computation cost of modelling and increase the overall accuracy.

In this recommender system, we should prepare the summary for the text vectorizers mentioned above. Text vectorizers create a set of terms (words), that is called vocabulary, which contains every unique term the vectorizer encountered while processing the corpus. Naturally, certain words contain less information than others, serving more in the textual connection or clarification domain, rather than the actual definition of ideas. These words are called *stopwords*, defined as the set of commonly used words in a language. Examples of English stopwords include "a", "the", "are" etc. Removing stopwords in the vectorization process introduces a natural filter to less useful features, without information loss. Using the Natural Language Toolkit (NLTK) [12] in Python, we were able to fetch an English stopwords dictionary and filter out English stopwords from movie summaries. Another filtering technique applied to the summary texts, is the removal of punctuation marks and words containing letters, not found in the English language (found most of the times in character and location names).

Transforming the already filtered text, is also a part of preprocessing that can aid in information compression. Many words can be found both with capital case, for

emphasis or in the beginning of a sentence, and in lower case. These words, though, have the same meaning in either case. Thus, we chose to lower the case of all the words found in the filtered texts. Another very useful technique of text transformation that aids compression, is the technique of stemming. In the domain of information extraction, *Stemming* is the reduction of words to their word stem. The word stem is the root of a word, which is used as the basis of derivative word generation. A word stem can be a valid word, used in a language as is, or a fragment, that is not a complete word on its own. Condensing words, derived from the same stem, into their root form and treating them as a single term in the vocabulary, typically enhances model performance. Although there is some loss of information due to reduction, this is outweighed by the efficiency gained through compression. In our work, stemming is applied using the *PorterStemmer* module of the Natural Language Toolkit (NLTK) package.

For example we can consider the summary of the movie *Toy Story* and observe the results of our preprocessing stage.

Original

Led by Woody, Andy's toys live happily in his room until Andy's birthday brings Buzz Lightyear onto the scene. Afraid of losing his place in Andy's heart, Woody plots against Buzz. But when circumstances separate Buzz and Woody from their owner, the duo eventually learns to put aside their differences.

Preprocessed

led woodi andi toy live happili room andi birthday bring buzz lightyear onto scene afraid lose place andi heart woodi plot buzz circumst separ buzz woodi owner duo eventu learn put asid differ

3.1.3 Feature Elimination

Feature Elimination is a form of preprocessing that takes place after the transformation of the text to vectors. The text vectorizers used in the system result in a matrix, containing an importance score of a term in a certain document. After calculating the above scores, through vectorization, we can observe that a large set of terms, acquires a low importance score. It is a well-known practice that one can eliminate these terms from the summary data. The definition of the importance score threshold, makes a certain term useful or not. This elimination process will ultimately decide on the word set that will be used for movie modelling. In this recommender system the 10.000 most important words are chosen to model the summary, out of a 52.333 word vocabulary generated by the preprocessed summary corpus. The number 10.000 was chosen empirically, based on the observation of the TFIDF value substantial decrease, after the 10.000 most important terms.

3.2 Text Vectorization

Text Vectorization is the representation of text features, like words or sentences, by vectors (vector embeddings). This transformation of data helps machines, process and extract information from text. A primitive text vectorization technique is the bag-of-words (BoW) representation. A BoW vector consists of all the terms found in the corpus, with their respective frequency in each document of the corpus. This representation lacks the ability to distinguish between words with high frequency,

due to correlation between their meaning, and words with high frequency, due to their high frequent nature in the English language. Note that quite frequent English stopwords have already been filtered out, but other, less frequent words, especially found in the movies domain, can still impact our frequency vectorizer.

TFIDF solves the aforementioned BoW term relevance problem, by calculating not only the *Term Frequency (TF)*, but the *Inverse Document Frequency (IDF)* of a term, as well. The IDF component increases the importance of terms that entail document specific information, via their low document frequency, and punishes terms that are equally frequent among documents of the corpus. In order to transform our summaries from text to vectors using TFIDF, we need to establish our objective: extracting terms crucial to defining the genres of a movie. Each summary can be viewed, initially as a document, but this assumptions creates a problem. Viewing each summary as document, will force the IDF calculation to offset term importance, based on each movie summary. Not grouping summaries together will result in a vectorizer whose representation would be useful on distinguishing movie summaries apart, but would not be helpful on defining their common attributes.

The use of text vectorization aids in our goal of extracting genre-specific information from our summary corpus. In order to classify a movie on certain genres, we shall first discover what movies, belonging to the same genre, share in their summary. In order to force *Text Vectorizers* to produce term importance metrics, relative to movie genres, extracting crucial relations between the two, we should group movie summaries on a genre-wide summary text. For each genre within our dataset, we create an initially empty summary, termed as the *genre collective summary*. This summary vector has a size of 18, corresponding to the number of distinct genres on our dataset. Subsequently, for every movie m in the dataset, associated with a specific set of genres g, we add the movie's summary to the genre collective summary vectors, determined by the set g. The above mechanism creates 18 collective summaries, with each individual instance, including the summaries of all the movies that belong to its respective genre.

The updated set of documents has a size of 18, each document representing a movie genre. This corpus is used to fit the TFIDF vectorizer, created with the help of the *scikit-learn* [13] Python package. Fitting, essentially means that the vectorizer learns the vocabulary and computes IDF component, across terms of the document set. Note that, as stated above, we keep only the 10.000 most informative terms, based on their TF-IDF, to the final vector representation.

The *CLFD* text vectorization technique was adapted, in Python, with the information provided by the creators, in the publication of CLFD [10]. The use of CLFD was introduced on our work, as the entailed stop-word filter of this technique, can increase movie modelling performance. Similarly to TFIDF, in the CLFD approach we use the updated set of documents, which has a size of 18.

The simple Count Vectorization technique was used for the vector-based representation of summaries for the *Long Short-Term Memory (LSTM)* recurrent neural network, mentioned later in Section 3.3. The Count Vectorization technique is a useful way to feed our text data to the LSTM network, as it does not interfere with the text itself. The use of this simple vectorizer was introduced due to the technique's simplicity. Our LSTM neural network initially applies a Word Embeddings layer on the input. Thus, the use of the simplest text-to-vector technique was necessary, in order to lose

no information. The Word Embeddings layer of the LSTM, will provide our system with the necessary information regarding word importance, making the use of a more advanced vectorization technique unnecessary.

3.3 Classification

In this work, we use the summary of a movie to collect data and ultimately create its model. Our Classification stage uses the result of our Text Vectorization transformation techniques, which is a vector-based representation of the summary's text. The Classification stage is able to use the above information to categorise movies on their respective genres. Each movie can belong to a set of genres $g \in Genres$, where Genres is the dataset genre list stated in Section 2.6. An important note should be made, regarding the reported genres that the dataset provides for each movie, and the genres that are discovered through this classification stage. This classification stage aims to exploit the information found in the summary of the movie and not just rely on the genres that the producers believe that the movie belongs to. The task of classifying movies, using genres as labels, is a multi-label classification problem and this section analyses the techniques used to tackle it. In the end of the classification stage, each movie summary, is classified with 3 different techniques:

- Classifier Chain with TFIDF transformed summaries
- Classifier Chain with CLFD transformed summaries
- LSTM with Count Vectorizer transformed summaries

These techniques will produce 3 different sets of genre predictions for each movie called *TFIDF genres*, *CLFD genres* and *LSTM genres* respectively.

3.3.1 Classifier Chain

Our goal is to test various classification algorithms and different Classifier Chain orders, on the process of categorising movies. Initially, we will test different Classifier Chain orders, on the same classification method, and keep the order that outperforms the rest. This step will result in one Classifier Chain architecture, for every classification algorithm that will be tested. More specifically, for each classification method, we organise the $|L|$ classifiers in 30 random orders and pick the best performing order. The metric that will be used for finding the best Classifier Chain order, is the *Jaccard Similarity* between the genres reported in the dataset and the ones discovered by the classification stage. Jaccard Similarity is a metric to gauge the similarity, or diversity, between two sets A and B. The formal definition of Jaccard Similarity is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{3.1}$$

$$0 \leq J(A,B) \leq 1 \tag{3.2}$$

Higher *Jaccard Similarity* values represent similar sets while low values represent different sets. Although our ultimate goal is to discover a different genre set for every movie, we should emphasise that the goal of this step is to identify the best structure of our classification stage. This means that we should initially respect the dataset's reported genres and try to build a classifier structure that produces relative, but not

identical to the dataset, results. After deciding the best order of the $|L| = 18$ classifiers in the Classifier Chain, another decision regarding the classification algorithm should be made. Each Classifier Chain uses a certain classification algorithm from the ones listed in Section 3.3.2. The classification algorithm benchmark will be based on multiple metrics including *precision*, *recall*, *F1-score*, *Jaccard Similarity* and *Hamming Loss*. The results will be presented in Section 4.2.1.

More specifically, for each classification algorithm, we produce 30 Classifier Chains of 18 classifiers in each one. Each classifier, in every Classifier Chain, is associated with one label-genre and is positioned in a random position on the chain. Overall, 30 randomly ordered Classifier Chains, for every classification algorithm, are built. For each different classification algorithm, the Classifier Chain that reaches the maximum Jaccard Similarity between the genres reported in the dataset and the ones discovered by the classification stage, is the one that will represent the respective classification algorithm, on the final benchmark.

3.3.2 Classifiers Used

The classifiers that are tested in this work are the *Naive Bayes Classifier (NBC)*, the *Logistic Regression* with a cut-off threshold, the *Random Forest Classifier* and the *Long Short-Term Memory (LSTM)* recurrent neural network. These classifiers were chosen for their great performance on text data and their ability to be used in the Classifier Chain.

Naive Bayes Classifier	Logistic Regression	Random Forest	LSTM
TFIDF, CLFD	TFIDF, CLFD	TFIDF, CLFD	Count Vectorizer

TABLE 3.1: Vector representation of text used in each classifier

For each classifier a set of hyperparameters is defined. The NBC hyperparameters are set to $a = 1$ and $fit\_prior = True$. The fit_prior hyperparameter introduces the a priori probabilities of the labels. Knowing the data label's prior probabilities can further aid the classification process instead of assuming that all labels are equally probable. The Logistic Regression classifier is tested using the sag solver, commonly used in multi-label classification. This classifier is tested both without and with balanced class weight. Balanced class weights aim to counter the uneven distribution of labels on data. The Random Forest Classifier is used with the $class\_weights = balanced$ hyperparameter set, for the same aforementioned reason. The LSTM recurrent neural network is built using an initial Embedding Layer of size equal to our vocabulary size, followed by an LSTM layer and a dense output layer, of size 18 equal to the discrete number of genres, in the end.

For each data point, or movie summary vector, passed through the classification stage, a probability vector $\hat{y}$ is produced. This probability vector contains the probabilities $\hat{y}_i$ of each label i to correspond to the data point. The naive decision rule of labelling a data point with label i, if $\hat{y}_i > 0.5$ can be improved using *adaptive thresholds*. This technique will set a unique, to each label, decision threshold based on the Jaccard Similarity metric between $\hat{y}_i$ and y_i, with y being the ground truth label vector found on the dataset and i being a certain label. The use of adaptive thresholds aids classification performance by treating each label output uniquely. In our work we search the best threshold vector t with $t_i \in [0.1, 0.5]$ with 0.05 increments. The optimal adaptive thresholds are computed for every classifier structure built in

this work. This means that each Classifier Chain built in Section 3.3.1 is tuned with adaptive thresholds before their final comparison.

3.4 Movie Model

A core aspect of our recommender system is movie modelling. Each item-movie is modelled using a multivariate Gaussian distribution based on the work of *You Are What You Consume* [1]. Features-genres that describe the items are binary. This means that a movie can belong, or not, to a certain genre. The exact form of each Gaussian distribution depends on the general rating of the movie, found in the dataset, as well as the features or genres that describe it. Note that the general rating of a movie is the consensus rating, averaged across all the reviews found in the MovieLens dataset.

More specifically, each multivariate Gaussian distribution has a mean vector of size k, where k is the number of features (dimensions) of the multivariate Gaussian; and with every element of the mean vector being equal to the consensus rating of the movie in question. This means that every Gaussian distribution has the same mean, disregarding if a feature describes the item or not. Information, regarding the features of our items, are entailed in the correlation matrix. In order to define the correlation matrix, we will use the features extracted from the classification stage and the features found in the dataset. The genres that are found through the classification stage depend heavily on the summary of the movie, while the genres found in the dataset are the ones that are reported by the movie producers. The correlation matrix, constructed as a diagonal matrix, is comprised of elements equal to:

- 1, if the genre is discovered through the classification stage and is also found in the dataset
- 10, if the genre is discovered through the classification stage but is not found in the dataset
- 20, if the genre is not discovered neither through the classification stage nor found in the dataset

The values for the correlation matrix were chosen empirically, based on the set of choices made in the work of *You Are What You Consume* [1]. The above mechanism models each movie's feature as a Gaussian distribution with an uncertainty level, relative to our belief regarding the features of the movie. If we can confidently identify a genre that truly characterises a movie, during the summary classification stage, and confirm our hypothesis using information from the initial dataset, we model the corresponding Gaussian distribution with low variance. Conversely, as our understanding indicates that a particular genre does not accurately characterise the movie, we subsequently raise the variance of the corresponding Gaussian distribution.

As stated above, our classification stage produces 3 different sets of movie genres, based on the 3 different techniques of text handling.

3.5 User Model

3.5.1 Bayesian User

In order to model the user and his/her preferences, we utilise the Bayesian user model, proposed by *You Are What You Consume* [1]. Each user has a preference vector, similar to the movie's feature vector, consisting of 18 features. We model the feature vector as a multivariate Gaussian distribution with unknown mean vector and covariance matrix. In order to infer the mean vector and covariance matrix, with the goal of making a recommendation, we use the Normal Inverse Wishart (NIW) distribution, as it is a conjugate prior of a multivariate Gaussian distribution with unknown mean vector and covariance matrix.

In order to model the user, we initially start with uninformative NIW, and as we observe new data, through user feedback, we update the NIW components with Bayesian Updating. When we need to construct the user model, we use the updated NIW components to infer the multivariate Gaussian distribution's mean vector and covariance matrix.

3.5.2 Normal Inverse Wishart

The Normal Inverse Wishart (NIW) distribution is a conjugate prior of the multivariate Gaussian Distribution. It is modelled by 4 components $\boldsymbol{\mu_n}$, k_n, v_n, $\boldsymbol{\Psi_n}$.

When the recommender system assembles the model of a new user, we have no prior knowledge about their preferences. An uniformed NIW is constructed that holds no information about the user's features. In order to construct an uninformed NIW we set:

$$\boldsymbol{\mu_0} = \left[\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right], \boldsymbol{\mu_0} \in \mathbb{R}^n$$
$$\kappa_0 = 0$$
$$v_0 = -1$$
$$|\boldsymbol{\Psi_0}| = 0, \boldsymbol{\Psi_0} \in \mathbb{R}^{nxn}$$

When the fresh user watches a movie and gives feedback to our recommender system, their model should be updated. The multivariate Gaussian model of the user is updated efficiently, through the NIW hyperparameters, as described in Bayesian Learning (2.4.2). On observation of new user feedback we perform Bayesian updating on the NIW hyperparameters with the mechanism from [1], stated in equation set 2.7. After the updating process, the NIW hyperparameters become:

$$\begin{aligned} \boldsymbol{\mu_n} &= \boldsymbol{x} \\ \kappa_n &= n \\ v_n &= n - 1 \\ \boldsymbol{\Lambda_n} &= \boldsymbol{S} = \boldsymbol{\Sigma_n} \end{aligned} \tag{3.3}$$

3.5.3 Niche User Index

Another feature that should be evaluated, regarding our user modelling method, is the *Niche User Index (NUI)*. The Niche User Index is used to assess a user's inclination towards popular movies. Various people feel more inclined to popular movies than

others. This notion lead to the decision of distinguishing the way that our recommender system processes niche and mainstream users. Using the Niche User Index metric we can distinguish between mainstream and niche users. Niche users are the ones that tend to watch/rate movies that are not popular, whereas mainstream users are the ones that tend to rate movies that are popular. In order to test our recommender system explicitly on niche or mainstream users, we shall initially define them.

In order to compute the Niche User Index, we shall first compute the *Cumulative User Vote Count - CUVC*. This metric uses the *vote_count* quantity that our dataset provides for each movie. The *vote_count* is the number of votes, or ratings, that a movie has received on the MovieLens website and therefore, is a measure of the movie's popularity index. The *CUVC* for a certain user is:

$$CUVC_i = \sum_{m \in Movies_i} vote\_count_m \tag{3.4}$$

where $Movies_i$ is the set of movies that user i has rated and $vote\_count_m$ is the *vote_count* quantity for movie m from the dataset.

The *Niche User Index* of user i is computed as the normalised form of $CUVC_i$:

$$NUI_i = \frac{CUVC_i - \min\limits_{k \in Users} (CUVC_k)}{\max\limits_{k \in Users} (CUVC_k) - \min\limits_{k \in Users} (CUVC_k)} \tag{3.5}$$

where *Users* is the set of the system's users. As a normalised value, for a certain user i, $NUI_i \in [0, 1]$. A high NUI value indicates a user that enjoys popular movies, while a low value indicates a user with a niche movie taste.

3.6 Recommendation Process

The recommendation process of our system uses the movie and user models stated above. As we previously discussed, the items and the users of our recommender system, are modelled using multivariate Gaussian distributions. This common probabilistic modelling technique, gives us the ability to use metrics based on probability divergence, as a way to indicate how similar the two models are.

The metric that is used on our work is the *Kullback-Leibler divergence*. The Kullback-Leibler divergence of two multivariate Gaussian distributions, $\mathcal{N}_0$ and $\mathcal{N}_1$, with common size k, is defined as:

$$D_{\mathrm{KL}}(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) - k + (\mu_1 - \mu_0)^{\mathsf{T}} \Sigma_1^{-1} (\mu_1 - \mu_0) + \ln\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

where μ_0 and μ_1 are their respective mean vectors and Σ_0 and Σ_1 are their respective covariance matrices. The result of the above equation is the divergence of the two distributions. The Kullback-Leibler divergence metric is only used comparably in our study. More specifically, we are interested in finding the movie models that reach the minimum divergence to the user model. Using Kullback-Leibler divergence we define a *predicted movie rating*, by normalising the divergence value on $[0.5, 5]$, which is our dataset's movie rating scale.

Our movie modelling technique takes into account the genres that the movie belongs to, as well as the consensus rating of the movie. It is known that audience like to watch popular movies even if their genre preferences do not always align with the movie's genres. Taking the above observation into account, we simultaneously employ a popularity-based technique on our recommender system. Our recommendation technique is based on recommending movies that correspond to the user's preferences, but take into account the user's inclination towards popular movies, as well.

In order to define a user's inclination towards popular movies we define *Popular Movie Inclination-PMI*. The PMI value maps the user's tendency to like popular movies, or more specifically their common characteristics, thus being a valuable indicator for our recommendation stage [1]. In order to define PMI we initially identify the movies that the user has rated with a rating ≥ 4, as well as their respective *vote_count* found on the dataset, called the *enjoyed_movies* set. Note that, as mentioned above, *vote_count* is movie popularity index present on our dataset. On the selected *enjoyed_movies* set, of user i, we define PMI as the normalised mean of *vote_count* on the *enjoyed_movies* set:

$$\mu_{vote\_count,i} = \frac{\sum_{m \in enjoyed\_movies_i} vote\_count_m}{|enjoyed\_movies_i|}$$

$$PMI_i = \frac{\mu_{vote\_count,i}}{max\_PMI}$$

where max_PMI, is the maximum PMI user value, found empirically on our user set.

To recommend a set of movies to the user, we employ a multi-winner election mechanism depicted on Figure 3.2. The candidate movies are the ones that have a *predicted movie rating* ≥ 4.5. This mechanism consists of two voter sets, *KL Voters* and *Popularity Voters*. Each voter has a unique preference vector and votes using *Borda Count*. More specifically, if the preference vector of a voter consists of n movies, the votes that a candidate receives is relative to the position of the candidate in the voter's preference vector. The most preferred movie receives n votes, while the least preferred movie receives 0 votes. Each voter's preference vector is build stochastically. More specifically, the movies that comprise the voter's preference vector are arranged based on probabilities. Regarding KL Voters, the probability with which a movie receives a better position in the preference vector is based on the movie's divergence score. On popularity Voters, the probability with which a movie receives a better position in the preference vector is based on the movie's popularity score.

The *KL Voters* voter set, consisting of 1000 voters, has a preference vector based on the Kullback-Leibler divergence between the user's and the movie's model. The *Popularity Voters* set has a number of voters that is relative to the user's inclination towards popular movies. The number of voters, n, in this set is calculated as:

$$n = 1000 * PMI$$

[1] Note that PMI is different from *Niche User Index (NUI)*, as the latter calculates the user's tendency to watch popular movies, but not necessarily like them.

Based on movie popularity, these voters have a preference vector that favours popular movies over less popular ones, arranged in the same probabilistic manner. Note that the preference vector of these voters consists of candidate movies that have a predicted rating score ≥ 4.5, based solely on Kullback-Leibler divergence.

After the completing 1000 voting rounds we average the results and build the final movie recommendation set, consisting of the 20 most voted movies.

For example, lets consider user u, with a PMI score of 0.4. Our voting mechanism will use 1000 KL Voters and 400 Popularity Voters. After comparing the model of u with our system's movie models, we gather the results on the *Predicted Score* row of Table 3.2. For each movie that has a predicted score ≥ 4.5, we depict its popularity $\in [0, 1]$ on the *Popularity Score* row. In our example we found 207 movies with a predicted score ≥ 4.5. Regarding KL Voters, each voter builds their preference vector based on the predicted score stochastically. The predicted score defines the likelihood of each item being chosen during the random sampling process. More specifically, each item is placed higher in the preference ranking, by being selected first in the random choice experiment with no replace, with a probability relative to its predicted score. This means that an item with a predicted score of 4.8 is more probable to receive a higher ranking than an item with a predicted score of 4.6 and less probable to receive a higher ranking than an item with a predicted score of 4.9. For example, a possible preference vector of a KL Voter is depicted on Table's 3.3 *Movie Title* row, with decreasing preference order.

Movie Title	C	A	...	G	F	...
Predicted Score	4.9	4.8	...	4.5	4.4	...
Popularity Score	0.8	0.2	...	0.7		

TABLE 3.2: Movie predicted scores for user u and their respective popularity scores, in predicted score decreasing order.

Movie Title	C	G	...	A
Votes	206	205	...	0

TABLE 3.3: Example preference vector with decreasing preference order of a KL Voter with its respective Borda Count votes.

Movie Title	A	C	...	G
Votes	206	205	...	0

TABLE 3.4: Example preference vector with decreasing preference order of a Popularity Voter with its respective Borda Count votes.

Movie Title	C	G	...	A	...
Total Votes	275003	190132	...	70007	...

TABLE 3.5: Voting round results of 1400 voters (1000 KL Voters and 400 Popularity Voters).

Regarding Popularity Voters, each voter builds their preference vector based on movie popularity stochastically. Similarly to KL Voters, now the movie's popularity score defines the likelihood of each item being chosen during the random sampling process. More specifically, each item is placed higher in the preference ranking, by being selected first in the random choice experiment with no replace, with a probability relative to its popularity score. Voter is depicted on Table's 3.4 *Movie Title* row, with decreasing preference order. Following the Borda Count voting rule, the most preferred movie on each voter's preference vector receives 206 votes, the next most preferred 205 e.t.c. The votes that each movie receives in the previous two preference vector examples can be seen on the *Votes* row of the respective tables. Table 3.5 depicts a possible result of the voting process, in a decreasing number of votes order, after we sum up the votes of all the 1400 voters. Afterwards, we rerun the above voting round until we reach 1000 simulation voting rounds and average out the final voting results. The recommendation set consists of the 20 most voted movies.

The above probabilistic voting mechanism is used to tackle the problem arising from finding a large number of movies that the user might like. Our experiments showed that it is possible to find a large set of movies that are in line with the user's preferences. Our probabilistic voting approach is able to pick a movie recommendation set, without undermining any of the nearly equally capable candidates that receive similar predicted ratings.

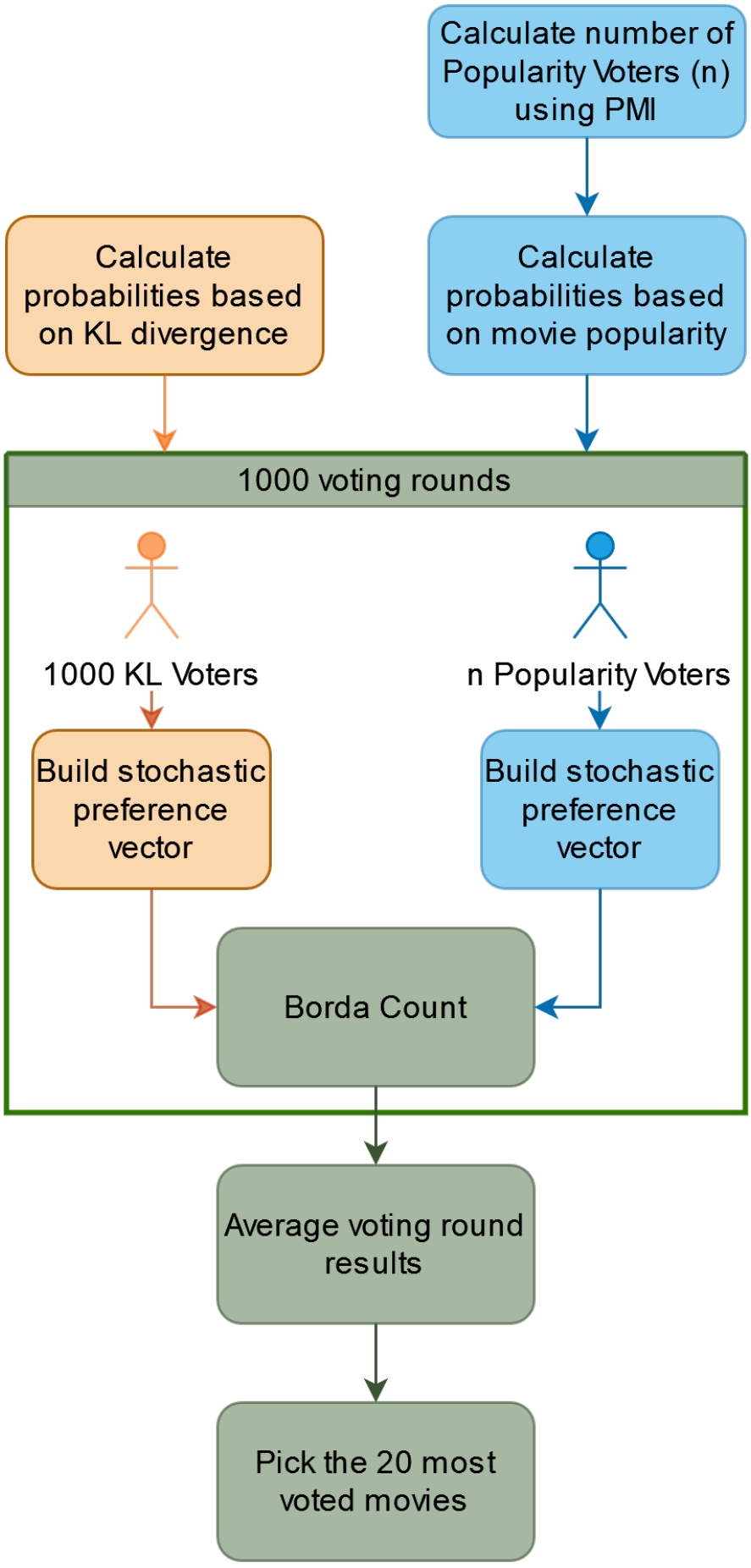


FIGURE 3.2: Outline of the Election Mechanism.

Chapter 4

Experimental Evaluation

This chapter includes the results of the experiments conducted in various development phases of this recommender system. A number of experiments were carried out to understand the structure of the dataset contents, with the ultimate goal of making a decision on the parts of the dataset that would be useful to our research. Furthermore, experiments were conducted on the movie classification phase, in order to verify that the summary is indeed a useful asset in movie categorisation. These experiments were used to decide on the classifier that would ultimately be used on our recommender system. Finally, using our complete recommendation approach, a series of experiments were conducted to verify the quality, as well as robustness, of the recommendation process and our recommender system in general.

4.1 Dataset Exploration

Our dataset includes 26 million ratings of 283110 unique real users, collected anonymously through the MovieLens website. Each user has rated an arbitrary number of movies, with a rating from 0.5 to 5. In order to understand the number of ratings that most of the users have provided, Figure 4.1 was plotted. This figure shows that most of the users in our dataset have rated fewer than 2000 movies, with number of ratings near 100 being quite common. Higher review quantities seem quite sparse, so their respective users should not be used for testing. The evaluation process of our recommender system will split the ratings of a user in a test set and a training set. The test set will later be used to evaluate the quality of our recommendations (4.4). In order to have a sufficient number of movies and their respective user ratings in the test set, we should select users that have rated an adequate number of movies. In our case, we found that users with a number of ratings between 950 and 1050, give us an adequate number of movies in the test set to evaluate the quality of our recommendation. The number of selected users, after applying the aforementioned filter, is 618. Moreover, users in this set show a not highly non-uniform distribution regarding the number of movies they have rated, depicted on Figure 4.2.

4.1.1 Niche Users

In order to obtain further information about the popularity of the movies, that the dataset's users have rated, Graph 4.3a was plotted containing the number of users that correspond to certain intervals of Niche User Index (3.5.3). Note that the Niche User Index is *normalised* with respect to the user set in both diagrams. We can indeed

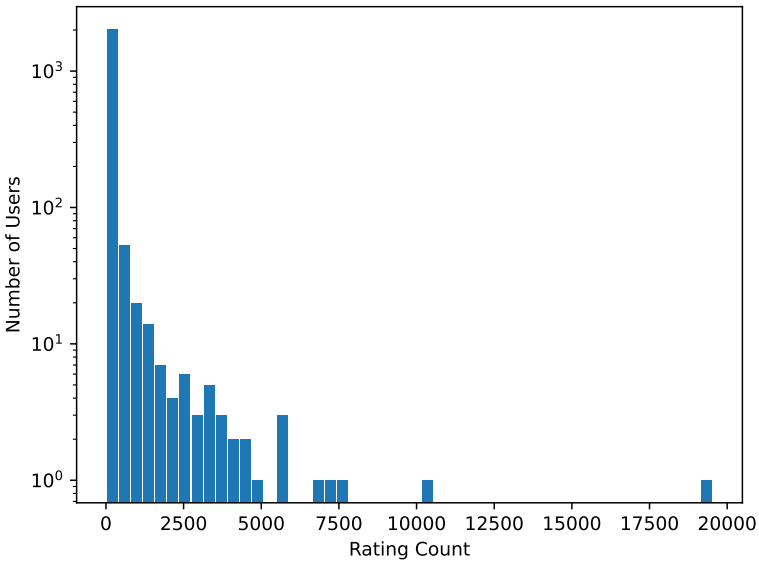


FIGURE 4.1: Number of users across rating count intervals.

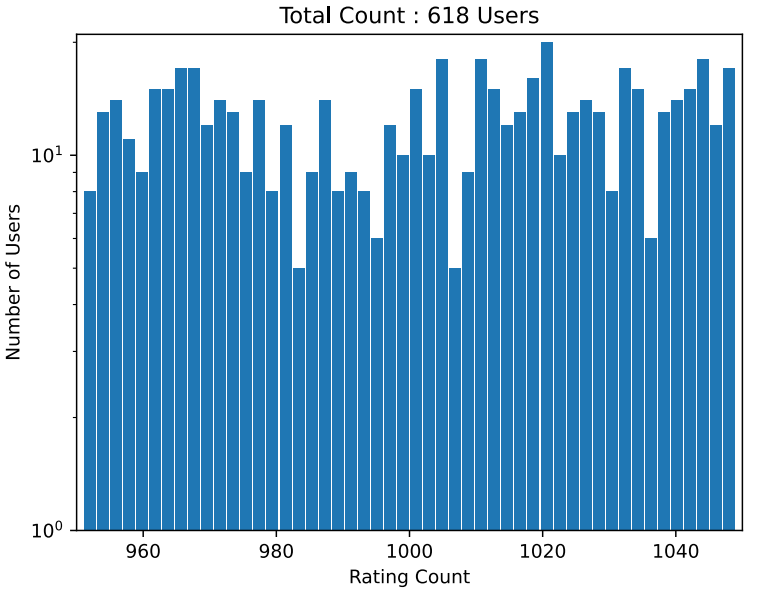
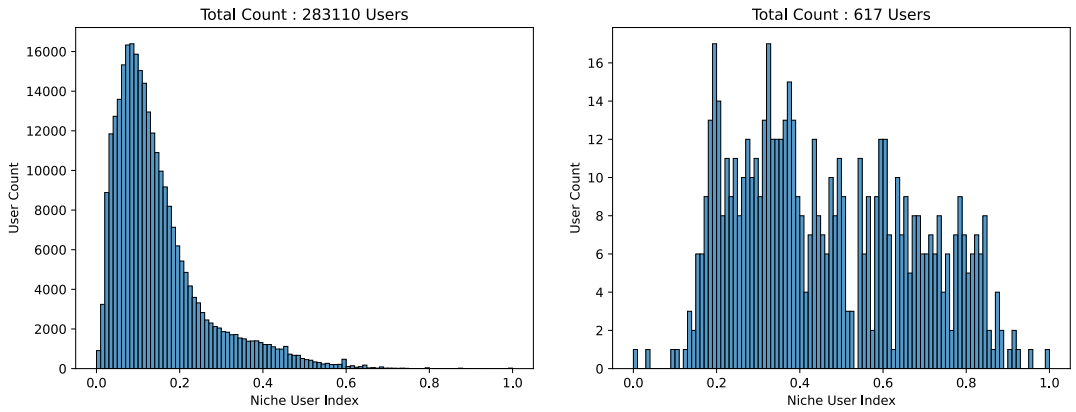


FIGURE 4.2: Number of ratings across selected users.

see that users have an arbitrary inclination towards watching and thus rating, popular movies. Moreover, we can deduct that most users have Niche User Index close to 0.1. A better understanding regarding the users that our test set would be comprised of, can be obtained through Figure 4.3b. This figure shows the number of users that correspond to a certain interval of Niche User Index, regarding only users who have rated 950 to 1050 movies. We can see a uniform distribution of Niche User Index among our selected users. Note that, in this case, Niche User Index is normalised, with respect to the filtered users set.

Taking into consideration Figure 4.3b, we see that our set of selected users can be



(A) All 283110 users depicted (B) Only the 618 selected users depicted

FIGURE 4.3: Number of users with certain Niche User Index.

rather easily split, due to the distribution that describes it. We define a user as Niche, if the user has a Niche User Index belonging to the bottom 10% of the set. A further understanding regarding the Niche user set can be obtained through Figure 4.4. As shown, we choose the Niche user set to be comprised of users with $Niche\_User\_Index \in (0, 0.2)$. The total number of niche users, using the 10% rule stated above, is 61. Therefore, we conclude that we have an adequate number of Niche Users to conduct our experiments.

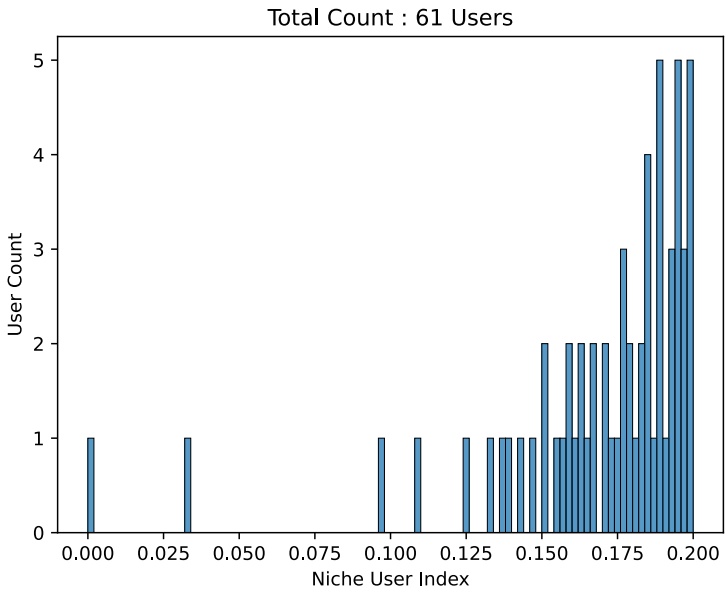


FIGURE 4.4: Niche User Index of Niche users.

4.2 Classification

The classification stage of our approach aims to categorise movies into genres, using movie summaries. The dataset that we use, includes genres that the movie's producers have reported. Our aim is to tune our classification stage in order to discover

hidden information, through movie summaries, and create a new genre set for each movie, solely based on its summary.

The experiments on this section start by evaluating the classification methods specified in Section 3.3.2. This assessment involves a comparison using various metrics across different text vectorization methods. The initial step is to determine the most effective classification method for each text vectorization approach. Following this analysis, we will then proceed to select the best combination of text vectorization method and classification algorithm based on the obtained results.

4.2.1 Metrics Used

This section includes information about the metrics that are used to tune our classification stage. The target stated above implies that our classification stage should not always produce results that are identical to the dataset. Our target is to produce results similar to the dataset, while keeping a generalised approach.

Precision, *Recall* and *F1-Score* are widely used classification metrics, that carry valuable information regarding classification efficiency. The formal definition of these metrics is:

$$Precision = \frac{True\_Positive}{True\_Positive + False\_Positive} \tag{4.1}$$

$$Recall = \frac{True\_Positive}{True\_Positive + False\_Negative} \tag{4.2}$$

$$F1\ Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

The definition of the above metrics can be aggregated to fit our multi-label classification problem, but for this work's testing purposes, precision, recall and F1-Score will be computed for each unique label or genre. Calculating these metrics for each unique genre provides valuable insights into the performance of our classifier across different genres. More specifically, the aforementioned comparisons will be conducted on the classification architectures and input data, depicted on Table 4.1.

Naive Bayes Classifier	Logistic Regression	Random Forest	LSTM
TFIDF, CLFD	TFIDF, CLFD	TFIDF, CLFD	Count Vectorizer

TABLE 4.1: Vector representation of text used in each classifier

Another metric that will be used is the Jaccard similarity between the dataset's reported genres and the output of the classification stage. *Jaccard similarity* or *Jaccard Index*, $J(A, B)$, between two sets A and B is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.4}$$

High Jaccard similarity coefficient indicates similar sets, while low values indicate diverse sets. In our multi-label classification problem we compute the *weighted* and *sampled* Jaccard similarity. The sampled Jaccard similarity represents the average similarity across all tested samples (movies). While this sampled metric provides an overall assessment of the classification mechanism's performance, it lacks information on performance, specific to individual labels. On the other hand, the weighted

Jaccard similarity is calculated as the weighted mean across all labels. This approach considers all positive instances for each label, comparing them across sets, in proportion to the number of samples for each label. Therefore, the weighted Jaccard similarity offers insights on the label-specific performance of the classification mechanism.

The third metric used to test our classification mechanism is the average Hamming loss. *Hamming loss* in the multi-label classification domain, corresponds to the *Hamming distance* between two sets y and $\hat{y}$. Hamming Distance between two label sets is the number of labels that are not matched between the sets. While Hamming Distance is the absolute number of unmatched labels, Hamming loss is the fraction of the labels that are unmatched between the two sets. Thus, smaller values indicate that the two sets are similar.

The last indicator of our classification stage performance, are normalised *confusion matrices*. Confusion matrices depict the True, False Positives and Negatives, regarding a certain label. These matrices can help us study the performance of our classification stage regarging a certain label. Moreover, a *Multi-label Confusion Matrix - MLCM* [7] is built to access the overall classification performance. MLCM is an *aggregated* confusion matrix whose interpretation carries valuable information regarding True, False Positives and Negatives. For a row k, that represents data for class C_k, the cell corresponding to the main diagonal of the matrix, shows to the True Positives for class C_k, while the other cells represent the False Negatives. For a column k, the cell corresponding to the main diagonal, shows the True Positives, while the other cells show the False Positives for class C_k

4.2.2 Results using CLFD

The results obtained via the CLFD vectorization technique will be presented first. Figure 4.5 depicts the distribution of the above metrics across different classifiers. This figure shows the variance of precision, recall and F1-Score across the different classification methods. Our results will be examined based on the median value and the interquartile range of the metrics. Our objective is to identify genres that are not explicitly listed in the dataset, but effectively characterise a movie. Therefore, our experiment comparison will emphasise more on recall and slightly less on precision. The classification method that will ultimately be chosen, should be able to predict the dataset genres, to the extend of trust on movie producers, while discovering new ones as well.

In the aforementioned figure we can see that the Naive Bayes classifier has the best median precision value followed by the Random Forest classifier, the Unbalanced Logistic Regression, and the balanced Logistic Regression, in respective order. The Unbalanced Logistic Regression reaches a max precision score of 0.85, followed by the Naive Bayes classifier on 0.75. The balanced Logistic Regression and the Random Forest Classifier follow with a max precision value near 0.65. Another observation that should be made is that the balanced Logistic Regression has a minimum precision value near 0.05, quite lower compared to the other classification methods.

The balanced Logistic Regression produces the highest median recall value, near 0.6, followed by the Random Forest Classifier and the Unbalanced Logistic Regression, near 0.55. The Naive Bayes Classifier has a median recall value near 0.3, which excludes this classification algorithm from the comparison. The remaining classifiers

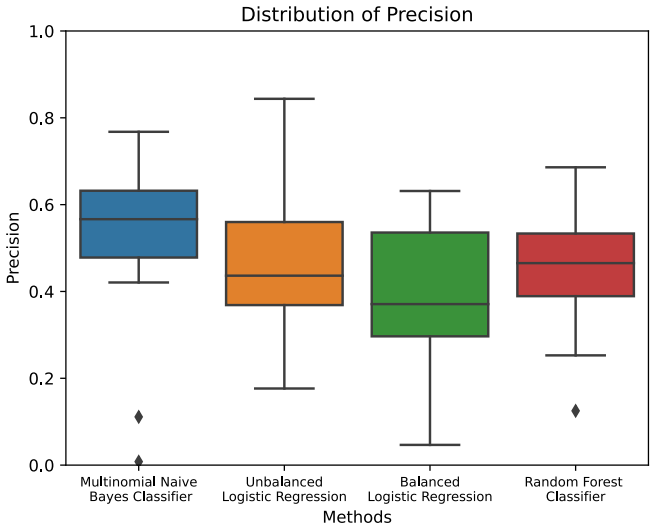
achieve a similar max recall score of 0.8, with the balanced Logistic Regression having the tighter interquartile range, followed by the Unbalanced Logistic Regression and the Random Forest Classifier.

The F1-Score is a balanced aggregation of the above two metrics and consequently is able to depict the general effectiveness of our classification methods. We observe that Unbalanced Logistic Regression, balanced Logistic Regression and Random Forest Classifier reach similar F1-Scores in their interquartile range, with Unbalanced Logistic Regression performing slightly better. The Naive Bayes Classifier, due to low recall score, is outperformed by the other methods. We should note that although the Random Forest classifier has the tightest interquartile range, an outlier exists near 0.05.

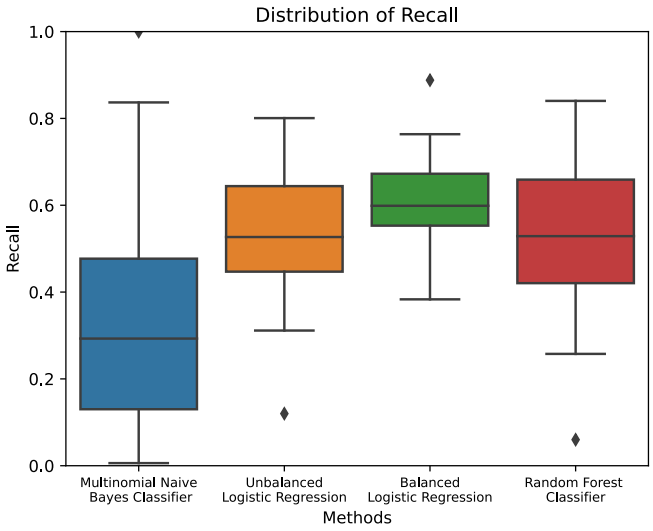
Table 4.5 includes the metrics Hamming loss, weighted Jaccard similarity and sampled Jaccard similarity. We can observe that the Unbalanced Logistic Regression and Random Forest classifier outperform the other two classification methods, on all three metrics. The Unbalanced Logistic Regression performs slightly better than the Random Forest classifiers achieving a Hamming Loss of 10%, a weighted Jaccard similarity near 0.4 and a sampled Jaccard similarity of 0.45. We can therefore deduct, through the Hamming loss metric that we misreport, relative to the dataset, only 10% of the labels. The low Hamming loss metric is an indicator that we approach our goal of discovering "new" movie genres, based solely on the movie's summary, that are not reported on the dataset but do, in fact, characterise the movie. By taking into consideration the above experiments and their outcomes, we deduct that the best classification method to pair with the CLFD text vectorization is the Unbalanced Logistic Regression.

After applying our Unbalanced Logistic regression classification method of choice to our data, we can observe the normalised confusion matrices. Confusion matrices are used as a quality assessment tool regarding CLFD transformed data and will later be compared among other text transformation methods. We will initially examine the individual confusion matrices for each label, which are depicted on Figure 4.6. We can observe that the True Negative score, amongst all labels, is quite high, near 0.95% for most labels. This observation demonstrates that our selected classification method does not attach more labels to a data point, than it should. An exception to the above result are the comedy and drama labels, with a True Negative score near 80%. These genres are quite popular, in the sense that in reality, they characterise a vast number of movies, while not being easily picked by movie producers, due to their mainstream nature. By classifying movies, based solely on movie summaries, we can see that, although some of them are not labelled by these specific genres in the dataset, they can indeed be characterised by them. The above reason leads to a decline in True Negative score and an increase on False Positive score, indicating that we succeeded in discovering new genres, even if they lie in the generally accepted mainstream genre domain. Regarding the True Positives, we observe that most labels reach a normalised score near 0.85, with the Film-Noir label being the outlier, with a score near 0.75. Film-Noir labelled movies are quite sparse in our dataset and little information can be obtained through the vectorizers and the classifier. The True Positive score of Film-Noir, although low, is acceptable.

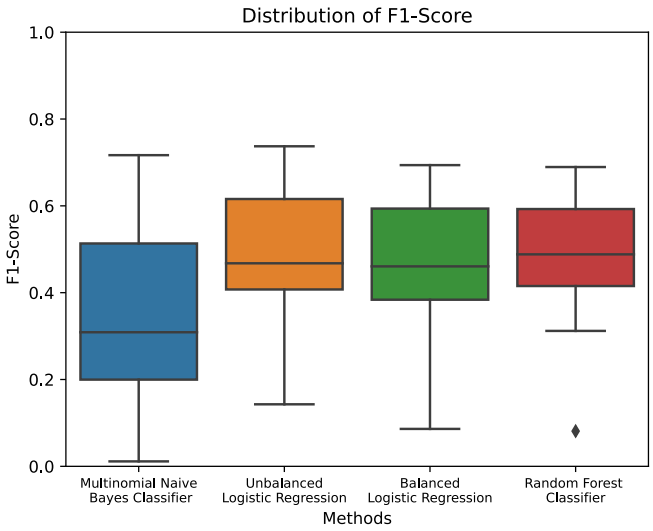
The aggregated MLCM confusion matrix is depicted on Figure 4.7 and can summarise the above results, as well as show the label-specific correlation that appeared in the results. By examining the Multi-Label Confusion Matrix we observe, in the



(A) Precision



(B) Recall



(C) F1-Score

FIGURE 4.5: Distribution of Precision, Recall, and F1-Score across different classifiers using CLFD vectorization

Movie Title	Reported Genres	CLFD Genre Discoveries
Star Wars	Action, Adventure, Scifi	Animation, Comedy, Musical
Goodfellas	Crime, Drama	-
Titanic	Romance, Drama	-
Psycho	Crime, Horror, Thriller	Mystery
No Country For Old Men	Crime, Drama	~~Drama~~, Action, Thriller, Western
Memento	Mystery, Thriller	Horror

TABLE 4.2: Genre Discoveries using the Unbalanced Logistic Regression on CLFD transformed data

form of (True Label-Predicted Label), the following classification mistakes, or genre discoveries:

1. Action-War
2. Animation-Comedy and Adventure
3. Children-Comedy
4. Comedy-Drama
5. Film Noir-Thriller
6. Horror and Thriller
7. Musical-Drama
8. Mystery-Thriller and Drama
9. Science Fiction-Thriller
10. Western-Adventure

It is known that War movies, do in fact contain Action movie parts, due to their nature. The same is true for Children and Comedy. On the above set we can identify meaningful genre correlations on most of our observations, except 5, 7 and 9. A series of example movies and their respective reported and discovered genres can be seen on Table 4.2. On the genre discovery column, we include the new non-reported, discovered genres, while the reported genres that are not labelled through our classification method, are crossed out.

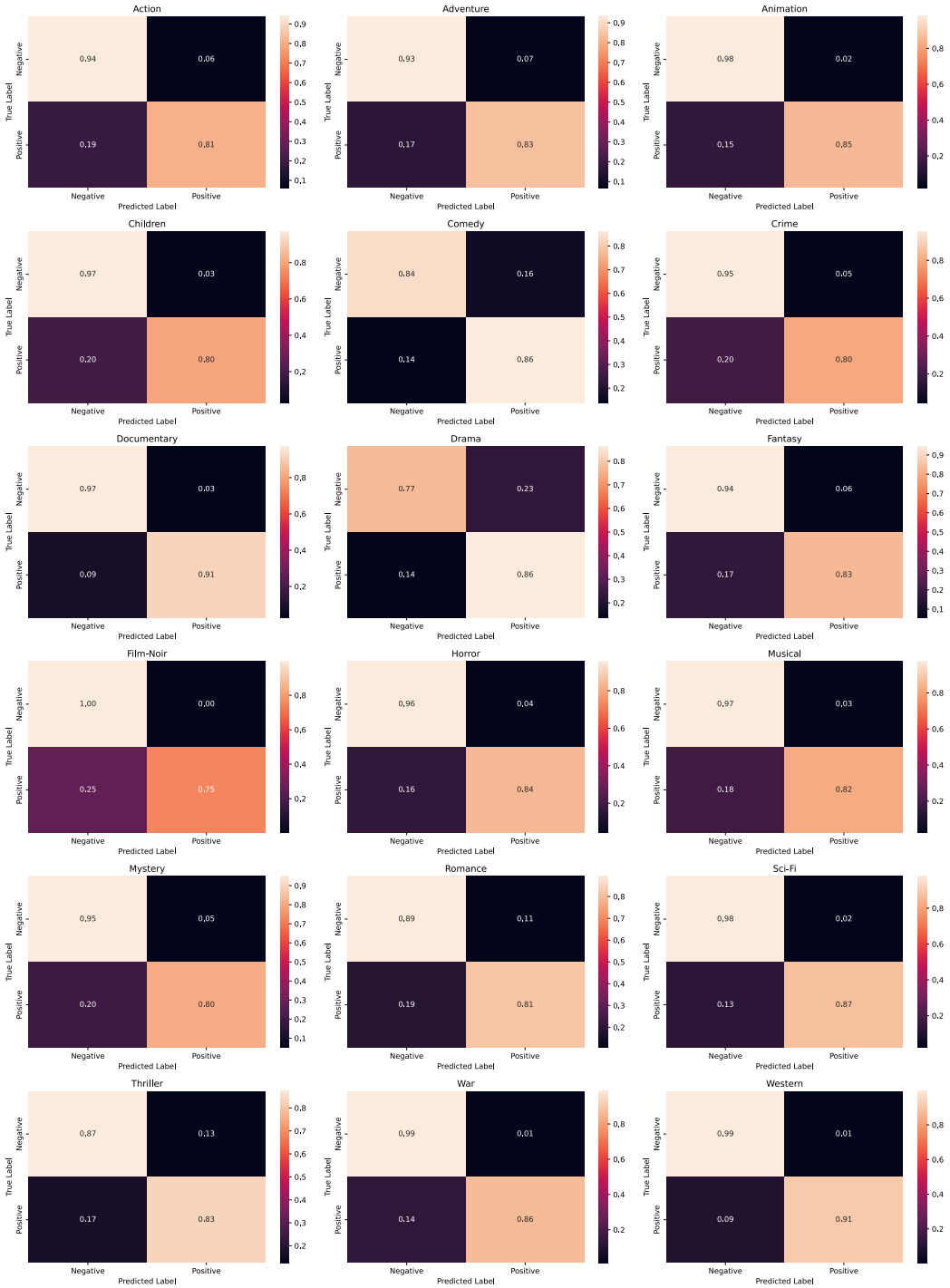


FIGURE 4.6: Confusion Matrices for each label using Unbalanced Logistic Regression on CLFD transformed data.

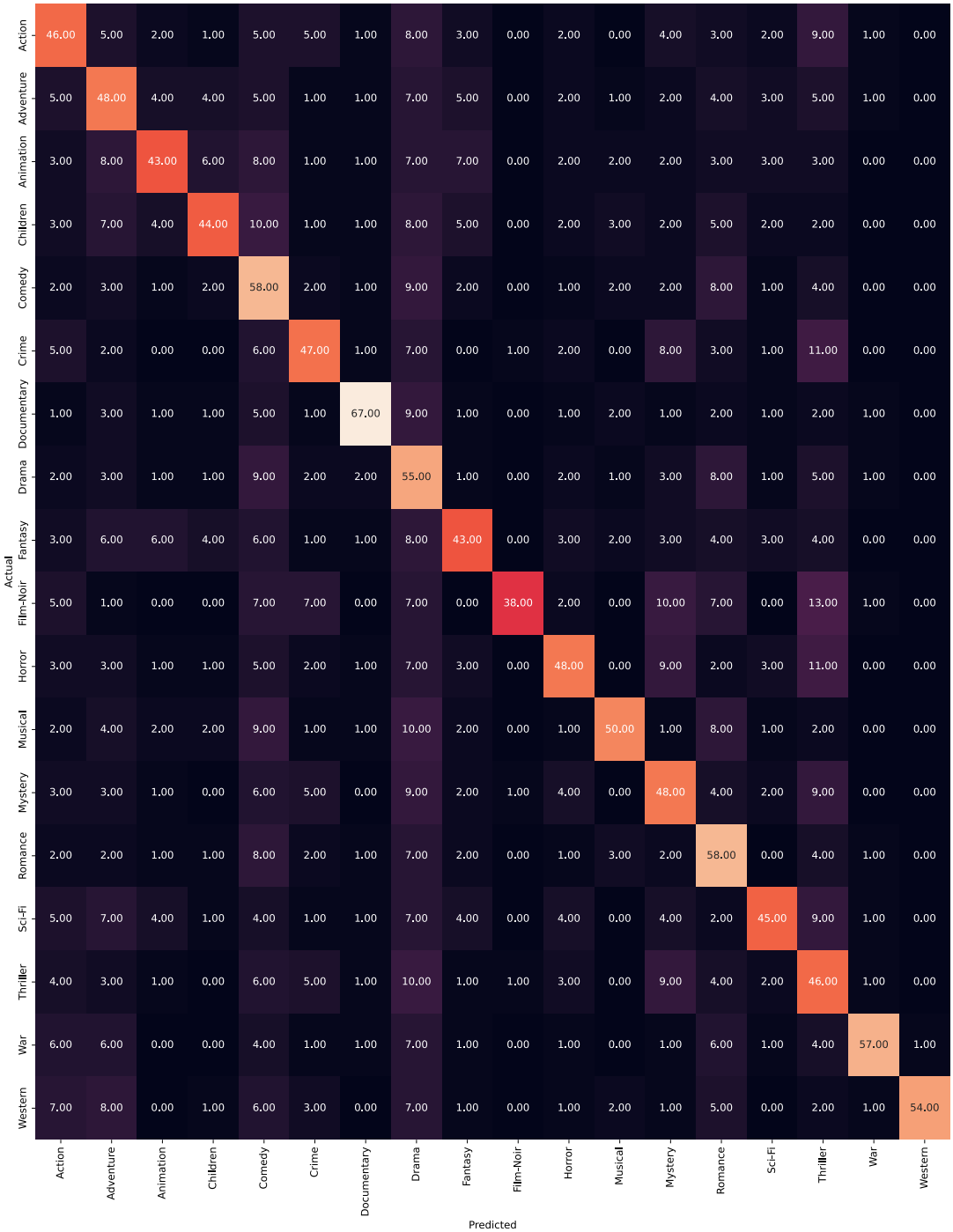


FIGURE 4.7: Multi-Label Confusion Matrix using Unbalanced Logistic Regression on CLFD transformed data.

4.2.3 Results using TFIDF

This section includes the results obtained through the TFIDF vectorization technique. Figure 4.8 depicts the *distribution* of the precision, recall and F1-Score across different classification methods.

The Naive Bayes classifier and the Unbalanced Logistic Regression reach a median precision near 0.45, followed by the Random Forest classifier near 0.4 and the balanced Logistic Regression on 0.3. The Unbalanced Logistic Regression reaches the highest max precision value on 0.85 with Naive Bayes classifier and Random Forest classifier following on 0.8 and 0.75, respectively. The balanced Logistic Regression under-performs significantly. We should also specify that the other three methods produce a similar interquartile range.

By examining the recall metric, we observe that the balanced Logistic Regression reached the highest median recall value of 0.7, followed by the other three methods near 0.55. The Naive Bayes classifier reached a max recall value of 1 and a min recall value near 0. This reason excludes the Naive Bayes classifier on the TFIDF domain as well. We should note that the other three methods reach a max recall value of 0.8, but the Unbalanced Logistic Regression and the Random Forest classifier reach a min recall value near 0.1.

The highest median F1-Score, of 0.5, is achieved by the Unbalanced Logistic Regression. The Random Forest classifier produces a slightly smaller median F1-Score, followed by the Naive Bayes classifier and the balanced Logistic Regression near 0.45. As we excluded the Naive Bayes classifier from this comparison, we will now focus on the three remaining methods. The tightest interquartile range as well as the maximum F1-Score, is achieved by the Unbalanced Logistic Regression.

Table 4.5 includes the metrics Hamming loss, weighted Jaccard similarity and sampled Jaccard similarity for the methods stated above. We can see that Unbalanced Logistic Regression and Random Forest classifier reach similar values on all the metrics, while the other two methods constantly under-perform. The Random Forest classifier performs, constantly, slightly better than the Unbalanced Logistic Regression and thus will be chosen as the best classification method for TFIDF transformed input. We should note that the results of the Random Forest classifier using TFIDF transformed data are outperformed by the Unbalanced Logistic Regression using CLFD transformed data.

After classifying each movie found on the dataset, using its TFIDF transformed summary, we can observe the final confusion matrices. Figure 4.9 shows the confusion matrix for each label. We observe similar, to the CLFD transformed data, performance on True Negatives, with the Drama label being chosen more precisely. More precisely the True Negative score is near 97%, for every label except drama, which reaches 80%. This is an indicator that this approach is more accurate that the previous one, producing results that are closer to the dataset format. Regarding the True Positives, we reach a mean score of 84%, with film-noir and fantasy labels not being so accurately predicted and scoring 77% and 70%, respectively. Regarding film-noir labelled movies we observe that, although not much data is present, this approach performs better than the Unbalanced Logistic Regression with CLFD.

Figure 4.10 shows the aggregated normalised Multi-label confusion matrix. By examining the matrix, an initial observation regarding less confused genres can be

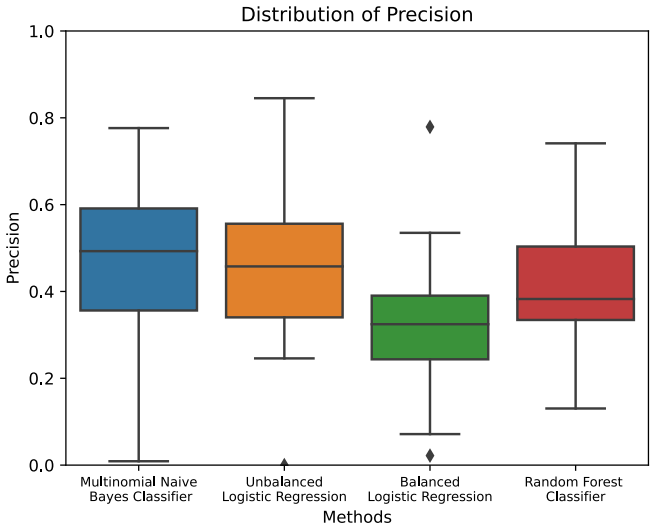
Movie Title	Reported Genres	TFIDF Genre Discoveries
Star Wars	Action, Adventure, Scifi	Animation, Fantasy
Goodfellas	Crime, Drama	Comedy
Titanic	Romance, Drama	-
Psycho	Crime, Horror, Thriller	~~Crime~~, Drama, Mystery
No Country For Old Men	Crime, Drama	~~Drama~~, Action, Horror, Thriller
Memento	Mystery, Thriller	Drama, Horror

TABLE 4.3: Genre Discoveries using the Random Forest Classifier on TFIDF transformed data

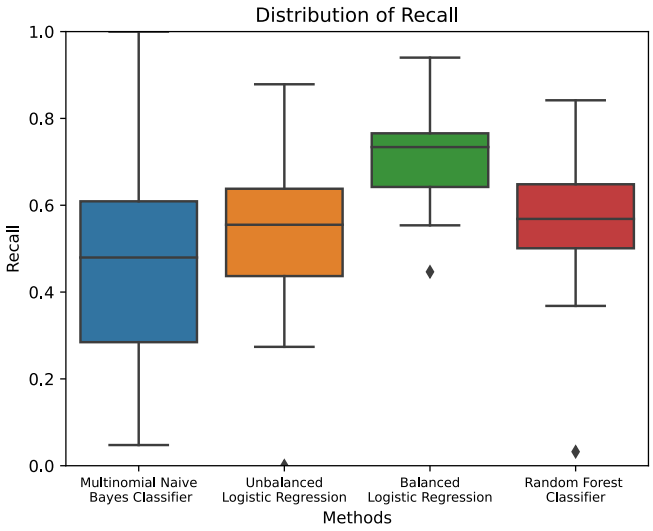
made, compared the previous one. The notable classification mistakes, or genre discoveries, that can be observed in the form of (True Label-Predicted Label) are:

1. Comedy-Drama
2. Documentary-Drama
3. Film-Noir-Thriller
4. Horror-Drama
5. Musical-Drama
6. Western-Drama

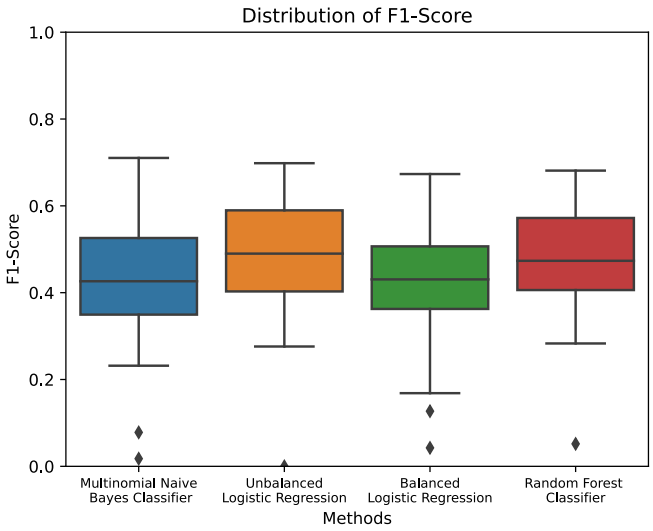
From Figure 4.10 we can see that the Drama label is used on many data points, indicating that this label is not correctly applied to our data. Moreover, we do not observe meaningful, generally accepted, genre correlations, as we did before e.g Action-War, Children-Comedy etc. Therefore, in comparison with the previous MLCM, using CLFD, these classification mistakes should be interpreted more confidently as mistakes rather than genre discoveries. A series of example movies and their respective reported and discovered genres can be seen on Table 4.3. Note that, on the genre discovery column, we include the new non-reported, discovered genres, while the reported genres that are not labelled through our classification method, are crossed out.



(A) Precision



(B) Recall



(C) F1-Score

FIGURE 4.8: Distribution of Precision, Recall, and F1-Score across different classifiers using TFIDF vectorization

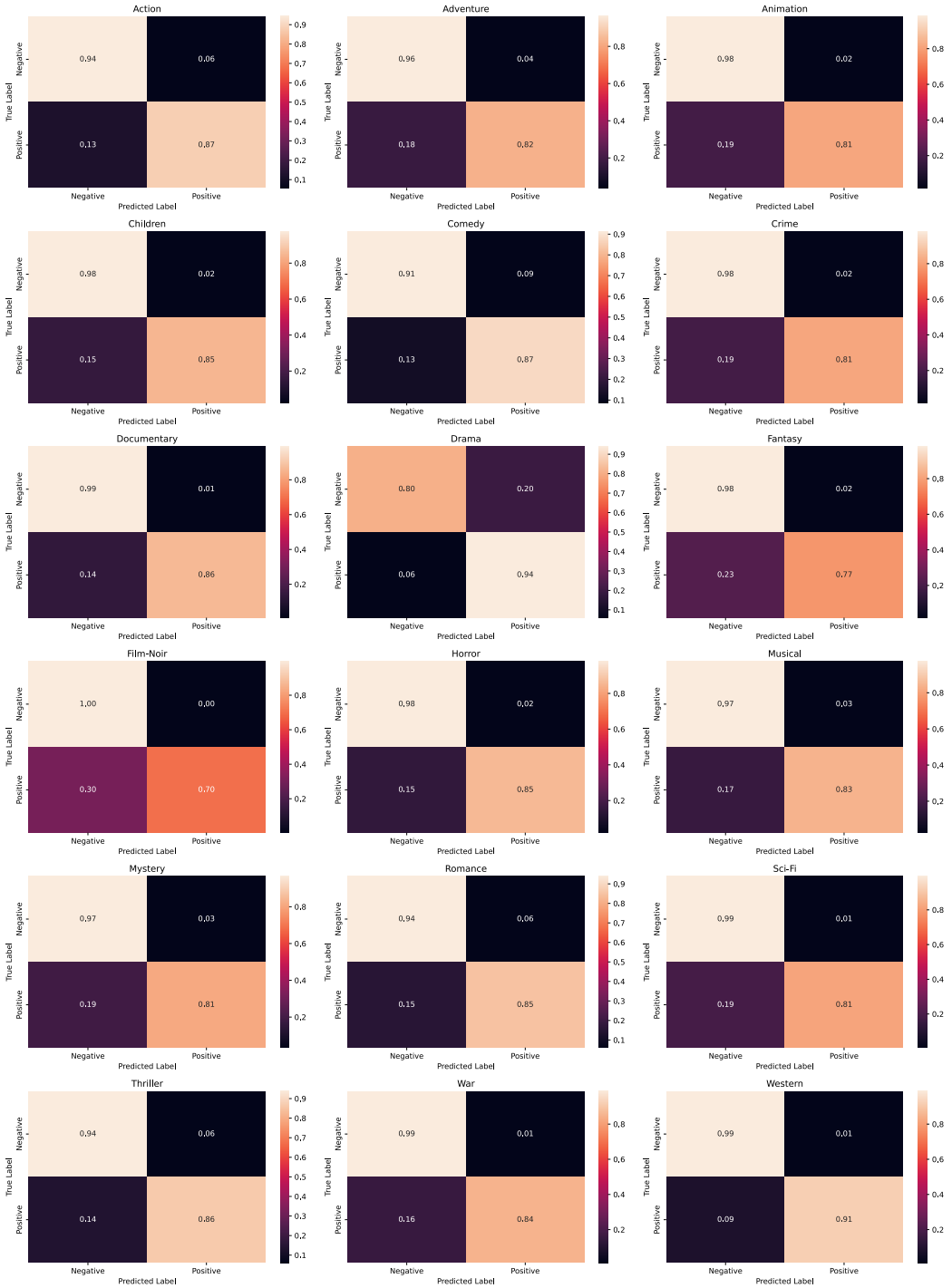


FIGURE 4.9: Confusion Matrices for each label using Random Forest classifier on TFIDF transformed data.

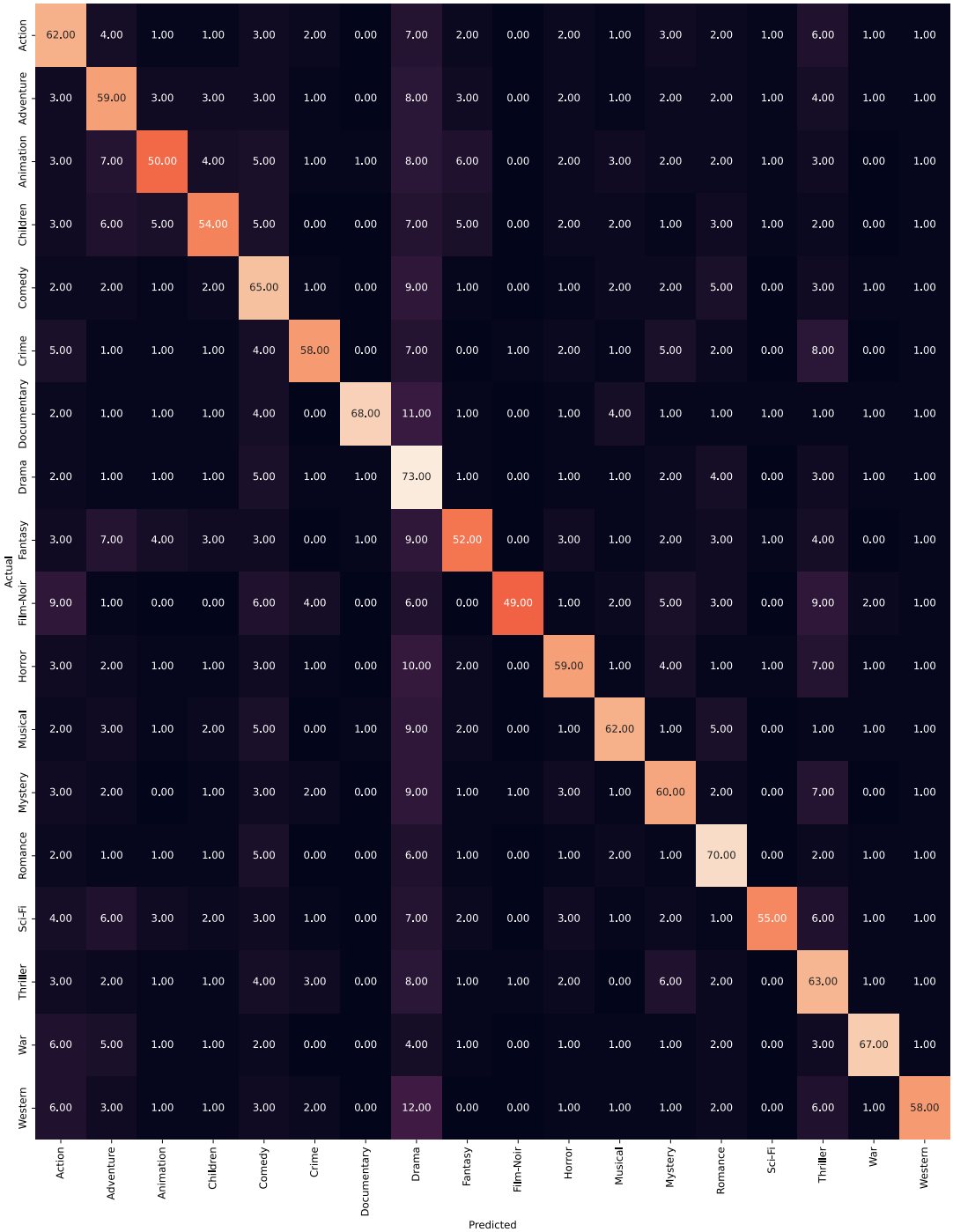


FIGURE 4.10: Multi-Label Confusion Matrix using Random Forest classifier on TFIDF transformed data.

4.2.4 Results of LSTM

The results of the LSTM approach can be observed on Table 4.8. We can see that LSTM performs quite well in the classification task and achieves similar metrics to the Unbalanced Logistic Regression using CLFD and the Random Forest classifier using TFIDF.

Figure 4.11 depicts the confusion matrices for each label. Compared to the previous confusion matrices, we observe that this classification technique is a lot more flexible. The True Negatives reach a mean score of 92.5%, but labels including Comedy, Drama and Romance demonstrate lower than average score. These labels are popular and sometimes over-used by the producers and thus their low score can be justified. Regarding True Positives we observe many labels that approach a score of 50%, including Adventure, Animation, Fantasy, Musical and Mystery. This result indicated that LSTM is a lot more flexible than the other two methods. Regarding the Film-Noir label, which was a challenge due to low number of samples, LSTM reaches a True Positive score of 38%, which is quite low compared to the other methods tested above.

Figure 4.12 shows the classification mistakes in more detail. We should note that these mistakes should be in line with our goal of discovering new genres. The classification mistakes, or genre discoveries, that are depicted on the MLCM are:

1. Adventure-Fantasy
2. Animation-Comedy and Fantasy
3. Children-Comedy, Fantasy and Musical
4. Comedy-Drama
5. Crime-Thriller
6. Fantasy-Comedy
7. Film Noir-Thriller
8. Horror-Thriller and Mystery
9. Musical-Romance
10. Thriller-Drama
11. Western-Drama and Action

The above classification "mistakes" can be interpreted as genre discoveries, as most of them are indeed correlated in nature. We should note that this technique produces the most reasonable "mistakes", as we cannot identify any non generally correlated classification "mistakes". Therefore, this technique is in line with our target of genre discovery. A series of example movies and their respective reported and discovered genres can be seen on Table 4.4. Note that, on the genre discovery column, we include the new non-reported, discovered genres, while the reported genres that are not labelled through our classification method, are crossed out. Word Embeddings is the key of the above results, as the technique deals with semantic analysis and word importance in a different, compared to text vectorization, manner.

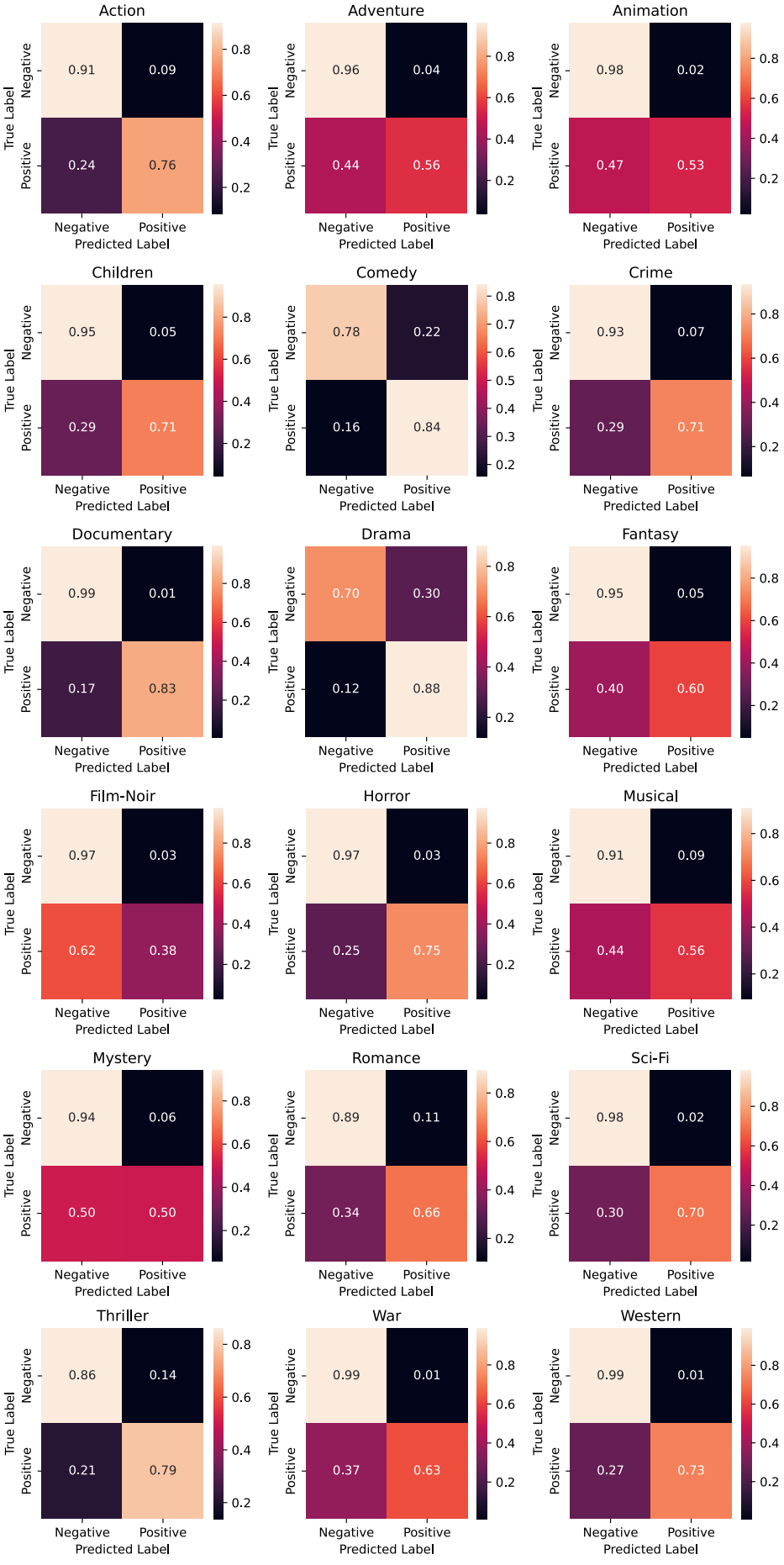


FIGURE 4.11: Confusion Matrices for each label using LSTM

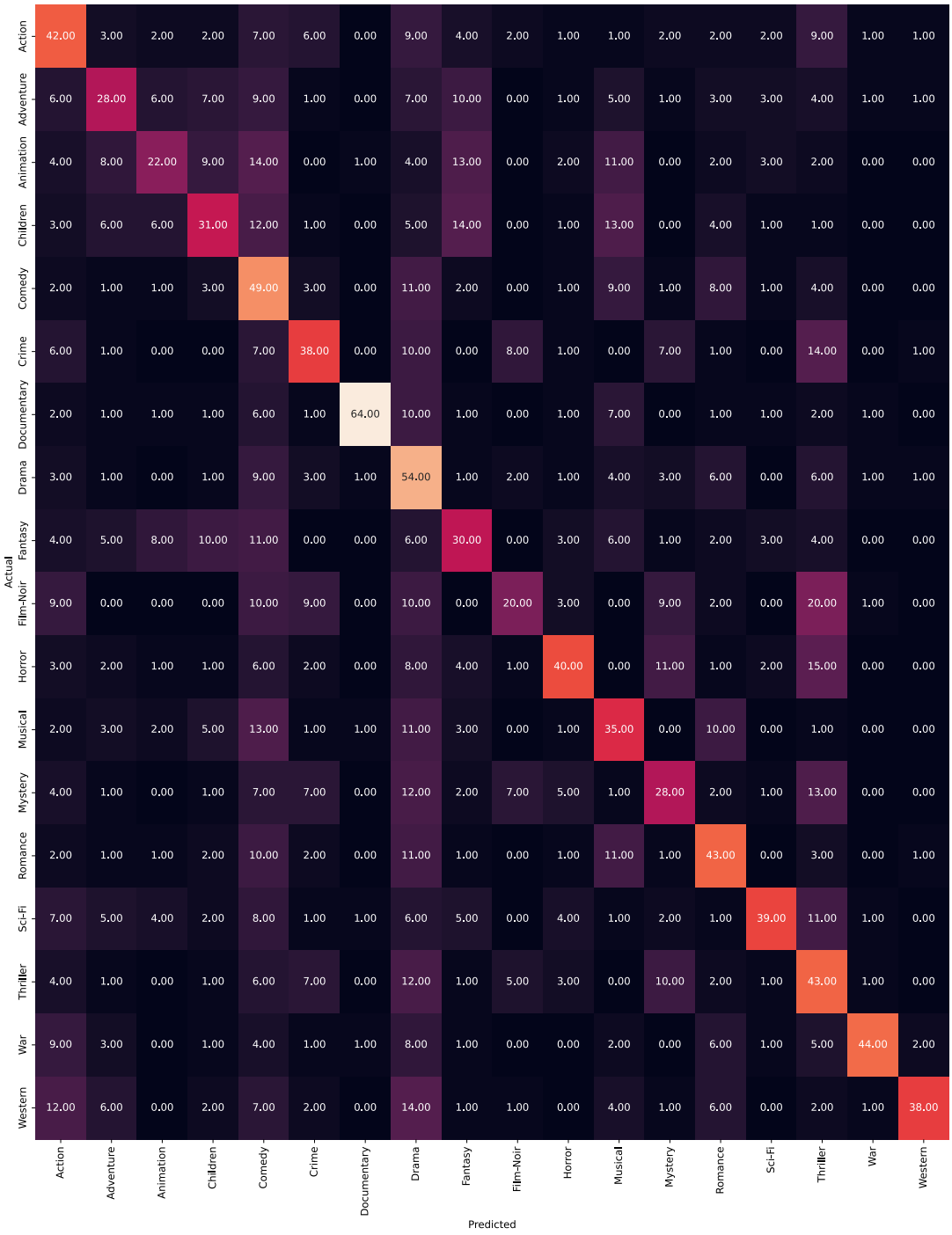


FIGURE 4.12: Multi-Label Confusion Matrix using LSTM.

Movie Title	Reported Genres	LSTM Genre Discoveries
Star Wars	Action, Adventure, Scifi	Animation, Comedy, Fantasy
Goodfellas	Crime, Drama	-
Titanic	Romance, Drama	-
Psycho	Crime, Horror, Thriller	~~Crime~~, Mystery
No Country For Old Men	Crime, Drama	Action, Thriller
Memento	Mystery, Thriller	~~Mystery~~, Action, Scifi

TABLE 4.4: Genre Discoveries using the LSTM neural network

4.2.5 Final Decision

Utilising Table 4.5, as well as the confusion matrices produced by the various classification methods tested in this section, we concluded that the Unbalanced Logistic Regression using CLFD transformed data and the LSTM are the classification methods that will be used for movie modelling. Although the Random Forest classifier with TFIDF transformed data performs quite similarly, we can observe that the Logistic Regression using CLFD transformed data achieves better metrics in all of the experiments carried out. Furthermore, as we observed during MLCM comparison, the Unbalanced Logistic Regression with CLFD and the LSTM approaches produce more meaningful genre discoveries compared to the Random Forest classifier with TFIDF. These genre discoveries are the core of this work and thus their importance should be stressed. Moreover, the Unbalanced Logistic Regression has the advantage of easy implementation as well as quick and efficient execution compared to Random Forest Classifier.

Our final decision is thus that our recommender system should be tested using the Unbalanced Logistic Regression approach on CLFD data, as well as the LSTM approach. Although these two mechanisms differ in development complexity, requirement of resources and execution duration, they can be efficiently adapted and used on the real-world recommender system domain.

	Classifier	Hamming loss	Jaccard similarity (weighted)	Jaccard similarity (sampled)
CLFD	Naive Bayes Classifier	0.1468	0.3573	0.2951
	Unbalanced Logistic Regression	0.1078	0.3948	0.4552
	Balanced Logistic Regression	0.1324	0.3829	0.4142
	Random Forest Classifier	0.1088	0.3928	0.4594
TFIDF	Naive Bayes Classifier	0.1583	0.3832	0.3087
	Unbalanced Logistic Regression	0.1126	0.3967	0.4585
	Balanced Logistic Regression	0.1912	0.3722	0.3427
	Random Forest Classifier	0.1200	0.3832	0.4386
	LSTM	0.1195	0.3723	0.4228

TABLE 4.5: Classification metrics

4.3 Training and Test Sets

The users of our recommender system are built based on the information found on our dataset. Each user is characterised by the ratings they have given to the movies they have watched. Our user modelling approach splits the rating set, of each user, on a training rating set and a test rating set, which from now on will be called training set and test set. respectively. The training set will be used to model the user's

preferences, while the test set will be used to evaluate our recommendations, based on real user data. Further analysis regarding the training and test set, of each user, exists in Section 4.3.1.

4.3.1 Bayesian User Modelling

Every user is modelled using a multivariate normal distribution, created using the their ratings on movies that they have watched. Each user's ratings are split in a training set, used for user modelling and a test set, used for evaluation. In order to test our Bayesian user modelling technique, based on *You Are What You Consume* [1], we employ the *Kullback-Leibler divergence* metric. The divergence metric is computed between the multivariate normal distribution, created with respect to the rating training set of a user u, and the multivariate normal distribution created using the whole rating set of user u. The training set reduces the number of ratings that a user provides to the system. Using this technique we evaluate the loss, in terms of divergence between distributions, introduced by the training set. We evaluate this metric for rating training set sizes $\in [0.02\%, 0.05\%, 0.1\%, 0.25\%, 0.5\%]$ of the original rating set.

Training Set Size	Kullback-Leibler Divergence
2%	0.0493
5%	0.0227
10%	0.0137
25%	0.0085
50%	0.0068

TABLE 4.6: Divergence between user models created with whole and reduced rating set

As we can see on Table 4.6, although the Kullback-Leibler divergence decreases as the rating training set size increases, it is kept on generally low quantities. A useful observation is that, divergence does not decrease linearly with the training set size increments. This means that using a small number of ratings, to model a user, can result in an accurate user model.

4.4 Recommendation Quality

This section evaluates our complete recommender system. Initially, we define the set of metrics that are used to test our system. Afterwards, we compare two versions of our recommender system, that use different mechanisms on the classification stage. The comparison is made between the LSTM classification approach and the Unbalanced Logistic Regression classification approach. Initially, we will evaluate our recommender system on a cross-validated domain, contrasting its performance with and without the integration of our social choice voting mechanism. With the use of the above experiment's results, we will evaluate the best performing version of our recommender system both on "niche" and "normal" users, and discuss our results.

4.4.1 Metrics Used

In order to test whether our recommendation mechanism succeeds in finding movies that a given user likes, we define a set of recommendation quality metrics.

The first metric that is used is the *Root Mean Squared Error-RMSE* between the rating of the movies that belong to the user's ratings test set and the predicted movie ratings calculated by our recommender system. The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}. \tag{4.5}$$

where n is the number of predictions, $\hat{y}_i$ are the predicted values and y_i are the actual values. Each user's real ratings, fetched from the dataset, are split in a training set, used to model the user, and a test set, used to for evaluation purposes. We should note that, during the recommendation process, a predicted rating is calculated for every dataset movie that the user has not already watched. The RMSE metric calculates the predicted rating error on movies that the user both likes and dislikes. Although our target is not to build an accurate recommender system for bad ratings, the RMSE metric provides an insight into the accuracy of our recommendations.

In order to test whether our movie recommendations are to the respective user's liking, we measure the divergence of the movie models found on the recommendation set and the user model. In order to measure the aforementioned divergence, we utilise the *Kullback-Leibler divergence* metric, defined in Equation 3.6, across two user models, the real user's model and one of a fictional user. The fictional user's model is constructed using information found on the recommendation set of the the real user, produced by our recommendation mechanism. More specifically, the fictional user, is a user that is modelled as a fresh pseudo user, who has rated only the movies that are found on the recommendation set, with a rating equal to the predicted rating that our recommender system believes the real user would give. Consequently, the fictional user model is a multivariate Gaussian distribution that entails the features of the movies that form the real user's recommendation set.

The last metric that is used to test the quality of our recommendations, is the *Mean Actual Rating*. As mentioned above, each user's real ratings, fetched from the dataset, are split in a training set, used to model the user, and a test set, used to for evaluation purposes. The recommendation set, consisting of 20 movies, is comprised of movies found in our vast dataset. The Mean Actual Rating is calculated as the average real rating, fetched from the user's test set, of the movies that are found both on the user's test set and our *extended recommendation set*. The extended recommendation set consists of the 100 movies, that received the most votes on the voting stage of our recommendation process in Section 3.6. As this metric is calculated using real world data, found on the user's test set, we define this metric as the ultimate test of our recommender system. We should note that the user's test set is only used for the calculation of this metric and its data are not used during training. Although this metric is of great significance, its calculation proposes a great challenge. The user's ratings test set is limited to a maximum of 1029 movies, for a training set size of 2%. Given the above two observations, a non-empty intersection of the above two sets is not guaranteed and therefore, the calculation of the Mean Actual Rating metric may not be able to be executed. In order to overcome the aforementioned challenge, we produced the extended recommendation set.

Training Set Size	K
2%	50
5%	20
10%	10
25%	4
50%	2

TABLE 4.7: Values of K to achieve certain training set sizes

4.4.2 Cross Validation

In order to make sure that our recommender system has real-world applicability, we should test the generalisation of our user and movie modelling techniques. Movie recommender systems often encounter the risk of overfitting, where a model becomes excessively sensitive to the training data and fails to generalise effectively to new, unseen data. k-Fold Cross-validation, used in this work, mitigates the risk stated above, by splitting the dataset into k subsets, training the model on different combinations of these subsets, and evaluating its performance across the aforementioned k splits. The averaged performance across the k splits is a robust evaluation technique of the recommender system's effectiveness. Consequently, the incorporation of k-Fold Cross-Validation is essential for the evaluation of our recommender system. On this section we present the cross-validated results of our recommender system, both with and without the use of our social choice voting mechanism. In the end, we will analyse our results and indicate the advantages and disadvantages of the two versions of our recommender system.

In our case, we need to evaluate our recommender system, under various training set sizes. Our user dataset includes the ratings that a certain user has a given to a set of movies. Initially, we split the user's dataset into k subsets, one of which will be used as a training set, in order to model the user. The number of ratings found on the training set depends on the number of splits. Therefore, we select the number of splits, k, on k-Fold Cross-Validation, in order to produce a predetermined training set size, following Table 4.7.

We will present our recommender system's cross-validated performance results, using 25 randomly picked normal users as well 25 randomly picked niche users. The choice of using 25 users, of each domain, was made given the computational cost of cross-validation. Moreover, we will analyse the recommendation process using both the Unbalanced Logistic Regression on CLFD transformed data (CLFD) and the LSTM approach. Moreover, we will compare our results with and without the use of our voting mechanism.

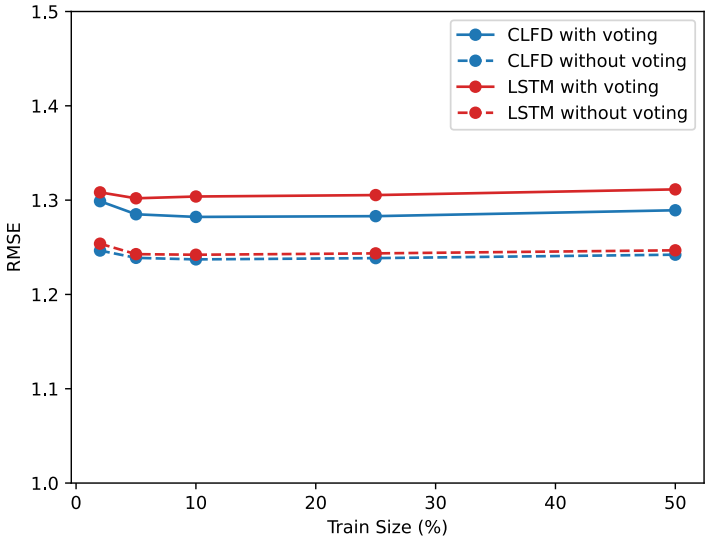
The voting mechanism that is utilised on our recommendation process uses voters that take into account both the movie's content and the movie's popularity. Using this voting rule aids in proposing a recommendation set that captures a wider variety of user preferences. Moreover, this rule captures several movie features that are entailed on movie content and popularity. On the contrary, using this voting rule, may result in a recommendation set that is not solely built on movie content (i.e., the divergence between user and movie models). In order to test our recommender system's performance without the use of the above voting rule, we carried out the same set of experiments, tweaking our recommendation process on using only the KL Divergence between the user's and movie's models. More specifically,

the recommendation set produced without the use of our voting mechanism, includes the 20 movies with the least KL Divergence between their models and the user model.

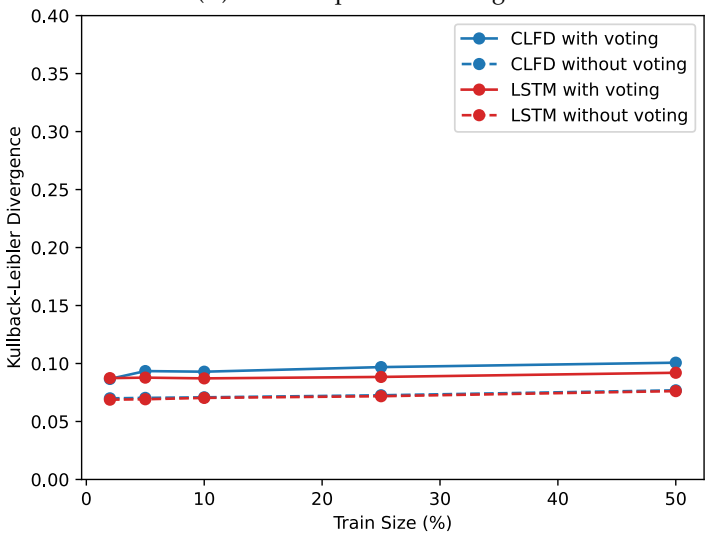
We start by presenting the results of our recommender system using our social choice voting mechanism on normal users first. The results of the metrics stated in the section above, regarding normal users, are depicted on Figure 4.13, using the "with voting" tag. Through the cross-validated form of our results we observe a similar behaviour of RMSE, Kl Divergence and Mean Actual Rating metrics, across the increasing training set size, on both methods. This is an indication that our system is able to make valuable suggestions, even with limited user feedback. The results indicate an RMSE value near 1.3 and a KL Divergence near 0.1, for both methods. Our ultimate recommendation quality test though, is the Mean Actual Rating of our suggestions. LSTM achieves the highest Mean Actual Rating of 3.4, compared to the CLFD, which reaches 3.25. Based on this observation, we can deduct that the LSTM approach is better on predicting ratings for movies that the user likes very much, or to put it differently, within a limited rating spectrum. CLFD, taking into account the common performance with LSTM on the RMSE metric, is able to accurately predict a bigger spectrum of ratings. If LSTM predicted the same spectrum of ratings, as accurately as CLFD, we would expect a similar performance regarding the Mean Actual Rating metric. We should also note, that we can observe a consistent higher rating on 2% training set size, on both methods. This abnormal result indicates that our system might slightly overfit the user's models.

The results of our recommender system quality metrics without the use of our social choice voting mechanism, on the normal user domain, are presented on Figure 4.13, using the "without voting" tag. We observe similarly linear behaviour of our recommendation quality metrics across the increasing training set size on both the Unbalanced Logistic Regression on CLFD transformed data and the LSTM. Regarding the RMSE quality metric, our results indicate a value of 1.25, for both methods. The KL Divergence value of both methods reaches a value of 0.7. The RMSE and KL Divergence metrics produced without the use of our voting mechanism indicate a performance increase compared to the results obtained with the use of our voting mechanism. Regarding the Mean Actual Rating metric, the LSTM method reaches a value just below 3.5, while the Unbalanced Logistic Regression with CLFD data reaches a value just below 3.4. We observe a similar performance increase on the Mean Actual Rating metric, on the both methods, compared to the results obtained with the use of our voting mechanism. The results of our experimental evaluation using our voting mechanism on the normal user domain, indicated that the best-performing LSTM method, reached a Mean Actual Rating score of 3.4. Comparing the LSTM with and without the use of our voting mechanism, we observe that the absence of our voting mechanism on the normal user domain, increases the Mean Actual Rating by 0.1. In any case, the use of the voting mechanism does contribute to the wider exploration of the recommendations' set, combining as it does content-related user preferences with movies' popularity. Therefore, the use of our voting mechanism on the normal user domain, using the LSTM method can be beneficial for our recommender system's performance, despite the negligible performance decrease on the Mean Actual Rating metric.

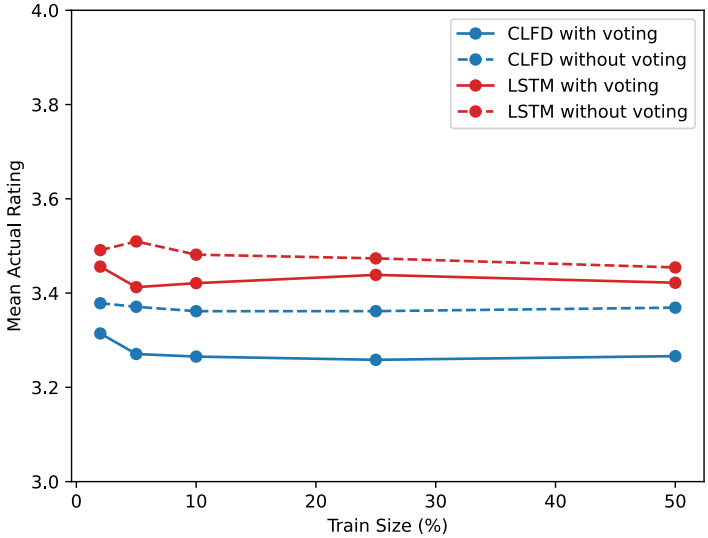
Our recommendation quality metrics for niche users with the use of our voting mechanism are depicted on Figure 4.14, using the "with voting" tag. Similarly to



(A) RMSE of predicted ratings



(B) KL-Divergence between real and fictional user

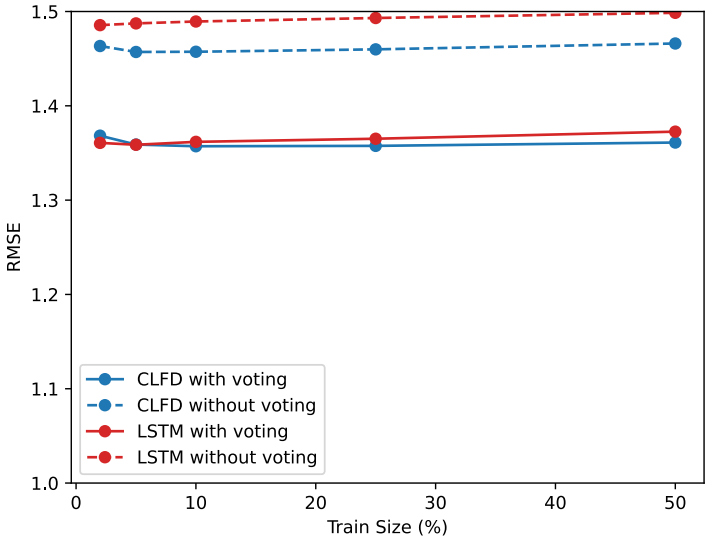


(C) Mean Actual Rating

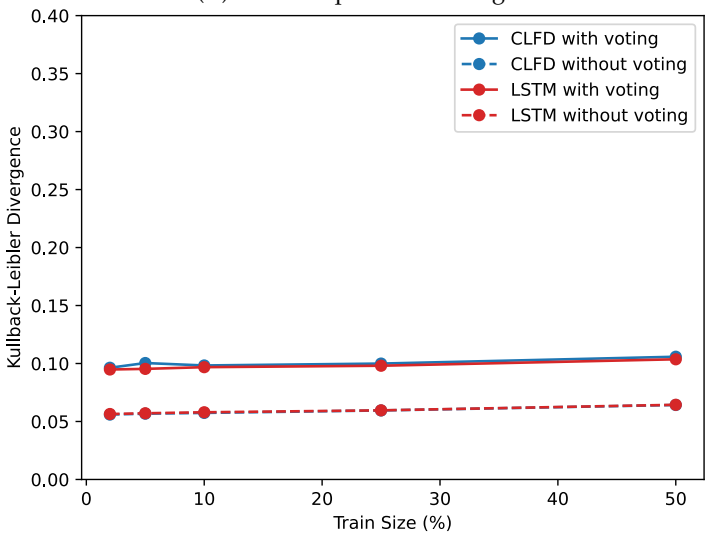
FIGURE 4.13: Recommendation quality metrics on 25 cross-validated normal users with and without the use of our voting mechanism

normal users, we observe a consistent performance of all our metrics across the increasing training set size. The RMSE and KL Divergence metrics perform the same on the CLFD and the LSTM method. We should note, though, that the RMSE metric is higher on niche users, indicating that their preferences are indeed, harder to infer. Our system reaches an RMSE of 1.37, between predicted and actual ratings of niche users, but we should note that this behaviour does not actually affect the performance. The nature of the RMSE metric makes it very sensitive to minor differences. The performance of our system on the niche user domain can be more clearly evaluated via the Mean Actual Rating metric. As we can see, both methods showcase a performance comparison, similar to that of normal users with the use of our voting mechanism. LSTM reaches a higher average Mean Actual Rating of 3.5 than CLFD, which reaches 3.2. Comparing this metric among normal and niche users, with the use of our voting mechanism, we observe that LSTM performs better on niche users, while CLFD preforms slightly better on the normal user domain. This result signifies that LSTM, through the use of Word Embeddings and neural network classification architecture, is able to capture movie features that the CLFD approach cannot. The advantage of LSTM is particularly noticeable on niche users, whose preferences are harder to infer.

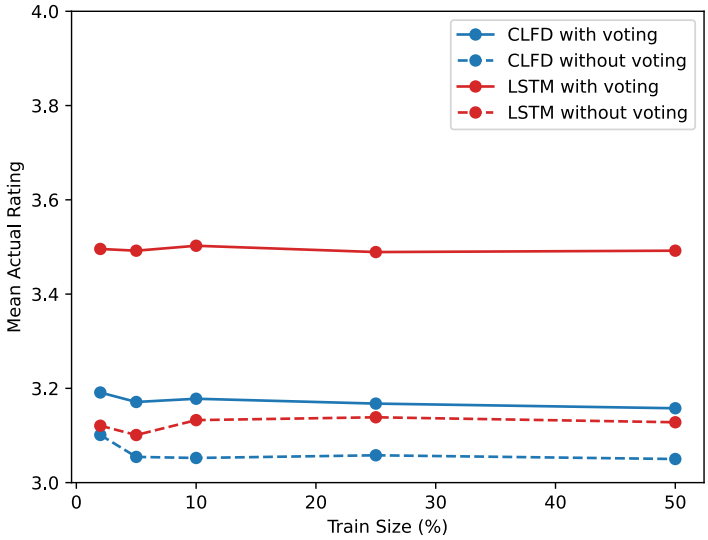
The results of our recommender system quality metrics without the use of our social choice voting mechanism, on the niche user domain, are presented on Figure 4.14, using the "without voting" tag. In this case as well, we observe that our recommendation quality metrics present a stable behaviour across the increasing training set size for both methods. On the RMSE and KL Divergence metric both the Unbalanced Logistic Regression on CLFD data and the LSTM approach reach similar results. More specifically, on the RMSE metric they reach a score just below 1.5, indicating a performance decrease compared to the results on the normal user domain, without our voting mechanism in use. Compared to the results on the niche user domain with the use of our voting system, we observe that the use of our voting mechanism has a positive impact on the RMSE metric, as the relative experiments reached a score of 1.37. Regarding the KL Divergence metric, both methods reach the score of 0.05, indicating a slight performance increase compared to the normal user domain without the use of our voting mechanism. Comparing the results on the niche user domain, with and without the use of our voting mechanism, we observe that a higher KL Divergence score is reached with the use of our voting mechanism. Regarding the Mean Actual Rating, our most indicative recommendation quality metric, the LSTM method reaches a score of 3.15, while the Unbalanced Logistic Regression on CLFD data reaches a score of 3.05. Compared to the normal user domain, without the use of our voting mechanism, we observe a performance decrease both on the LSTM method and the Unbalanced Logistic Regression. We should note that regarding the best-performing LSTM method, the use of our voting mechanism increases the performance on the Mean Actual Rating metric by 0.35. Therefore, the use of our voting mechanism has a strong impact on the niche user domain.



(A) RMSE of predicted ratings



(B) KL-Divergence between real and fictional user



(C) Mean Actual Rating

FIGURE 4.14: Recommendation quality metrics on 25 cross-validated niche users with and without the use of our voting mechanism

Comparing the results with and without the use of our voting mechanism, we observe that the use of our voting mechanism enhances the performance of our recommender system on the niche users, while maintaining similar performance results on the normal user domain. As stated before though, the utilisation of our voting mechanism contributes to the broader exploration of the recommended set. Therefore, the final version of our recommender system will include this mechanism on the recommendation process.

Examining our results further, Figure 4.15 depicts the variance of the Mean Actual Rating metric, that the recommender system achieved using our voting system. We can see that both the Logistic Regression using CLFD transformed data, as well as the LSTM approach, achieve generally low variance. We also observe that the variance of niche users, due to their niche nature, is higher than the variance of normal users. Moreover, on the normal user domain we can see that the LSTM approach has a lower variance than the Logistic Regression using CLFD transformed data. On the niche user domain both the LSTM and the Logistic Regression, achieve similar variance values.

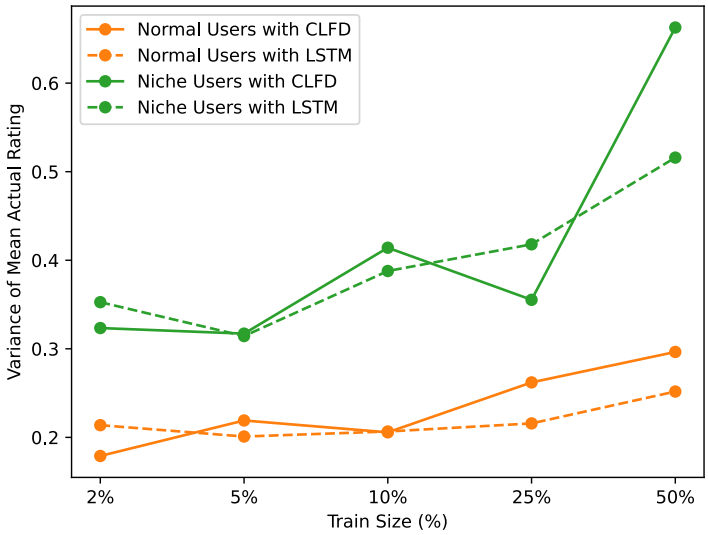


FIGURE 4.15: Variance of Mean Actual Rating using our voting mechanism

4.4.3 Further Results on Larger Datasets

In order to test our recommender system on a larger set of users, and thus acquire an indicative representation of performance results, we now recommended movies to a larger number of randomly picked normal and niche users. Note that the results that are presented in this section are achieved using our social choice driven voting mechanism on the recommendation process. Movie modelling was done using the Unbalanced Logistic Regression on CLFD data and the LSTM architecture. The Unbalanced Logistic Regression on CLFD data technique will be simply called *CLFD* for abbreviation purposes. Regarding normal users, we recommended movies to 300 randomly picked users from the normal user set and present the average results across our recommendations. The same evaluation technique was followed for niche users, with the exception of using all the 61 niche users, that are found on our filtered dataset. The results regarding the quality of our recommendations on 300 randomly picked normal users and the 61 niche users are depicted on Figure 4.16 and Figure 4.17, respectively. The aforementioned figures include the results using both the LSTM and the CLFD movie modelling techniques. The data used to plot the figures is present on Table 4.8, for further analysis.

We will initially observe the correlation of our quality metrics and the training set size. On both normal and niche user's quality metrics of CLFD and LSTM, we observe that the RMSE and the Mean Actual Rating metrics do not change significantly with the increase of the training set size. This is a good indicator that our system can produce meaningful recommendations, even when there is limited user data present. On the LSTM movie modelling technique, we observe a clear reduction of the KL divergence as the training set's size increases, on both normal and niche users. This reduction, although present on both normal and niche users, is less obvious on the CLFD approach.

During the Mean Actual Rating metric calculation, we observed that the metric on normal users was calculated among an average of 67.4 movie matches per recommendation. On niche users the metric was calculated among an average of 15.7 movies per recommendation. The above results signify that the Mean Actual Rating metric is calculated among an acceptable number of instances and therefore, can be used to accurately evaluate our recommender system.

Regarding normal users, we observe that the LSTM movie modelling technique outperforms CLFD, on the Mean Actual Rating metric. More specifically, our recommender system, using the LSTM movie modelling technique, reaches a higher Mean Actual Rating, on every training set size, compared to the CLFD approach. The Mean Actual Rating of the LSTM approach is consistently above 3.4, while the CLFD approach reaches a maximum of 3.36. On niche users, the distinction between the two movie modelling methods is more obvious. The LSTM approach reaches a Mean Actual Rating of 3.52, while the CLFD approach reaches 3.21. As we stated before, the Mean Actual Rating is the ultimate test of our recommender system. Recommending movies that a user will enjoy is the target of our recommendation approach. Therefore, our discussion over the the results of our experiments will be heavily focused on Mean Actual Rating.

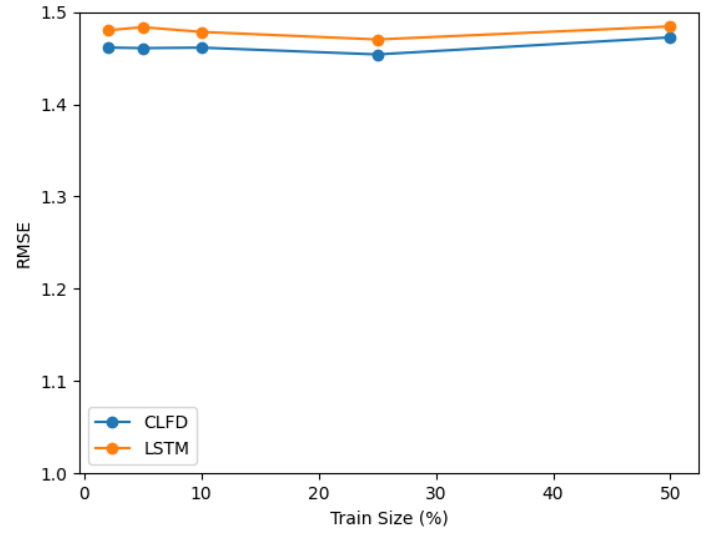
Regarding the KL divergence of fictional and the real normal users, we observe that CLFD reaches lower divergence values than LSTM, on the 2% training set size, but is then outperformed. On niche users, KL divergence is quite lower on the LSTM

	User Type	Logistic Regression with CLFD				
Training Set Size		2%	5%	10%	25%	50%
RMSE	Normal	1.3144	1.2783	1.3173	1.3209	1.3173
	Niche	1.4615	1.4608	1.4614	1.4540	1.4725
KL Divergence	Normal	0.2394	0.1997	0.2700	0.1614	0.2894
	Niche	0.3857	0.3068	0.3287	0.2808	0.2755
Mean Actual Rating	Normal	3.3101	3.3629	3.2811	3.2942	3.3036
	Niche	3.1915	3.2195	3.2163	3.2280	3.1547
		LSTM				
Training Set Size		2%	5%	10%	25%	50%
RMSE	Normal	1.3251	1.3232	1.3287	1.3269	1.3336
	Niche	1.4802	1.4835	1.4784	1.4702	1.4844
KL Divergence	Normal	0.2672	0.1816	0.1658	0.1435	0.1418
	Niche	0.1949	0.1846	0.1485	0.1683	0.1498
Mean Actual Rating	Normal	3.4418	3.4286	3.4323	3.4043	3.4123
	Niche	3.5218	3.5095	3.5007	3.5079	3.5275

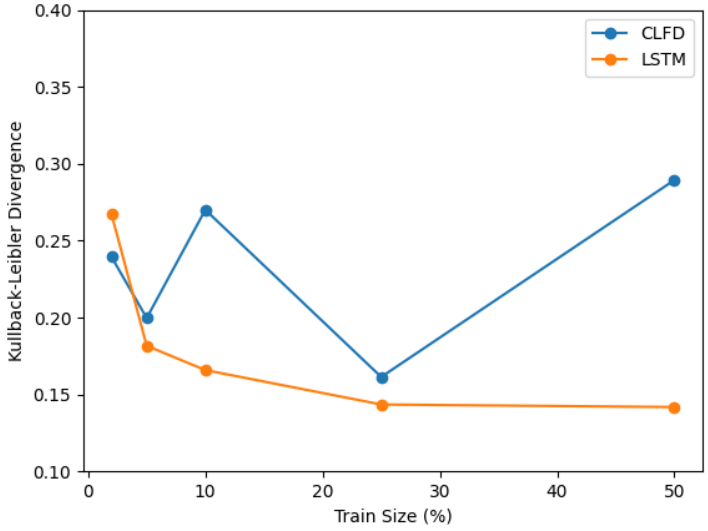
TABLE 4.8: Recommendation Process performance on different user types, across various rating training set sizes

approach, reaching a value of 0.17, compared to the CLFD approach, which averages 0.3. On the RMSE metric, both methods perform nearly the same, with CLFD having a slight advantage. On normal users the RMSE is just below 1.5, while on the niche user domain, the error decreases to 1.3 for both methods. The lower KL divergence of the LSTM method signifies that the aggregation of the recommendation's set movie models, is closer to the user model, than the recommendation set produced with the CLFD approach. Moreover, the fact that the RMSE metric is similar on both approaches, signifies that CLFD has the ability to accurately predict a wider spectrum of movie ratings. If LSTM had the aforementioned ability, it would reach a lower RMSE value than CLFD, given that it outperforms CLFD on every experiment. The above leads to the conclusion that CLFD is more accurate across the whole span of predicted ratings, while LSTM is accurate only on high ratings. This discovery can prove useful, if applied in the group recommendation domain.

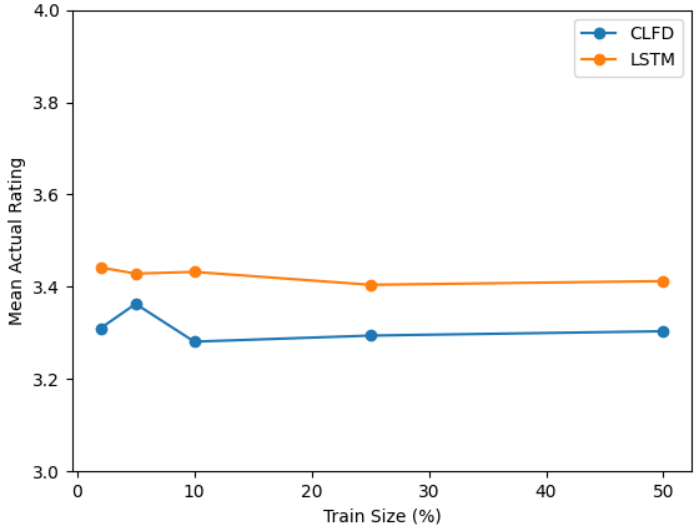
It is clear that the LSTM approach performs better on the niche user domain, contrary to the CLFD approach, which performs better on the normal user domain. Niche user's preferences are more vibrant and distinct, compared to normal user preferences. As we saw on the MLCM matrices of the LSTM and the CLFD approach, the LSTM technique produced results that had a significant deviation from the dataset, but included meaningful misclassification errors that could be interpreted as genre discoveries, due to their natural correlation. Therefore, the LSTM movie modelling technique favours niche users, whose preferences depend more on movie content rather than other, non modelled features, such as movie budget and cast, which do, most of the time, have an impact on normal movie audience. All in all, using the above experiments, we can conclude that the LSTM approach outperforms CLFD for both niche and normal users, with a specific focus on catering to the niche user domain.



(A) RMSE of predicted ratings

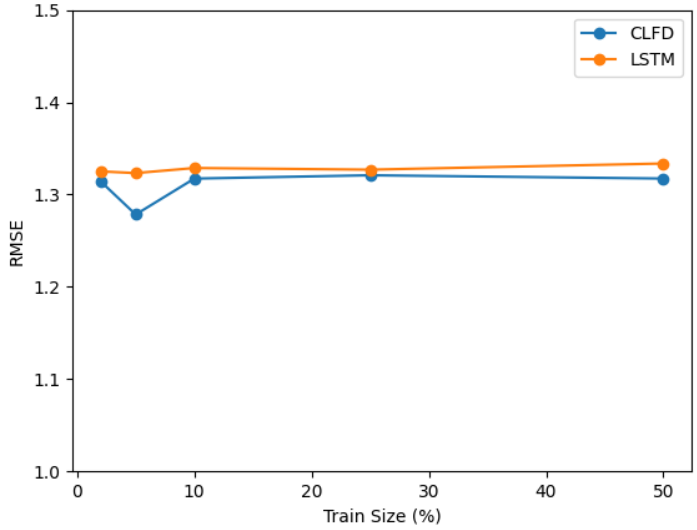


(B) KL-Divergence between real and fictional user

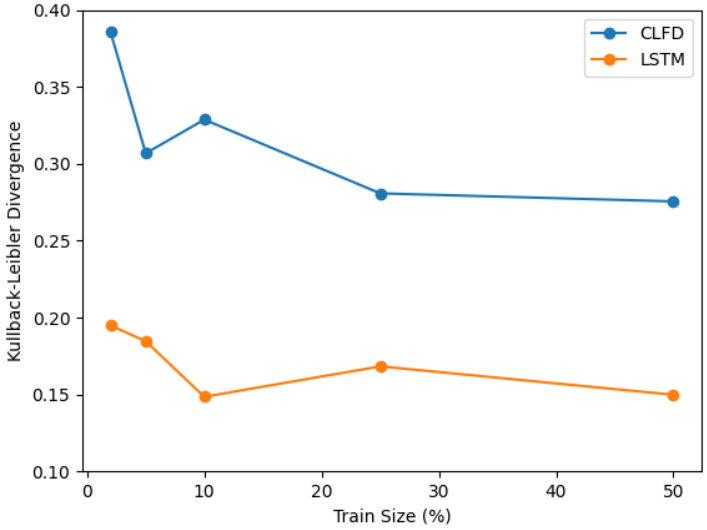


(C) Mean Actual Rating

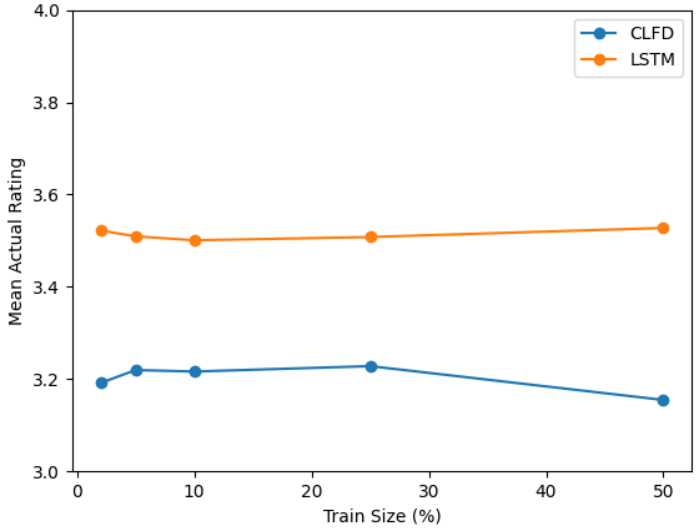
FIGURE 4.16: Recommendation quality metrics on 300 randomly sampled users



(A) RMSE of predicted ratings



(B) KL-Divergence between real and fictional user



(C) Mean Actual Rating

FIGURE 4.17: Recommendation quality metrics on 61 niche users

Chapter 5

Conclusions and Future Work

In this work, we proposed a novel movie modelling technique based on movie genre labels, produced through text summaries; interestingly, combining various text vectorization techniques with a Bayesian recommendation method and a social choice theory-originating voting mechanism. We tested the TFIDF, CLFD and Count text vectorization techniques, as well as various Natural Language Processing filters, to effectively transform summary text to a usable format. Using the transformed data, we tested a set of different classification architectures to label each movie with the set of genres that characterise it.

In order to evaluate our text vectorization and classification stage, we employed various metrics, to help us test our different techniques. The analysis of our experiments, led to the conclusion of choosing the Unbalanced Logistic Regression using CLFD transformed data and the LSTM approach, as they produced the most meaningful results. Therefore, these techniques were ultimately chosen, and tested on our recommender system.

Lastly, we evaluated our social-choice based recommendation approach on real users and compared our results to a version of our recommendation system without the use of our social-choice mechanism. Our experiments were carried out among two types of users, normal and niche. The selection of two different user types was utilised, in order to examine our system's performance on both relaxed, and strict, preferences. The recommendation stage evaluation included the definition of recommendation quality metrics, the cross-validation of our results and the analysis of our recommender system's performance for various train set sizes. Ultimately we discussed the advantages and disadvantages of our chosen methods.

The results indicated that our social choice theory-aided recommender method exhibits very good performance in terms of recommendations' quality. More specifically, our recommender system's suggestions reach an average rating, on a scale of $[0.5, 5]$, of 3.5 on niche users, and an average rating of 3.4 on normal users. These results are consistent among various degrees of user feedback, thus we conclude that our system can provide valuable movie suggestions, to users that have rated more than 19 movies. Moreover, we should note that our recommender system is tested on real user data and uses computationally efficient techniques, that can be employed on the real world. We should also note that this recommender system, despite the fact that was tested using small train set sizes, is not tailored to tackle the cold start problem. On the real-world domain, recommender systems tend to follow a hybrid approach, differentiating their suggestion mechanisms among fresh and accurately modelled users. Our approach can be used as part of a larger hybrid

recommendation technique, focusing on users that have provided, even minimal, feedback.

5.1 Future Work

This section focuses on a set of proposed actions that can further aid our system's performance. Movie modelling can benefit from the use of more text-based data, including user reviews or different summary versions. This extended input domain can provide our system with more information, thus increasing the ultimate performance.

Moreover, future work involves extending the application of our modelling techniques and recommendation approach, to a real-world hybrid recommender system. In particular, we propose an integration of our methods on a multi-winner election mechanism, combining various recommendation approaches, aiming on enhanced performance. The use of movie summaries, although entailing many features that characterise a movie, is not able to capture every aspect that the audience examine. It is, though, a modelling technique that can prove valuable, when used on particular users. It is known that many people's satisfaction relies, solely, on movie content. These users can benefit from our heavily content-based approach, thus proving that our work can have a significant impact on the recommender system domain.

Another application area of our recommender system is the book domain. As our item modelling technique relies on text data, applying our approach on books could be beneficial. Using book summaries, or even its whole text content, could result in a quite accurate representation of its nature, therefore increasing our recommender system's performance.

Another extension of our approach is its use on a group recommender system. As stated in our experimentation stage, the movie modelling technique of the Unbalanced Logistic Regression with CLFD transformed summaries, has a significant advantage when it comes on accurately predicting a wide range of ratings. Consequently, the use of the above method can benefit the group recommendation domain, as it can accurately model the extent of a user's liking towards certain items.

Bibliography

[1] Konstantinos Babas, Georgios Chalkiadakis, and Evangelos Tripolitakis. "You Are What You Consume: A Bayesian Method for Personalized Recommendations". In: *Proc. of the 7th ACM Conf. on Recommender Systems*. RecSys '13. Hong Kong, China: ACM, 2013, pp. 221–228. ISBN: 9781450324090.

[2] Georgios Chalkiadakis, Ioannis Ziogas, Michail Koutsmanis, Errikos Streviniotis, Costas Panagiotakis, and Harris Papadakis. "A Novel Hybrid Recommender System for the Tourism Domain". In: *Algorithms* 16.4 (2023). ISSN: 1999-4893. DOI: 10.3390/a16040215. URL: https://www.mdpi.com/1999-4893/16/4/215.

[3] G. Chowdhury. *Introduction to Modern Information Retrieval, Third Edition*. 3rd. Facet Publishing, 2010. ISBN: 185604694X.

[4] Carlos A. Gomez-Uribe and Neil Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4 (2016). ISSN: 2158-656X. DOI: 10.1145/2843948. URL: https://doi.org/10.1145/2843948.

[5] GroupLens. *MovieLens Latest Datasets*. URL: https://grouplens.org/datasets/movielens/latest/.

[6] Colton J. Herrington. "Film Marketing and the Creation of the Hollywood Blockbuster". In: 2015. URL: https://api.semanticscholar.org/CorpusID:114479943.

[7] Mohammadreza Heydarian, Thomas Doyle, and Reza Samavi. "MLCM: Multi-Label Confusion Matrix". In: *IEEE Access* 10 (Feb. 2022). DOI: 10.1109/ACCESS.2022.3151048.

[8] *IMDB definition of movie plot*. URL: https://help.imdb.com/article/contribution/titles/plots/G56STCKTK7ESG7CP#.

[9] Yanli Liu, Yourong Wang, and Jian Zhang. "New Machine Learning Algorithm: Random Forest". In: *Information Computing and Applications*. Ed. by Baoxiang Liu, Maode Ma, and Jincai Chang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 246–252. ISBN: 978-3-642-34062-8.

[10] Michail Mersinias, Stergos Afantenos, and Georgios Chalkiadakis. "CLFD: A Novel Vectorization Technique and Its Application in Fake News Detection". English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020,

pp. 3475–3483. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.427.

[11] Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[12] *Natural Language Toolkit*. URL: https://www.nltk.org/.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. "Classifier Chains for Multi-label Classification". In: (2009). Ed. by Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor, pp. 254–269.

[15] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems Handbook". In: vol. 1-35. Oct. 2010, pp. 1–35. ISBN: 978-0-387-85819-7. DOI: 10.1007/978-0-387-85820-3_1.

[16] Josef Schmee and T. Anderson. "An Introduction to Multivariate Statistical Analysis". In: *Technometrics* 28 (1986), p. 180.

[17] Ralf C. Staudemeyer and Eric Rothstein Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019. arXiv: 1909.09586 [cs.NE].

[18] Errikos Streviniotis and Georgios Chalkiadakis. "Preference Aggregation Mechanisms for a Tourism-Oriented Bayesian Recommender". In: Nov. 2022, pp. 331–346. ISBN: 978-3-031-21202-4. DOI: 10.1007/978-3-031-21203-1_20.

[19] Errikos Streviniotis and Georgios Chalkiadakis. "Preference Aggregation Mechanisms for a Tourism-Oriented Bayesian Recommender". In: *PRIMA 2022: Principles and Practice of Multi-Agent Systems*. Ed. by Reyhan Aydoğan, Natalia Criado, Jérôme Lang, Victor Sanchez-Anguix, and Marc Serramia. Cham: Springer International Publishing, 2023, pp. 331–346. ISBN: 978-3-031-21203-1.

[20] *The Movies Dataset*. URL: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset.

[21] Evangelos Tripolitakis and Georgios Chalkiadakis. "Probabilistic Topic Modeling, Reinforcement Learning, and Crowdsourcing for Personalized Recommendations". In: *Multi-Agent Systems and Agreement Technologies*. Ed. by Natalia Criado Pacheco, Carlos Carrascosa, Nardine Osman, and Vicente Julián Inglada. Cham: Springer International Publishing, 2017, pp. 157–171. ISBN: 978-3-319-59294-7.