



Department of Management Science and Technology
MSc in Business Analytics

Movie Recommendation Engine

using an interactive conversational agent

By

Spanos Nikolaos

Student ID Number: F2821826
Name of Supervisor: Louridas Panagiotis

August 2020
Athens, Greece



© Copyright by Athens University of Economics and Business &
SPANOS NIKOLAOS, 2020
All Rights Reserved



To the Faculty of Athens University of Economics and Business:

The members of the Committee appointed to examine the thesis of SPANOS NIKOLAOS find it satisfactory and recommend that it be accepted.

Supervisor: Louridas Panagiotis, Associate Professor

Examiner 2: Diomidis Spinellis, Professor

Examiner 3: Damianos Chatziantoniou, Associate Professor

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

DEDICATION

This dissertation/thesis is dedicated to my beloved family and Pitarokoili Olga who provided emotional support
and to my professor Louridas Panagiotis who shared with me his knowledge and expertise.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ACKNOWLEDGMENTS

I must chiefly acknowledge Louridas Panagiotis, Perakis George and Pitarokoili Olga for their support.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ABSTRACT

To improve the quality of services provided to the users, researchers, and online streaming companies have extensively researched and deployed content-based movie recommender systems. They improve the satisfaction and time-efficiency of users by utilizing Artificial Intelligence (AI) applications and deep learning. Those applications are applied upon standard business solutions to automate old-fashioned ways of human-computer interaction. Content-based movie recommender systems' problem lies in the lack of content provided either by the user or related to the movie. Recommendation algorithms lacking content cannot propose accurate recommendations for single genre movies or movies linked simultaneously on multiple genres (e.g., action and romance movies). This thesis uses Natural Language Processing (NLP) techniques to assemble an accurate content-based movie recommendation engine that will exploit content from users and online movie sources. Because of their content, movies are classified into multiple genres. For example, a movie can be at the same time romantic and adventurous. A movie's connection to more than one genre adds an extra hurdle on a recommender engine since the latter should correctly identify the keywords and the semantics that make a movie both romantic, adventurous, and documentary. Thus, the need arises to create an accurate recommender engine that will understand and exploit user preferences and movie content to propose multi-genre movies.

The movie recommendation engine developed in the context of this thesis comprises two tightly-coupled components. The first component is a neural network that solves the multi-genre text classification problem. The thesis elaborates on the development, training, and evaluation of five neural networks for multi-label text classification. The neural networks will be optimized to infer a set of relevant movie genres from movie content by using five different movie features, precisely a plot summary, the actors, the movie features, the title,

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

and the user reviews. Then, we compare the ability of the five different neural networks to address the text classification problem. The first neural network is a custom-made neural network created by the authors to explicitly solve the movie genre multi-label classification problem, whereas, the other four neural networks are pre-trained and saved model estimators that solve similar NLP text classification tasks. For highlighting the custom-developed neural network as the best model estimator out of the five neural networks, the document applies a thorough selection plan that involves: (a) the hamming loss and binary cross-entropy loss values on never-seen-before movies, (b) the learning curves of the training and the validation samples, (c) the bias-variance tradeoff and (d) the confusion matrices and the classification reports of the movie genre classifiers. The second component is an interactive conversational agent that recommends movies in real-time based on the content written by the user(s) and provided from the movie itself. The user can write anything related to the movie and like about the movie. For example, write a short review of the movie or a compliment for the plot and the movie's cast. The agent ranks the movies from the most similar to the least similar, based on the predictions of the best classifier and then calculates a *movie-score* based on two elements: (1) the information provided by the user, (2) a general user opinion ranking (e.g., IMDB rating, sentiment rating, and viewer rating). The agent recommends the movie with the highest movie score. Finally, we compare the functionality and recommendation engine results with similar services of the e-market by asking the public opinion.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Contents

Page

1 Introduction 1
- 1.1 Summary 1
- 1.2 Thesis Scope 3
- 1.3 Thesis Outline 5
- 1.4 Python code outline 8

2 Data Text Analysis 13
- 2.1 Introduction 13
- 2.2 Data Requirements 15
- 2.3 Data Collection 19
  - 2.3.1 IMDB Website and Grouplens dataset 19
  - 2.3.2 Data Extraction - Web Scrapping 21
- 2.4 Data Preprocessing 24
  - 2.4.1 Data Cleaning 24
  - 2.4.2 NLP Text Preprocessing 26
  - 2.4.3 Prepare the input for deep learning experimentation 51

3 Neural Network Development 59
- 3.1 Introduction 59
- 3.2 Custom Neural Network 60
  - 3.2.1 Development 60
  - 3.2.2 Compiling & Training the neural network 74
  - 3.2.3 Select best model classifier (per optimizer) 81
  - 3.2.4 Hamming loss model estimator vs. Binary accuracy model estimator . 112
- 3.3 Neural networks of pre-trained word embeddings 115
  - 3.3.1 GloVe pre-trained word embeddings 115

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3.3.2 English Google News 130GB corpus | 20 dimensions & without OOV token . . . 118
3.3.3 English Google News 130GB corpus | 20 dimensions & OOV token . . 120
3.3.4 English Google News 7B corpus | 50 dimensions & OOV token 122
3.3.5 Select best model classifier (out of the five neural networks) 124

4 Engineering an Interactive Conversational Agent . . . 137
4.1 Introduction . . . 137
4.2 Neural Network Weights | Word Vector Embeddings . . . 138
4.3 Assembling an Interactive Conversational Agent . . . 144
4.3.1 What is a chatbot . . . 144
4.3.2 Architecture of the chatbot . . . 145
4.3.3 Dialogflow-Hosting the chatbot . . . 150
4.4 Conversational Agent-Opinion based research . . . 153
4.4.1 Opinion-based questionnaire . . . 153
4.4.2 Questionnaire results . . . 156
4.4.3 Questionnaire conclusions . . . 169

5 Conclusions . . . 173

Appendices

A First Appendix . . . 179
A.1 List of Abbreviations . . . 179

References . . . 184

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

List of Figures

2.1 Text data preprocessing framework Source: Text Preprocessing Framework [13] 27

2.2 Percentage of genres per movie 56

2.3 Stratified shuffle split on training & test samples 56

3.1 Neural network's structure 73

3.2 Output 1: Training per epoch 79

3.3 Output 2: Neural Network loss function and hamming loss, f1 score progress per epoch 79

3.4 Output 3: Training-Validation Hamming loss, F1 score & Loss per epoch . . . 80

3.5 Output 4: Hamming loss, Loss score performance of the neural network on never-seen-before movies 80

3.6 Adam optimizer Classification Report results 92

3.7 SGD optimizer Classification Report results 92

3.8 RMSprop optimizer Classification Report results 93

3.9 Underfitting Learning Curve - Part 1 103

3.10 Underfitting Learning Curve - Part 2 103

3.11 Overfitting Learning Curve 103

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3.12 Good fitting Learning Curve 103

3.13 Adam optimizer Training - Hamming loss per training epoch 105

3.14 SGD optimizer Training - Hamming loss per training epoch 105

3.15 RMSprop optimizer Training - Hamming loss per training epoch 105

3.16 Adam optimizer Training - Loss per training epoch 107

3.17 SGD optimizer Training - Loss per training epoch 107

3.18 RMSprop optimizer Training - Loss per training epoch 107

3.19 Bias-Variance trade-off | Bullseye diagram Source: BiasVariance [22] 109

3.20 Performance Learning curves - Binary accuracy per training epoch 114

3.21 Optimization Learning curves - Loss per training epoch 114

3.22 Neural Network Structure | English Google News 130Gb corpus with 20 dim & without OOV token 118

3.23 Neural Network Structure | English Google News 130Gb corpus with 20 dim & with OOV token 120

3.24 Neural Network Structure | English Google News 7B corpus with 50 dim & with OOV token 123

3.25 Custom Keras model - Hamming loss learning curves 130

3.26 GloVe embeddings - Hamming loss learning curves 130

3.27 English Google News 20dim 130GB no OOV - Hamming loss learning curves 131

3.28 English Google News 20dim 130GB with OOV - Hamming loss learning curves 131

3.29 English Google News 50dim 7B with OOV - Hamming loss learning curves . 131

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3.30 Custom Keras model - Loss learning curves 132

3.31 GloVe embeddings - Loss learning curves 132

3.32 English Google News 20dim 130GB no OOV - Loss learning curves 133

3.33 English Google News 20dim 130GB with OOV - Loss learning curves 133

3.34 English Google News 50dim 7B with OOV - Loss learning curves 133

4.1 Overview of how the weights are updated during a training epoch 138

4.2 Basic flow for intent matching and responding to the end-user Source: Dialogflow-Intent Overview 151

4.3 The implementation of the chatbot on Dialogflow 152

4.4 Question 1 162

4.5 Question 2 163

4.6 Question 3 164

4.7 Question 4 165

4.8 Question 5 166

4.9 Question 6 167

4.10 Question 7 168

4.11 Question 8 168

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

List of Tables

3.1 Summary table of training the custom neural network (Hamming Loss - F1 score performance metrics) 77

3.2 XOR operations 84

3.3 Adam Optimization function Results - Hamming loss performance metric (best model highlighted) 88

3.4 SGD Optimization function Results - Hamming loss performance metric (best model highlighted) 89

3.5 RMSprop Optimization function Results - Hamming loss performance metric (best model highlighted) 89

3.6 Summary table of the best model selected per optimizer (applying the first step of the comparison plan) 90

3.7 Adam optimizer Confusion Matrix 94

3.8 SGD optimizer Confusion Matrix 95

3.9 RMSprop optimizer Confusion Matrix 95

3.10 Movie Predictions (20 random movies) | Adam optimizer 98

3.11 Movie Predictions (20 random movies) | SGD optimizer 99

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3.12 Movie Predictions (20 random movies) | RMSprop optimizer 100

3.13 Summary table of training the custom neural network with binary accuracy . 113

3.14 Scoring table of the model estimator trained on binary accuracy metric 113

3.15 Trained Neural Networks on pre-trained GloVe word embeddings 116

3.16 Summary table of English Google News 20dim without OOV token 119

3.17 Trained Neural Networks on pre-trained English Google news 20dim without OOV token 119

3.18 Summary table of English Google News 20dim with OOV token 121

3.19 Trained Neural Networks on pre-trained English Google news corpus 20dim with OOV token 121

3.20 Summary table of English Google News 50dim 7B with OOV token 123

3.21 Trained Neural Networks on pre-trained English Google news 50dim 7B corpus with OOV token 124

3.22 Scoring table of the selected model classifiers 126

3.23 Confusion matrix - model 1 (Custom Keras model) 127

3.24 Confusion matrix - model 2 (GloVe embeddings) 127

3.25 Confusion matrix - model 3 (English Google News 20dim 130GB no OOV) . 128

3.26 Confusion matrix - model 4 (English Google News 20dim 130GB with OOV) 128

3.27 Confusion matrix - model 5 (English Google News 50dim 7B with OOV) . . 129

A.1 Abbreviations 179

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter One

Introduction

1.1 Summary

The aim of this post-graduate thesis titled "Movie Recommendation engine using an interactive conversational agent" is to give solution to a multi-genre text classification problem for movie recommendation using techniques from the field of Natural Language Processing and deep learning. Furthermore, the authors of the thesis will introduce a unique selection plan to identify and evaluate a well-performed mathematical model, developed to predict multi-label classification targets. Moreover, the thesis will document the development of an interactive conversational agent using the trained weights of the most accurate model classifier. Based on the output of the neural network and the user's text input, a custom-made recommendation algorithm will propose movies to outdo similar movie recommender engines currently in the e-market. The thesis is composed of five chapters with each one to play an essential role in achieving the goal mentioned in the first sentence. The first chapter describes the main scope and the problems tackled by the thesis. The second chapter is focused on data extraction, data cleaning, and data preparation. The concepts presented in the second chapter constitute a differentiation factor that could substantially affect a neural network's performance and behavior. The goal of the second chapter is to provide a thorough presentation and

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

description of the critical concepts to clean and prepare data coming from text and going to train a multi-label text classification model. The third chapter presents the multi-genre movie classification problem built to make accurate predictions of a movie's gene sequence, given its movie content. A movie genre sequence is an array of more than one genre linked to a specific movie (e.g., [action, romance, comedy]). To deal with the multi-genre movie classification problem, the authors deployed five different neural networks. The first neural network is a custom-made model classifier build explicitly from scratch. The other four classifiers are pre-trained and saved classifiers, either using pre-trained word embeddings (e.g., GloVe embeddings) or using a whole pre-trained model structure saved in Tensorflow Hub community. The selection plan to select the best model classifier among the five options includes various metrics and methodologies from the machine learning and deep learning field that will assess each model estimator's predictions, and the most accurate one will be used to build the recommendation algorithm. The fourth chapter will use the model estimator with the most accurate predictions to create a recommendation algorithm operated to propose movies to the end-user via an interactive conversational agent on Facebook Messenger. Additionally, in the fourth chapter, the authors will introduce an opinion-based questionnaire to compare the interactive conversational agent with two similar well-established services. The questionnaire assesses the three services on three topics. The first topic assesses the quality of the recommendations proposed by each one of the conversational agents. The second topic evaluates the sufficiency of their recommendations, the agents' UI, and the quality of the human-computer interaction with the user. Finally, the third topic evaluates the features offered by the chatbots. The questionnaire's goal is to estimate the quality experience and the movie recommendations, each chatbot provides to the end-user. The fifth and final chapter describes the conclusions and re-cups the results of the third and fourth chapters; the chapters conducted the experimentation and the research.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

1.2 Thesis Scope

Before the Internet's disruptive era, people could only purchase items available to physical stores. Contrary to the past, today, customers can transact among an abundance of different items and choices. They will feel more secure and confident to make new purchases when services and applications consider their personal needs and preferences. Recommender systems have been invented to save time and allow users to filter the content they view according to their preferences. Recommender systems nowadays exist for commercial goods (e.g., clothing, FMCG's, electronics), movies, music, places, and reservations. The recommendation engine can suggest content or products that better match the needs of a particular user by gathering relevant information.

The types of recommendation systems are many; however, the two most popular categories are the collaborative filtering systems and content-based systems. The former fits very well in services and applications where users interact with each other and have formed a source of connections and relationships. The collaborative filtering system considers the preferences of two inter-connected users and recommends to a third user a similar content only if this user has similar preferences to one of the two connected users. The latter system provides recommendations only when it has collected enough content and feedback from the end-user to provide an accurate recommendation. The recommendation algorithm developed in the thesis belongs to the second type of recommender systems.

The content-based movie recommender system is the focus of this thesis. The cause of irrelevant recommendations in such systems is mainly the lack of content. The thesis goal is to develop an original movie recommendation engine that will exploit movie content provided by users and online sources. The thesis will introduce two components to accomplish this goal. For the first component, the authors researched the fields of Natural Language Processing and deep learning to build a unique multi-genre movie classification algorithm that will predict

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

the movie's multiple genres based on movie content. The second component is a competitive movie recommender engine that will use both the first component's predictions and the user's feedback to propose relevant movie recommendations that could outperform those proposed by already well-established and functional applications of the e-market.

The reader of the thesis will get a well-explained presentation on the following topics:

- The preparation plan to transform online data of textual movie content destined to train a multi-genre movie classification model.
- The development of five model classifiers, one custom-made and four pre-trained, to accurately predict a movie's genre sequence based on movie content.
- The original selection plan to assess the predictive power and the performance of multi-genre movie classification algorithms.
- The approach to use the most accurate neural network's trainable weights to create a recommendation algorithm. Alternatively, the steps to move from development of a neural network to the production of a business application.
- The development of a user-friendly conversational agent accessed by Facebook Messenger.
- The opinion-based questionnaire that will evaluate the recommendations and the functionality of the custom-made conversational agent to other applications currently in the market.

Each of the topics mentioned above is both theoretically and technically justified, so the reader can get familiar with the potential NLP applications of neural networks and witness an end-to-end journey of developing an idea and transitioning to a productive and user-friendly application.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

1.3 Thesis Outline

The two main scopes of the thesis are the development and selection of a multi-genre movie classification model explicitly using movie content and the creation of a competitive content-based movie recommender system. In general, text classification experiments consist of three different categories based on the output of the target variable:

- Binary text classification: The target variable has only two different values. For example, a sentiment analysis algorithm tries to predict whether a user review is positive or negative. Alternatively, when a cybersecurity algorithm tries to identify if an email sent is malware/ spam or a valid one.
- Multi-class text classification: The target variable has a broader range of values rather than only two. For example, a user review is angry, toxic, enthusiastic, informative, or judgemental. The target variable is assigned to only one of these values.
- Multi-label text classification: The target variable is a sequence of more than one value that identifies the independent variables. For example, a user review is toxic and at the same time angry based on the words written by the user.

The thesis's target experiment belongs to the third category, which is the most trickier and, at the same time, the most challenging category out of the three mentioned above. In the third category, a mathematical model should correctly classify text content to more than one target variable, or in other words, multi-output target variables. The metrics used to evaluate the multi-label text classification algorithms are different from those used for binary classification, like the model's accuracy, which is one of the most common performance metrics of model performance. Hamming loss is a more suitable metric to derive proper conclusions about the performance of a multi-label text classification model.

The second scope of the thesis, as stated in the first sentence, is the documentation of using

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

the output of a machine learning algorithm, trained to classify movies to their correct genre sequence, to create an interactive conversational agent that will recommend movies similar to those preferred by the user. More specifically, the second scope is to introduce a methodology of exploiting the output of a trained model to create a user-friendly application. Neural networks developed to offer many more capabilities than just correctly classifying a movie to its correct genre sequence.

Three approaches exist to deal with multi-label text classification problems. The first approach is by using a pre-developed algorithm such as Support Vector Machines (SVM), Decision Trees, Random Forest, and XGBoost. The second and third approaches include the development of neural networks, either with custom training layers or pre-trained layers. The critical factor in choosing between the first approach and the other two is the volume of data. Since the movie content used for training in the current experiment derives from around 50,000 movies, the last two approaches are more suitable. After the use of the second and the third approach, the thesis documentation compares the results achieved with the custom trained neural network to the results achieved with the pre-trained saved neural network approach. The authors make the comparison through a thorough and unique selection plan, which incorporates evaluation metrics and scores, learning curves, bias-variance tradeoff, confusion matrices, and classification reports.

To fulfill the second scope's purpose, the weights of the best model classifier will help create a movie recommendation mechanism. More specifically, to identify similar movies, the word embedding vectors of a movie are the key. The latter corresponds to a vector of numbers used to map the content of movies. Utilizing the cosine similarity measure, the authors could quickly identify similar movies based on their word embeddings representation. The cosine similarity measure and the original recommendation algorithm are the two crucial and necessary components to develop an interactive conversational agent.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

As already mentioned above, the multi-label text classification experiments are trickier than the binary or multi-class classification problems. Moreover, the most typical tutorials that exist online usually tackle binary classification problems because the new audience quickly understands them. The thesis will address the multi-label text classification approach and present efficient ways to estimate the performance of algorithms built to predict text data of multi-label target variables. Additionally, it is common to create and train successfully machine learning algorithms that can predict the results of an action; however, many of those attempts stop in the development phase. Not few are the cases of many exceptional projects that cannot find a commercial implementation to include their results. Thus, one additional challenge of the thesis is to present an alternative and efficient approach of making proper use of an algorithm's output to create a practical and functional business application.

Since two are the thesis's main goals, two are the approaches to tackle each challenge; two are also the results derived and discussed upon each different challenge. The first discussion addresses a comparison between the results achieved from a custom-made neural-network to those results achieved by the pre-trained saved algorithms. Select the best model estimator out of those compared and make a statement if the results of the custom-developed model built by the authors of the thesis outperform the pre-trained saved models. The second discussion address an opinion-based comparison between the interactive conversational agent built in the context of the thesis with two similar recommendation agents. The end-users of the three application agents will fill in a questionnaire, answering questions related to the quality of the recommendations, the graphical user interface of the recommendation agent, and the applications' overall functionality.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

1.4 Python code outline

Apart from the thesis document, the project includes a well-established library of python code displayed using the Jupyter Notebook IDE. The Python notebooks delivered along with the thesis document, follow a specific structure, and are separated into different parts. The separation of each part is similar to a pipeline where the output of an element is the input to the workflow's next element.

The accompanying python notebooks are:

1. Part 1: Concatenate data from different sources and Download URLs
    - **Description:** This is the starting notebook that creates the raw dataset to be later used in the rest of the notebooks.
    - **Input 1**: Online data sources of movie data
    - **Input 2**: Tabular dataframes containing the movie information extracted from the subsequent parts (part 1.1-1.12).
    - **Output**: The final tabular dataframe with the total number of movies and their respective information column-separated.
        (a) Part 1.1: Extract data - 5000 movies
        (b) Part 1.2: Extract data - 10000 movies
        (c) Part 1.3: Extract data - 15000 movies
        (d) Part 1.4: Extract data - 20000 movies
        (e) Part 1.5: Extract data - 25000 movies
        (f) Part 1.6: Extract data - 30000 movies
        (g) Part 1.7: Extract data - 35000 movies



(h) Part 1.8: Extract data - 40000 movies

(i) Part 1.9: Extract data - 45000 movies

(j) Part 1.10: Extract data - 50000 movies

(k) Part 1.11: Extract data - 55000 movies

(l) Part 1.12: Extract data - 58098 movies

- **Description:** Each subsequent part make use of specific pythonic modules to extract data from online digital sources in batches of 5000 movies. More information in the second chapter.

2. Part 2: Transforming and Cleaning the columns of the final dataset

   - **Description:** Since the most information extracted is the raw text from online URLs, they will inevitably contain "noise." Text's noise includes unnecessary punctuation, text, or any other form of character that may spoil the context reasoning of a sentence. Additionally, the notebook presents data needed transformations in a manner that will benefit the use of it in deep learning alike algorithms.
   - **Input**: The final tabular data frame with approximately 58098 movies from part 1.
   - **Output**: The cleaned and transformed dataset to serve as input in Part 3.

3. Part 3.1: Data Tokenization

   - **Description:** Any text data inserted into any deep learning or machine learning algorithm should first get appropriately transformed because mathematical models can understand numeric characters and not text characters.
   - **Input:** The data frame of Part 2
   - **Output:** Data in the appropriate format inserted into the algorithms.



4. Part 3.2: Deep Neural Networks

    - **Description:** The notebooks composing this part present the training and evaluation of five different neural network models. The evaluation is related to each model estimator's ability to predict the movie genres of a movie accurately. The first model trained is a custom neural network model, while the other four models are already pre-trained models using either pre-trained embeddings or using the network structure of a model stored in the TensorFlow Hub community.
    - **Input:** Data preprocessed from part 3.1
    - **Output 1:** The model trained.
    - **Output 2:** The weights and the structure of the model. More information about word embeddings and the model outputs in chapter 4.
    - **Output 3:** Model history in tabular format.

5. Part 3.2.1: Compare the different models trained on part 3.2 for the model 1

    - **Description:** The first model, a custom-trained neural-network, uses three different optimization algorithms to train and update the loss function and the weights of the model. Thus, out of the three models, one per optimizer, only one is selected and will proceed to the next part. The criteria to select the best-performed model include scoring metrics, classification reports, accurate prediction of the movie genre sequence of 20 random movies, the fitted learning curves of the performance and optimization metrics, and the bias-variance tradeoff
    - **Input:** Model estimators trained on part 3.2 related only to the custom made neural network
    - **Output 1:** The best model estimator out of the three optimizers (Adam, SGD, RMSprop). Only one optimizer can train the best model classifier.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

6. Part 3.3: Model Comparisons

    - **Description:** The notebook compares the results of the five trained model classifiers, and the model estimator that meets most of the criteria is the one selected. The selection criteria for the best-performed model include a shorter version of the criteria presented in part 3.2.1.
    - **Input:** The five models trained in parts 3.2-1 - 3.2-5
    - **Output:** The model that will satisfy most of the criteria defined apriori.

7. Part 4: Load the chosen models and extract the word embeddings

    - **Description:** In this part, it is described the process of extracting the most valuable asset of a model trained on textual data. The so-called word embeddings, the more effective representation of text in mathematical terms so far. The word embeddings will be used by the recommendation algorithm to identify the similarities between the movies.
    - **Input:** The best model classifier selected from part 3.3
    - **Output 1:** The word embeddings of the best model
    - **Output 2:** The final tabular dataframe with only the relevant movie content utilized by the recommendation algorithm.

8. Part 5: Content-Based Movie Recommendation algorithm

    - **Description:** The development and testing of the algorithm that will recommend the most relevant movies based on the user preferences. The custom algorithm is built based on some specific steps, which structure a complete approach to movie recommendation.
    - **Input 1:** The dataset of part 4.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- **Input 2:** The saved word embeddings for all the movies.
- **Output:** The four movie recommendations to the user.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter Two

Data Text Analysis

2.1 Introduction

Data analysis is collecting, inspecting, transforming, and modeling raw data to discover useful information, informing conclusions, and supporting the creation of the ideal input used in predictive or decision-making models. It is essential that at the final step of the data analysis, the input created will not get trapped into the concept of Garbage In, Garbage Out (GIGO). A concept in computer science which states that nonsense input data results to nonsense output. Data analysis [42] breaks into the following stages, data requirements, data collection, data processing, and data cleaning. The third chapter will thoroughly cover and justify the stages mentioned above tailored around text data analysis since the data used for training NLP related neural networks entirely consists of text data. The first stage, **data requirements** answer the question "What data is the appropriate input for my text classification problem?", "The kind of data that I need to answer my question" and "Where to find this data?". The second stage **data collection** refers to the techniques used to extract, store, and maintain the data. **Data prepossessing**[1] is the most essential concepts, when dealing with NLP classification tasks,

[1]The meaning and the use of data preprocessing are adjusted to the needs of the current NLP experiment. The reader should not take their application for granted to any tasks.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

because it is consist of steps that prepare the text data appropriately before proceeding to deep learning model algorithms. Data preprocessing consists of two parts.

The first part, described as data cleaning or with the term substitution, is built upon the fact that the raw data should be cleaned and correctly preprocessed of any noise before used as inputs to a prediction model. Noise for textual data, is mainly text input that spoils the meaning of a sentence (e.g., HTML tags of online source, symbols, punctuation Part of this stage is also the feature engineering, creating new features out of the current ones. Data cleaning arises from the fact that the data input could substantially affect the final result of a predictive model. Once the data is extracted, it may include inconsistencies, noise, duplicates, errors, or even missing values. Thus, the cleaning of numerical and categorical data is critical to correct any of the inconsistencies mentioned earlier. The cleaning of categorical text data is even more critical, because of the many errors found to the raw text of words.

The second part, which completes the data preprocessing phase, is described as NLP text preprocessing. It involves techniques that will transform the cleaned textual data into a proper form for text analytic algorithms. Algorithms that deal with tasks such as text analysis, text classification, sentiment analysis, question answering methods, or even language translation models, they require the data to be appropriately transformed. It is common knowledge that computers cannot understand human language but instead numbers or, more precisely, mathematical expressions. Data extraction, data collection and text preprocessing techniques are the main discussion subject of this chapter. Techniques that will introduce to the reader the proper way of collecting and preprocessing textual data, destined to train deep neural prediction models.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

2.2 Data Requirements

As already have been mentioned from the first chapter, one of the problems the thesis is trying to solve is to research and develop different deep learning classification models for the creation of a movie recommender engine that will outperform movie recommenders currently in the market. The second problem is to accurately predict the sequence of movie genres of a movie based explicitly on movie content. The recommender system follows the content-based approach instead of its counterpart, the collaborative filtering approach. Given that, the reader can easily conclude that the features and variables would be data related to movies and feedback written by the users in the form of reviews or numerical rating. Movie content data such as the movies' plot summary, the movie's cast, the title of the movie, and the directors of the movie are some concrete examples that can directly distinguish a movie from other movies, and it is what a viewer would search for when looking for a movie to watch at his/her free time. What also a viewer would search is the reviews of a movie, if they express a positive or a negative opinion, but also the ratings the movie received from other users who have watched it in the past. There is only one official and credible source to find all those types of data, and this is the website of IMDB (Internet Movie Database)[2]. The IMDB website is an up-to-date source of data related to movies, TV-series, and TV-shows. The data available in IMDB can be either text (plot summary, director and cast names, reviews, etc.) or numerical like IMDB rating, the number of users rated the movie, year of release, etc. Both the movie recommendation algorithm and the NLP neural model deployed for the movie text classification are using data found explicitly on the main movie content website, and the website where all the user reviews are written. Even though the data was easily accessible, the access to a large pool of movie content and user reviews wasn't fictitious, since no other structured database of movies already existed. Thus, it was for sure that the data had to be

[2] https://www.imdb.com/

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

extracted and collected. The next section will enlighten the reader with the techniques and sources used in data collection.

Once the authors defined the requirements of what type of data should be collected to solve the problem, they have to preserve the number one rule of data analytics, and that is the data quality. To ensure that the data collected is of high quality, the authors strictly defined and followed the rules below: *The title of each rule is found on Wikipedia's page about data quality [43]*

1. Validity: The degree to which the data conform to defined business rules or constraints. Below there is a set of some particular rules that should be followed to ensure that the data collected is valid.

    - Data-Type Constraints - values in a particular column must be of particular data type.

        (a) **Movie Title:** Text Data of type "string" with the official title when the movie was published.

        (b) **Genres:** Text Data of type "string" of the official movie genres accompanying the official movie publish.

        (c) **User Rating:** Numerical data of type "float".

        (d) **IMDB rating:** Numerical data of type "float".

        (e) **Actors:** Text Data of type "string".

        (f) **Director names:** Text Data of type "string".

        (g) **User Reviews:** Text Data of type "string".

    - Range Constraints - numerical data should fall within a certain range.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

(a) **Movie Title:** Any textual data[3] or phrase in the range from 1 to n. With N to symbolize a logical number of words. For example, a movie title cannot contain 25 words.

(b) **Genres:** A sequence of words from 1 to 3.

(c) **User Rating:** In range from 0.5 to 5.

(d) **IMDB rating:** In range from 0 to 10.

(e) **Actors:** A sequence of names in the range from 1 to n. With N to symbolize the maximum number of actors in a movie.

(f) **Director names:** A sequence of names in the range from 1 to n. With N to symbolize the maximum number of directors in a movie.

(g) **User Reviews:** An array of N number of textual content. Where N is the number of reviews found in the relative web page of a movie before clicking the button *Load more*.

- Missing-value Constraints: No column should contain missing data. The above six columns and any other column crafted from those six columns should not contain any missing value. The more the information gathered for a movie, the better the help to the model estimator in understanding the difference between the movies of different genres. Thus, the experiment cannot tolerate missing information.

- Unique Constraints or constraint of duplicate data: Data in some columns cannot be present twice.

  (a) **Movie Title:** This column is the only column that falls into this rule. The final dataset should not contain the same movie twice. An exception is those

[3]Textual data is defined as any type of text such as words, human names, or symbols.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

movies with the same title but different cast and plot.

- Set-Membership constraints: The values of a column must follow a specific pattern to be considered as valid members.

  (a) **Movie Title:** Any kind of phrase consisting of one or more words based on the range constraint.

  (b) **Genres:** Any word belonging to the family of the official movie genres ('Drama', 'Comedy', 'Action', 'Romance', 'Thriller', 'Horror', 'Crime', 'Documentary', 'Adventure', 'Sci-Fi', 'Mystery', 'Children', 'Animation', 'Fantasy', 'War', 'Western', 'Musical', 'Family', 'Biography', 'Music', 'History').

  (c) **User Rating:** No textual data is allowed.

  (d) **IMDB rating:** No textual data is allowed.

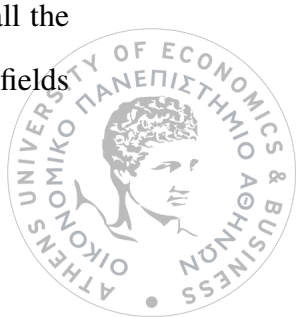
  (e) **Actors:** No numerical data is allowed. Actor first and last name or nickname.

  (f) **Director names:** No numerical data is allowed. Director first and last name or nickname.

  (g) **User Reviews:** Sentence of more than one word.

2. Accuracy: The degree of conformity of a measure to a standard or a true value. Access to an external source that follows the grand truth can sometimes secure the accuracy of the data. In the current experiment, the use of the IMDB database, an official place of movie content, can re-assure that the data collected from that source are valid and accurate.

3. Completeness: The degree to which all required measures are known. Based on the scope of the experiment, the movie recommender system should understand and filter information about the movie. This kind of information should be available for all the movies in the dataset. Movies that are missing one or more of the above seven fields



(movie title, genres, user rating, IMDB rating, cast, director(s) and user reviews) will be omitted.

4. Consistency: The degree to which a set of measures is equivalent across systems—data extraction from only one online source.

5. Uniformity: The data should possess the same format when shared across different systems. Since movie content is retrieved from only one source, the IMDB database, the uniformity is also a rule not violated.

The rules mentioned above are constructed to ensure the quality and the validity of the data. A simple algorithm can outweigh a complex one just because it was trained on enough and high-quality data. Quality can beat the most sophisticated algorithms [20].

2.3 Data Collection

2.3.1 IMDB Website and Grouplens dataset

This part will thoroughly present the steps followed in collecting, and extracting the data in their raw format before any data transformation applied. Since the data source for collecting movie features and information for a single movie would be the movie's online page on IMDB, the authors noticed that a unique code identified each different web page. This code is composed of two letters ("tt") and a sequence of numbers with length from 1 to 7 (e.g., tt1234567). Thus, by collecting this unique id number per movie, the IMDB web page was easily accessible via an HTTP protocol request. The next step is to find a list with a variety of such unique ids for a movie, since the manual collection and storing of approximately 50,000 unique ids was out of scope. Grouplens [26] is one of the many online databases storing movie features for research and development projects. The data downloaded from

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

this website includes 27,000,000 ratings for 58,000 movies, last updated on 09/2018. The movie data organized in 5 different CSV files *(the information presented below is based on the attached README.md file of the grouplens folder)*:

1. Genom-scores.csv: The tag genome is a data structure that contains tag relevance scores for movies. The tag genome encodes how strongly movies exhibit particular properties represented by tags (atmospheric, thought-provoking, and realistic). This file, although downloaded, was not used.

2. Genom-tags.csv: The tag descriptions for the tag IDs in the genome file. Similarly, as the previous file, this one was also not used further in our research.

3. links.csv: The file contains identifiers used to link to other sources of movie data.

    - movieId is an identifier for movies used by https://movielens.org. E.g., the movie Skyfall has the link https://movielens.org/movies/1.

    - imdbId is an identifier for movies used by http://www.imdb.com. E.g., the movie Skyfall has the link http://www.imdb.com/title/tt0114709/.

    - tmdbId is an identifier for movies used by https://www.themoviedb.org. E.g., the movie Skyfall has the link https://www.themoviedb.org/movie/862.

4. movies.csv: Movie information such as the movie id, the title, and the movie genre is part of the file. This CSV file is also one of the most important files provided by MovieLens and used for the thesis since it contains the respective genre and title for each of the 58,000 movies.

5. ratings.csv: The movie rating is a 5-star scale value between 0.5 stars to 5.0 stars. A CSV file that depicts viewers' positive and negative popularity over a movie. The more the number of stars, the more popular a movie is.

The authors of the thesis project used the following CSV files:



- links.csv
- movies.csv
- ratings.csv

Each of the three csv datasets has a different number of rows. To concatenate them, the authors should bring them to several rows equal to the number of movies in the dataset. The only data frame with 58,098 number of rows is the "movies.csv" containing the title and the genre(s) of each movie. The "links.csv" can be directly concatenated with the "movies.csv" dataset to include the online URLs related to each movie. As already noted, the URLs are a direct root to the movie's main web page on the IMDB website. On the contrary, to handle the information in the "ratings.csv" the authors grouped the relative table. To achieve the grouping, the authors calculated the average rating per movie, including in such way the twenty-seven million ratings given for all the 58,098 movies.
The concatenated dataset has 58,098 rows and the following columns:

1. title: The movie title.
2. genres: The movie genres (1 or more).
3. rating: The average user rating (0.5-5).
4. IMDB URL: The online URL rooting to the main web page of the movie on IMDB.
5. synopsis URL: The online URL rooting to the synopsis web page of the movie.
6. reviews URL: The online URL rooting to the user reviews of the movie.

2.3.2 Data Extraction - Web Scrapping

It is essential to mention the need to create three different types of URL (IMDB, synopsis, reviews). The **imdb url** contains information related to the content of a movie. The movie's

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

content is any text relevant to the plot summary, the cast, the plot summary, the directors, and the IMDB rating of the movie. The **synopsis url** provides access to a well-established context of summarizing each movie's story-line with more information to take into consideration, contrary to the short text of the plot summary. Finally, **reviews url** provides a very different kind of information about a movie. Apart from references to the actors and the movie's plot, reviews can create a positive or a negative impact on the movie itself. IMDB rating depicts this impact.

With the three URLs ready, the next step is to download the online HTML document linked to each URL. Web scraping is the extraction of data found on web pages. Web scraping is the automatic mining of data or information collected from the World Wide Web. The web scraping involves fetching data from a web page and then extract the data from it. Fetching is downloading a web page, which is actually what a web browser does, to store the HTML document locally used for processing. Python language offers programming modules, such as the library "requests", to fetch/download data from online sources. When the fetching completes, first parse the data, and then the data extraction process can start. Parsing [45] in computer science is the process of creating a parser, a software component that takes input text data and builds a data structure. Some abstract syntax tree gives a structural representation of the input while checking for correct syntax for HTML and XML documents.

The document presents the *How To* implementation of the data extraction and collection. The authors extracted the information in batches of 5,000 movies. The extraction of 58,098 movies completed in batches due to computing resources restriction (lack of enough temporary RAM). As a result, the 58,098 rows separated into 11 batches of 5,000 rows each, and one batch of 3,098 rows. The procedure to extract an HTML document from the three URLs is the following:

The document below mentions technical terms and python packages.

Step 1: Download the URL for each movie found in columns "imdb url", "synopsis url" and



"reviews url". For the download of each URL, the python module requests[4] used. The output of this process is the HTML document of each URL. An HTML document contains all the information needed to contrast the web pages written in programming languages like HTML[5], JavaScript or CSS.

Step 2: Save the three HTML documents per movie into serialized files with the suffix pickle. Definition by the Hitchhiker's Guide to Python [32]: *Data serialization converts structured data into a format to allow sharing, storing, and recovery of the data to their original format. Moreover, serialization minimizes the data size and reduces the risk of space and bandwidth requirements.* The data serialization in python is available through the module "pickle" or "joblib". Based on the Python documentation, the pickle module [23] implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process of converting a Python object hierarchy into a byte stream, and "unpickling" is the inverse operation, converting a byte stream (from a binary file or bytes-like object) back into an object hierarchy. Pickling (and unpickling) is alternatively known as "serialization", "marshalling" or "flattening". Similar to pickle, the joblib [34] module embrace two critical features. Transparent and fast disk-caching of output value and fast compressed persistence of data. Each batch of 5,000 movies consumes about 2-3 GB's of memory on a local hard drive.

Step 3: Parse the three HTML documents per movie to begin the web-scraping. Parsing HTML documents is easily achievable by making use of the python module Beautiful Soup. The latter is a famous HTML parser.

Step 4: Having the data parsed; the next step is to iterate over each HTML document and

[4]Requests is a Python HTTP library, licensed by the ApachURLicense 2.0., aiming to make HTTP requests simpler and more human-friendly

[5]Hypertext Markup Language (HTML) is the standard markup language for documents designed for display in a web browser. Web browsers receive HTML documents from a web server or local storage and render the documents into multimedia web pages [44]

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

extract the desired information about each movie. The fields extracted are the plot summary, the actors, the director(s), the IMDB rating, the plot synopsis, and the user reviews.

Step 5: Delete the movies that had at least one missing field. The final dataset should not include any missing text information, so only those movies with information to all the six earlier mentioned fields included.

Step 6: Create the final dataset by appending all the batches of 5,000 movies. The final dataset has 49,393 movies, less than 58,098, due to missing movie-content information per batch.

Having the data extracted and appended to a final dataset, the next and final step to the data text analysis pipeline is the Data Preprocessing.

2.4 Data Preprocessing

2.4.1 Data Cleaning

Data cleaning is an important set of actions in the field of Machine Learning and Artificial Intelligence. The process of data cleaning aims to identify incorrect, incomplete, inaccurate, or irrelevant data and modify, replace, or delete data to achieve the goal of the experiment. Cleaning text data is the fundamental starting point towards a great model instead of a biased model classifier. Textual data may contain online tags, punctuation, symbols, parts of speech, or Unicode characters that add noise to the corpus and lead to irrelevant or wrong decisions made by the model classifier. In machine learning, irrelevant or error-prone data lead to an incorrect model building [15].

Cleaning the collected movie data before they transformed into an input is suitable for natural language processing experiments. Below are the cleaning steps identified as appropriate to clean from inconsistencies the data collected from the IMDB database.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- **Cleaning step 1:** Clean the year (i.e 2001) from each title.

  Purpose: Common users will never reference a movie title by typing the release year next to the title.

- **Cleaning step 2:** Change the position of words (, The), (, A), (, An) & (, Les) to the front of the title.

  Purpose: For example, the movie The Batman Begins, was written as "Batman Begins, The" which is a mistake during the dataset creation by GroupLens.

- **Cleaning step 3:** Clean the column of plot summary and user reviews off punctuation and strip any extra space from the text. This is a starting cleaning step. The next chapter implements a thorougher plan of text cleaning.

  Purpose: The importance of this step will be highlighted in the next sub-section of data transformation. Briefly, punctuation adds nothing but noise to the textual data. Such noise does not help at all a mathematical algorithm to understand the meaning of the sentence. With punctuation, the same word may appear with a comma and with an exclamation mark. While it is the same word written, an algorithm will conceive it as two different words. This conceptual error can spoil the results of language processing models. Cleaning punctuation in a sentence is vital.

- **Cleaning step 4:** Remove movies with no data in the columns: plot summary, and user reviews.

  Purpose: As already mentioned, missing values is a ground rule for data quality and a strict rule for this experiment. Thus, movies with values such as "Add a plot" and an empty user reviews array omitted from the dataset.

- **Cleaning step 5:** Remove duplicate movies found in the movie's title column.

  Purpose: As already mentioned, duplicate values in the relative column is a ground

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

rule for data quality and a strict rule for this experiment.

In addition to data cleaning, feature engineering is a type of data analytics process that could amplify an algorithm's ability to make useful predictions. A more sophisticated definition of feature engineering found online [41] says *Feature engineering uses domain knowledge to extract features from raw data. Features assembled can improve the performance of machine learning algorithms.* By joining the columns of the movie title, actors, directors, plot summary and genres the authors created an additional column of text data. Finally, the authors created a sentiment value related to each movie. The sentiment value depicts the positive or negative sentiment of the user over a specific movie. The value of the rating column (provided by Grouplens) is used to assemble the sentiment value column. The sentiment is not used for the training of the model classifier. Instead, it will be used in the fourth chapter by the recommendation algorithm to filter the movies recommended to the user.

Having the data cleaned and assured that they strictly follow the data quality rules mentioned earlier in the chapter, it is time to preprocess the data accordingly to create input variables suitable for a Natural Language Processing algorithm.

2.4.2 NLP Text Preprocessing

Introduction to Text Preprocessing

Data text preprocessing aims at making the raw data at hand more amenable to neural networks [5]. It is a sub-field of the Natural Language Processing domain that aims to form of text into a more digestible input so that a machine learning algorithm can perform better than random. Text preprocessing documents the steps to transform the text data columns into a predictable and analyzable format. A format suited to solve a language processing task. *Mathematical algorithms find it challenging to analyze raw context data, and because of that, the format*

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

of the data has to change. All inputs and targets in a neural network must be tensors of floating-point data (or tensors of integers). Data vectorization turns the data, either text or image, into tensors. Later the document will thoroughly explain and demonstrate how a word sentence will turn to a sequence of tokens and from there into a sequence of tensors of float data.

In general, many approaches exist to implement the text preprocessing phase. However, as Matthew Mayo mentioned on his article[6], the concept of data text preprocessing is summarized in three main components:

- Tokenization,
- Normalization,
- Substitution

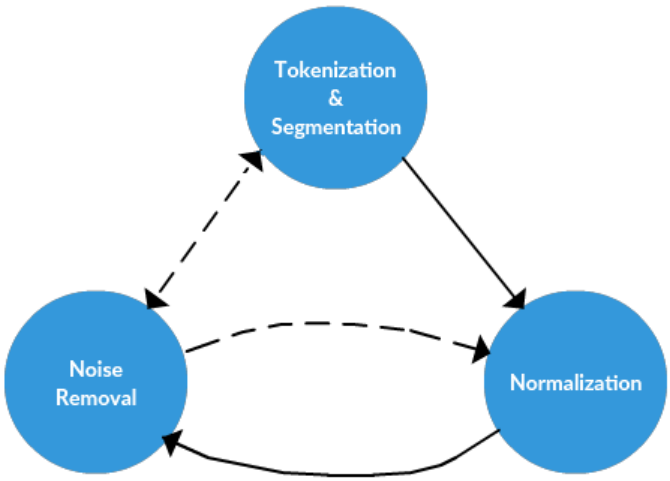


Figure 2.1 Text data preprocessing framework
Source: Text Preprocessing Framework [13]

Tokenization refers to the replacement of a sentence by its words/ tokens. For example, the sentence "Nikos love machine learning" is replaced by a list of tokens "[Nikos, love,

[6]https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

machine, learning]." Those tokens should be replaced by a sequence of float numbers since a mathematical model can only understand numbers instead of words. Python modules like TensorFlow[7] (developed by Google) include methods to tokenize a given corpus of sentences automatically.

Normalization refers to a series of related tasks meant to put all text on the same level. Converting text to either upper or lower case, getting rid of punctuation, converting numbers to their word equivalents, and vice versa, and removing the stop words are some of the techniques implemented to give a uniform shape to the text data. Based on the framework, depicted in the image above, the normalization approach follows three steps: (1) Stemming, (2) Lemmatization, (3) a group of text cleansing tasks. Stemming and Lemmatization are case-sensitive approaches, and someone should implement only one of the two upon the text data. The choice of which approach to follow depends on the experiment itself, the volume of data, the length of sentences per feature, and the final prediction results achieved. Whereas, the third step, a group of cleansing tasks, consist of cleaning tasks suitable to every text data, after excluding experiments of sentiment analysis where the existence of numbers is essential for the experiment.

Substitution is the third and final cycle of the framework. It is related to steps responsible for removing the unnecessary parts of a text data originated from the data source. Different data sources demand different types of cleaning. For example, when downloading an HTML or XML document, the text would inevitably contain tags not relevant to the text. Is is critical to remove those unnecessary tags from the text. In similar scenarios, decode JSON format data to string format to exploit the JSON data efficiently. The Substitution step can occur either before or after the above two steps of Tokenization and Normalization.

[7]TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Apart from the text preprocessing done before any machine learning model that deals with textual data, there is another form of preprocessing that is mandatory and applies to any form of data either textual, numerical, image, or categorical. This prepossessing boils down to the machine learning pipeline's essence, and it is essential for any problem. Variations and different concepts of this preprocessing can significantly affect the predictions of an artificially intelligent model. The basic concepts of this preprocessing path are the following:

- Split the dataset into two parts, training and test.
- When the dataset is imbalanced, apply re-balancing techniques.

Having introduced the text preprocessing and the parts that assemble this procedure, it is time to document and present the application of those terms on the movie content of the current experiment. After the application of the preprocessing tasks, the final number of movies is equal to 48,991[8] movies.

Normalization

In the chapter's introduction, the documentation mentioned that three elements characterize the Normalization procedure. Lemmatization, Stemming, and a group of supplementary actions. Stemming and Lemmatization are text normalization (or sometimes called word normalization) techniques in the field of Natural Language Processing used to prepare text, words, and documents for further processing[9] [28].

Lemmatization

The breakthrough question(s), being the origin of building both Lemmatization and Stemming techniques, is briefly the following: "How can someone filter a sentence by omitting the words that grammatically provide the same meaning? How to make a computer pay attention to a

[8]The initial number of 58,098 downgraded due to the cleaning that has been applied and documented

[9]https://www.datacamp.com/community/tutorials/stemming-lemmatization-python

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

wider range of different words?" The example below presents the need to answer those two questions.

Sentence 1: People are buying goods online, because the price of a good on the Internet is cheaper.

$$\text{are, is} \rightarrow \text{be,}$$

$$\text{goods, good} \rightarrow \text{goods}$$

Based on the underlined words of the sentence above, the pair of words (are, is) and (goods, good) hide the same effect on the sentence's meaning. By finding the origin word format for each pair, the authors can increase the ability of a model to better understand the sequence of the words above and effectively focus on other words.

A definition written by Stanford Edu states that *Lemmatization refers to the use of vocabulary and morphological analysis of words, purposed to remove inflectional endings and the lemma, the base or dictionary form of a word.* In grammar, **inflection** is the grammatical modification of a word to different grammatical levels, for example, tense, case, voice, viewpoint, individual, number, sexual orientation, and state of mind. An emphasis communicates at least one syntactic class with a prefix, postfix or infix, or another inner alteration, for example, a vowel change, for reference see here [40].

In the example above, the words (are, is) will be both replaced by their lemma, the word (be). By grouping the different inflected forms of a word, so they get analyzed as a single item, a mathematical model's probability to differentiate this sentence from a similar one exponentially increases. Besides, the ultimate goal of a predictive model that learns from the text is to find the semantics and the different meanings among sentences. The word "be" is such a common and frequently used word that adds no difference in the meaning of a sentence whatsoever.

In the experiment developed the Lemmatization of the textual columns: (a) plot summary,

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

(b) movie content, and (c) user reviews, implemented using the python module NLTK and more specifically the method WordNetLemmatizer(). Natural Language Toolkit (NLTK) is the number one platform for developing Python programs to operate with human language content. It offers easy-to-use interfaces to over 40 corpora and lexical sources such as WordNet and a set of text processing modules for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries [31]. Most of the actions undertaken for preprocessing textual data use the NLTK package.

Before continuing to stemming, it is important to mention that the columns meticulously preprocessed are the five input columns of the multi-genre model classifier. The characterization input refers to those columns that explicitly offer the corpus of text data to accurately learn and predict the correct genre assigned to a given movie. Those columns are the following five: **(a) actors**, **(b) plot summary**, **(c) movie features**[10], **(d) user reviews** and **(e) movie title**.

Stemming

Stemming is eliminating affixes (suffixes, prefixes, infixes, circumfixes) from a word to obtain a word stem. Stemming refers to a crude heuristic process that chops off the end of words, and often includes removing derivational affixes[11].

Sentence 2: All social interactions are forbidden but running.

$$\text{forbidden} \rightarrow \text{forbid},$$

$$\text{running} \rightarrow \text{run}$$

In general, Stemming is much of a simpler approach than Lemmatization. A word stem need not be a similar root as a word reference-based morphological root (lemma) but rather is just an equal or smaller form of the same word. Stemming algorithms are typically rule-based. A

[10]The feature engineered column that includes the information found in columns (title, actors, plot summary, director names, genres)

[11]Removing suffixes from a word called Suffix Stripping

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

heuristic procedure that cuts off the finishes of words. A word is looked at and go through a progression of conditionals that decide how to chop it down, see here [27] for reference.

Even though being a more heuristic rather than sophisticated approach like Lemmatization, stemming is not selected as the approach of text preprocessing due to the following two reasons (a) over-stemming, (b) under-stemming, as explained accurately in this article. *Over-stemming* occurs when cutting off a high proportion of a word. Over-stemming can result in a single letter or too nonsensical stems. Alternatively, it can result in words being resolved to the same stems, even though they probably should not be. For example, if someone examines the words *astronomy, astronomical, astronomies, astronomer*, a stemming algorithm that resolves these four words to the stem word "astronom" has fallen into the trap of over-stemming. A better resolution might have the words (astronomer, astronomical) and (astronomy, astronomies) stemmed together. *Under-stemming* is the opposite event. It happens when several words are the forms of one another. The proper solution would be to resolve to the same stem, although actually, they do not. By example, the words *data, datum, date* stem-like,

$$\text{data} \rightarrow \text{dat},$$

$$\text{datum} \rightarrow \text{datu},$$

$$\text{date} \rightarrow \text{dat},$$

Even though the stemming algorithm performs poorly, it is clear that a machine learning model will perform deficiently since the words date and data have a completely different meaning. Selecting the Stemming over the Lemmatization, and vice versa, to preprocess data can drastically affect a model's classifier results. It is advisable when choosing one of the two to check the performance of a prediction model and the data itself. Moreover, other languages even if they seem somewhat related, have drastically different results with Stemming and Lemmatization.

Group of additional normalization tasks



Additional normalization tasks represent the third component of implementing normalization preprocessing to textual data, and it consists of actions that are optional and non-applicable to all textual data. Thus, the experimentation is also an important step as it was with the application of Lemmatization or Stemming. The additional preprocessing actions that transformed the textual data of this experiment are the following:

- Convert Accented Characters to ASCII

  Example: If a French user made a review for an IMDB movie, then it's highly probable that words like *cafe* and *latte* would be written like *café* and *latté*. An NLP classification algorithm would have treated them as totally different words, even though they refer to the same thing.

  Python implementation: The unidecode module transforms words into their ASCII form. The function unidecode() takes Unicode data to represent it in ASCII characters (e.g., the universally displayable characters between 0x00 and 0x7F), chosen to match what a human with a US keyboard would choose.

- Expand Contractions

  Example: Contractions are shortened words, e.g., I'm and haven't. Stretching such words to "I am" and "have not" helps to standardize/ normalize the text.

  Python implementation: A custom function that transforms several contractions into their decontracted form.

- Remove punctuation

  Example: *Nikos loves training deep learning models!*. Splitting this sentence to word-level the result is ("Nikos", "love", "training", "deep", "learning", "models!") or ("Nikos", "love", "training", "deep", "learning", "models", "!"). In either case, the exclamation mark is a part of speech and has nothing more to offer to the sentence's meaning. Apart from enhancing the meaning of the sentence, punctuation oppose an

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

extra hurdle to text preprocessing and NLP model training.

Python implementation: The python module *string* has the method *punctuation* connected to a dictionary of different punctuation symbols.

- Strip white spaces

  Remove consecutive word brakes to avoid turning them into tokens.

- Treatment for Numbers

  Example: *The food on that restaurant was terrible even on my* ***third*** *visit* Transform the word in bold format to its numeric representation and remove it from the sentence, since as a numeric word has nothing to offer in the meaning of the sentence. Additionally, remove dates and years from the text. Even though this might be an effective transformation when dealing with a sentiment analysis experiment, numeric data may not play an important role in the model's performance to distinguish a positive from a negative review. However, if the NLP task is to extract the number of tickets ordered in a message to a chatbot, removing the numbers is unnecessary. Either way, the current experiment transformed the numbers to their numeric format and then removed them from the text along with the dates.

  Python implementation: Use the Python module *word2number*[12].

- Remove stop-words

  Example: *Natural Language Processing* ***is a*** *popular field* ***of*** *Artificial Intelligence.* The words highlighted as bold are stop-words. Stop-words are common words that usually connect sentences or phrases, start or end sentences, or ensure a grammatically correct sentence. Words like "we" and "are" probably do not help in NLP tasks such as sentiment analysis or text classifications. Thus, remove those words to save computing time and effort in processing large volumes of text.

[12]Python module to convert number words (e.g., twenty-one) to numeric digits (21). It works for positive numbers up to the range of 999,999,999,999 (e.g., billions).

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Python implementation: Use the Python module *text*.

- Lowercase the words

  Example: ***The*** *text classification is one of* ***the*** *problems solved with NLP models.* The final standardized transformation implemented on the textual movie data is often a default one, with the user to choose between uppercase or lowercase letters. Apply this normalization step to handle words like "The" and "the" highlighted in the example sentence above. A mathematical model will treat those words differently since their written form is not the same, even though they refer to the same word. Not handling similar issues, might lead to a decline in the model's classification predictive power.

  Python implementation: Build-in python method *lower()*.

Concluding, in the current experiment of text classification, the textual data of column(s): (a) actors and (e) movie title transformed based on the following steps of normalization:

- Lowercase the words

, the textual data of column(s): (b) plot summary and (c) movie content transformed based on the following steps of normalization:

- Convert Accented Characters to ASCII
- Expand Contractions
- Remove punctuation
- Strip white spaces
- Treatment for Numbers
- Remove stop-words
- Lowercase the words

and finally, the textual data of column(s): (d) user reviews transformed based on the following



steps of normalization:

- Convert Accented Characters to ASCII
- Expand Contractions
- Remove punctuation
- Strip white spaces
- Remove stop-words
- Lowercase the words

The authors did not apply the treatment of numbers for column of user reviews, because many were the cases of incorrectly separated words written by the user. Thus, the word2number algorithm confused those words as numbers and could not provide an appropriate numeric replacement.

Follow the example below to view a demonstration of the normalization steps applied on the five columns of the first movie in the dataset "Toy Story":
Column 1: Actors column for the movie Toy Story **before** the normalization:
['Tom Hanks', 'Tim Allen', 'Don Rickles', 'Jim Varney', 'Wallace Shawn', 'John Ratzenberger', 'Annie Potts', 'John Morris', 'Erik von Detten', 'Laurie Metcalf', 'R. Lee Ermey', 'Sarah Freeman', 'Penn Jillette', 'Jack Angel', 'Spencer Aste']

Actors column for the movie Toy Story **after** the normalization:

'tom hanks,tim allen,don rickles,jim varney,wallace shawn,john ratzenberger,annie potts,john morris,erik von detten,laurie metcalf,r. lee ermey,sarah freeman,penn jillette,jack angel,spencer aste'

Column 2: Plot Summary column for the movie Toy Story **before** the normalization:

"A cowboy doll is profoundly threatened and jealous when a new spaceman figure supplants

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

him as top toy in a boy's room."

Plot Summary column for the movie Toy Story **after** the normalization:

"cowboy doll profoundly threaten jealous new spaceman figure supplants toy boy room"

The cleaning algorithm removed stop-words like **is**, **and**. Moreover, it replaced words like **threatened** by their lemma, which is, in this case, the word **threaten**. Last but not least, the algorithm removed any form of punctuation.

Column 3: Movie Features column for the movie Toy Story **before** the normalization:

'Toy Story Tom Hanks Tim Allen Don Rickles Jim Varney Wallace Shawn John Ratzenberger Annie Potts John Morris Erik von Detten Laurie Metcalf R. Lee Ermey Sarah Freeman Penn Jillette Jack Angel Spencer Aste John Lasseter A cowboy doll is profoundly threatened and jealous when a new spaceman figure supplants him as top toy in a boy's room. Adventure Animation Children'

Movie Features column for the movie Toy Story **after** the normalization:

'toy story tom hanks tim allen don rickles jim varney wallace shawn john ratzenberger annie potts john morris erik von detten laurie metcalf r lee ermey sarah freeman penn jillette jack angel spencer aste john lasseter cowboy doll profoundly threaten jealous new spaceman figure supplants toy boy room adventure animation children'

Column 4: User Reviews column for the movie Toy Story **before** the normalization:

"Andy's toys live a reasonable life of fun and peace, their only worries are birthdays and Christmases, when new toys could easily replace those already there. One such birthday Andy's top toy, Woody the cowboy, finds himself in direct competition with Andy's new Buzz Lightyear doll. When rivalries boil over Woody tries to hide Buzz down the side of the bed but accidentally pushes him out the window, the other tops expel Woody, and he leaves with no choice but to find Buzz and return him to the house. But with only two days before Andy

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

moves house, time is of the essence. Given how often the same mix of animation, wit, jokes and kids humour has been used since Toy Story (Ice Age, Monsters Inc, Bugs Life) it is easy to forget how refreshing it was when it first came out. I have just watched it again and it is dating a little in comparison to more recent twists on the formula. It seems each one has to be sharper and have more references etc in the background. However it is still very funny and deserves praise for being the first of a successful formula. The plot is simple but effective and actually has genuine drama and excitement to it. The main story is fun but the degree of character development is what really shores it up. The conflict between Buzz and Woody is taken deeper than this and, when confronted by the truth of his status as a toy, Buzz's turmoil is very real as opposed to him being a cartoon character and nothing more. Despite the two strong leads there is a real depth in the support cast. They may not actually have that many lines, but they have all the funniest lines. Most of the 'adult' wit comes from the Potato Head, dinosaur, the pig and slinky dog. They are funny and are very well used. In fact the majority of this humour and plot will go right over kids heads. Looking back on it, I do feel a cynical edge on it in so much as this film must really have helped sales of the toy companies in the film. It's hard not to see the marketing department standing behind this film rubbing their hands. However the actual product is so wonderfully fun that I forgot this quickly. The voice work is excellent and the characters match the actors. Hanks is good as Woody and Allen has a good B-movie type voice for Buzz. Varney, Ratzenberger, Ermey (doing his usual), Rickles and others are all really good in the support roles and, probably, come out as the favourite characters for adults. Overall this is a classic film that will appeal to adults as much as to kids (if not more). A good plot and a really sharp script make the already short running time fly by. The only downside is that your kids will want you to go out and buy the damn things!"

User Reviews column for the movie Toy Story **after** the normalization:

"andy toy live reasonable life fun peace worry birthdays christmases new toy easily replace birthday andy toy woody cowboy find direct competition andy new buzz lightyear doll rivalry

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

boil woody try hide buzz bed accidentally push window top expel woody leave choice buzz return house day andy move house time essence given mix animation wit joke kid humour use toy story ice age monsters bugs life easy forget refresh come just watch date little comparison recent twist formula sharper reference background funny deserves praise successful formula plot simple effective actually genuine drama excitement main story fun degree character development really shore conflict buzz woody take deeper confront truth status toy buzz turmoil real opposed cartoon character despite strong lead real depth support cast actually line funniest line adult wit come potato head dinosaur pig slinky dog funny use fact majority humour plot right kid head looking feel cynical edge film really help sale toy company film hard marketing department stand film rub hand actual product wonderfully fun forget quickly voice work excellent character match actor hanks good woody allen good b movie type voice buzz varney ratzenberger ermey do usual rickles really good support role probably come favourite character adult overall classic film appeal adult kid good plot really sharp script make short running time fly downside kid want buy damn thing".

Column 5: Movie Title column for the movie Toy Story **before** the normalization:

"Toy Story (1995)"

Movie Title column for the movie Toy Story **after** the normalization:

"toy story"

Tokenization & Segmentation

Tokenization is the third and final step of the NLP text preprocessing framework. Tokenization is a word directly connected to the term data vectorization. Having the data cleaned off noises and normalized/ standardized (referring to the first and the second element of the framework presented in figure 2.1), the final step is to vectorize the textual data before starting to insert the data to a mathematical model. Data vectorization, transforming a text corpus to a sequence

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

of float or integer points called tensors, consists of the following steps:

Step 1: Tokenize the data

TensorFlow and Keras offer the text tokenization utility class to automate a text corpus's vectorization process, turning each text into a sequence of integer numbers with each integer being the index of a token in a dictionary. The Tokenizer class consists of six arguments, the definition of which plays an important role in the input data that will be handled by the mathematical model. More specifically, the arguments are [13]:

- num_words: The maximum number of words to keep, based on word frequency. Only the most common (num_words-1) words kept. The specific argument is the most determinant one out of the six. It specifies the actual number of words/tokens used as part of the vocabulary built upon the corpus of each of the five columns (actors, plot, movie features, user reviews, movie title). For example, from the 260,000 actors in all 48,991 movies, the 95% most frequent actors have been kept to build the actor's column corpus vocabulary. The authors dropped the rest 5% of the actor names. It is advisable not to assign to the *num_words* argument, the total length of the words in the text. Because then the tokenizer will saturate with plenty of words. Moreover, it is also not advisable to use only one percent of the whole text because then the algorithm will not have enough vocabulary to make proper classifications.
- filters: Each element is a character filtered from the text. The default is all punctuation, plus tabs and line breaks, minus the apostrophe character. Specify this argument to the punctuation that splits the corpus of sentences (e.g. ',' or ' ').
- lower: Boolean. Whether or not to convert text to lowercase. An argument that may be left to its default value False if the data is already standardized.

[13] based on the Keras documentation at https://keras.io/preprocessing/text/

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- split: A string symbol. Separator for word splitting (e.g. ’,’). Set this value to space if the words in the corpus are separated by space or comma if commas separate the data. Any type of punctuation or word suitable to split words into individual tokens.
- char_level: If True, treat every character as a token. This argument separates a corpus of sentences not only by word but also by character per word.
- oov_token: if given, it will be added to word_index and used to replace out-of-vocabulary words during text_to_sequence calls. A second determinant argument that specifies the index of the rest of the words/tokens that have been thrown out by the num_words arguments. The 5% of the most uncommon words, even though discarded, should have a unique token to take part in the text classification experiment.

Implementation on the current experiment: Having explained each argument of the tokenizer class meticulously, it is time to present their settlement on the current experiment of movie classification. Deploy as many tokenizers as the number of textual columns.

- num_words: Apply a word frequency table to accurately specify the number of words per column, using the Python method CountVectorizer[14] [35] of the Scikit-learn module. CountVectorizer converts a collection of text documents into a matrix of token counts. With CountVectorizer the authors could easily specify for example the number of actors played in more than 2 movies, or how many movie plots contain the word "obnoxious".

  Column 1: Actors

  Example: [(’christopher lee’, 153), (’john carradine’, 130), (’john wayne’, 119),

[14] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

('clarence nash', 115), ('gérard depardieu', 110), ('michael caine', 109), ('donald sutherland', 104), ('eric roberts', 102), ('anthony quinn', 100), ('robert de niro', 99)], Christofer Lee is the actor who played in the most films of all the actors in the dataset.

Summary numbers: 262,252 actors in total. 84,204 of them played in more than two movies. Thus, 178,048 played in only one movie.

Argument value: 249,140 actors. The 249,140 (95.0%) most frequent actors have been selected, the rest 13112 (5.0%) assigned to the Out-of-Vocabulary token.

Column 2: Plot Summary

Example: [('young', 5472), ('life', 5469), ('man', 4489), ('woman', 4203), ('year', 3646), ('love', 3480), ('new', 3426), ('family', 3287), ('story', 3247), ('world', 2787), ('girl', 2696), ('old', 2604), ('friend', 2593)], "young" is the most frequent word in the dataset, present in the plot summaries of 5,472 movies.

Summary numbers: 43,623 words/tokens in total. 21,725 of them used in the plot summaries of more than two movies. Thus, 21,898 are written in the plot summary of only one movie.

Argument value: 41,442 words. The 41,442 (95.0%) most frequent words have been selected, the rest 2,181 (5.0%) assigned to the Out-of-Vocabulary token.

Column 3: Movie Features

Example: [('drama', 23177), ('comedy', 15025), ('john', 12853), ('michael', 9916), ('david', 8690), ('robert', 7731), ('action', 7120), ('james', 7107), ('romance', 6908), ('young', 6846), ('thriller', 6462), ('richard', 6035), ('paul', 5752), ('crime', 5738)], "drama" is the most frequent word in the dataset, present in the movie features of 23,177 movies. Drama is also the most dominant movie genre in the dataset.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Summary numbers: 169,274 words/tokens in total. 84,816 of them used in the movie features column of more than two movies. Thus, 84,458 are written in the movie features of only one movie.

Argument value: 160,811 words. The 160,811 (95.0%) most frequent words have been selected, the rest 8,463 (5.0%) assigned to the Out-of-Vocabulary token.

Column 4: User Reviews

Example: [[('film', 128949), ('movie', 94127), ('do', 44028), ('like', 42321), ('time', 35331), ('story', 34224), ('character', 32963), ('make', 31545), ('good', 31179), ('just', 30482), ('watch', 23713), ('scene', 22601), ('play', 22466), ('great', 21534)], "film" is the most frequent word in the dataset, present in the movie reviews of 128,949 movies.

Summary numbers: 126,042 words/tokens in total. 62,953 of them used in the movie reviews of more than two movies. Thus, 63,089 written in the movie reviews of only one movie.

Argument value: 119,740 words. The 119,740 (95.0%) most frequent words have been selected, the rest 6,302 (5.0%) assigned to the Out-of-Vocabulary token.

Column 5: Movie Title

Example: [[('the', 15713), ('of', 5243), ('of', 44028), ('a', 2409), ('in', 1796), ('and', 1732), ('to', 1125), ('love', 700), ('man', 675)], "the" is the most frequent word in the dataset, present in the movie title of 15,713 movies.

Summary numbers: 27,749 words/tokens in total. 9,996 of them used in the movie reviews of more than two movies. Thus, 17,753 written in the movie reviews of only one movie.

Argument value: 27,749 words. The 27,749 (95.0%) most frequent words have

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

been selected, the rest 1,387 (5.0%) assigned to the Out-of-Vocabulary token.

- filters: The authors specified this argument to the element that splits the sentences into. Set to space for the columns: plot summary, movie features, movie reviews, and movie title. Set the respective argument to comma for the actors' column.
- lower: Specify the argument to True for each of the five columns. Even though the text is already lowercased, to cover any uppercase incident, the argument took the positive boolean value.
- split: For the cast column, this value is set to "," (comma without the following space), whereas for the rest of the four columns the split argument is set to " " (empty space).

Actor names preprocessed before splitting

"tom hanks$_{\bigcirc}$,tim allen$_{\bigcirc}$,don rickles$_{\bigcirc}$,jim varney,wallace shawn,john ratzenberger,annie potts,john morris,erik von detten,laurie metcalf,r. lee ermey,sarah freeman,penn jillette,jack angel,spencer aste"

↓

Actor names into tokens after splitting on (",")

['tom hanks', 'tim allen', 'don rickles', 'jim varney', 'wallace shawn', 'john ratzenberger', 'annie potts', 'john morris', 'erik von detten', 'laurie metcalf', 'r. lee ermey', 'sarah freeman', 'penn jillette', 'jack angel', 'spencer aste']

Plot Summary preprocessed before splitting

"cowboy$_{\bigcirc}$doll$_{\bigcirc}$profoundly threaten jealous new spaceman figure supplants toy boy room"

↓

Plot Summary word tokens after splitting on (" ") space

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

['cowboy', 'doll', 'profoundly', 'threaten', 'jealous', 'new', 'spaceman', 'figure', 'supplants', 'toy', 'boy', 'room']

The same split string value applied to the column of the plot summary is also applied to the columns of movie features, user reviews, and movie titles.

- char_level: False, characters not treated as tokens.
- oov_token: Set Out-of-Vocabulary token to "<OOV>" for each of the five columns.

Apply the **Tokenizer class** to each of the five cleaned and prepared columns. When calling the Tokenizer class upon the preprocessed data, the result is an object used for the following purposes:

- Get the word index for each token
- Transform the textual data to a sequence of integers based on the word index of both the tokens present in the vocabulary and the OOV token index.

Step 2: Get the **Word Index** for each token

Word index is a dictionary type object that maps words in the vocabulary to their numeric representation, a mapping essential for sequence encoding.

Word index representation of column Actors

'sean bean': 323, 'john ireland': 324, 'michael lerner': 325, 'cloris leachman': 326, 'ciarán hinds': 327, 'kevin corrigan': 328, "vincent d'onofrio": 329,

with the keys to represent the actor name and the values the assigned word index.

Word index representation of column plot symmary

'<OOV>': 20001, 'life': 2, 'young': 3, 'man': 4, 'woman': 5, 'year': 6, 'love': 7, 'new': 8, 'family': 9, 'story': 10, 'world': 11

The out-of-vocabulary token has also its index.

Thus, the vocabulary built for each of the five columns has a total length equal to the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

num_words set for each column plus one word corresponding to the OOV token.

Step 3: Transform textual data to a **sequence of integers**

Next, having created the corpus vocabulary with the most frequent words per column and assigned those tokens to a unique word index, it is time to replace the actual words with their index. For example, the word index 323 will replace every occurrence of the actor name *Sean Bean* found in the dataset. Similarly, replace the word *life* with the word index 2, every time found in a movie's plot summary.

Sequence of Integers representation for column actors

Random movie one:

[2024, 3228, 451, 1615, 124, 7888, 9046, 4817, 3229, 3919, 1150, 6131, 18118, 249141, 18119]

Random movie two:

[3230, 7889, 12357, 9047, 1616, 2643, 4344, 3231, 5439, 14759, 249141]

If an actor name has a word index in the vocabulary, replace the actor's name with the index range from 1 to 249140. If the name is not present in the vocabulary then replace it by the OOV token of index 249140+1 as seen in both random movies one & two.

The interesting thing, also necessary to reference at this point, and directly connects to the rest of the steps remaining to finish the tokenization cycle, is the length of the sequence per movie. In the example above, the two sequences have different lengths. The first sequence has 15 integers, whereas the second sequence has only 11. The reasonable question raised based on that difference is, how the different lengths might cause a problem. The answer hides on the concept of how a deep learning model works and expects the input data. Neural networks training expects to receive the data on batches of same length dense vectors. For example, if in a random experiment, the number of sample data was 1050, and the specified batch size was 100, then it would take approximately 1050/100 = 10 batches to train the model. In other words, ten

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

different samples of the data used to train the model. By default, a batch requires the data samples per input layer to have the same length; otherwise, the training procedure stops. Moreover, the input sequences should be of the same shape. Padding and masking are two techniques used to resolve the different sequence length of the input data.

Step 4: Padding and Masking the sequences

Padding is the process of balancing the sequences to the same length. TensorFlow and Keras offer the *pad_sequences* method to automate the process of padding. This method transforms an array of sequences (lists of integers) into a 2D Numpy array of shape (num_samples, num_timesteps). Where num_samples is the total number of samples available, whereas num_timesteps equal to a given value or the longest sequence otherwise. The arguments of the *pad_sequences* method are the following:

- sequences: List of lists, where each element is a sequence. In the current experiment, the sequences of integers of the columns actors, plot summary, movie features, user reviews, and movie title.
- maxlen: Int, the maximum length of all sequences.
- dtype: Type of the output sequences. Argument not specified in the current classification experiment.
- padding: String value equal to 'pre' or 'post'. Pad either before or after each sequence. Set to 'post' for the current classification experiment.
- truncating: String value equal to 'pre' or 'post'. Remove values from sequences larger than maxlen, either at the beginning or at the end of the sequences.
- value: Float or String value, which represents the padding value shown when the sequence's length is shorter than the maxlen argument. Not specified in the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

current classification experiment.

Apply padding following one of the two methods:

- Method 1: **Expand** all sequence lengths to the length of the highest sequence in the dataset.
- Method 2: **Truncate** all sequence lengths to the length of smaller sequences.

Among the two, the authors selected the first method. Based on the first method, the authors calculated each sequence's maximum length in the dataset and selected for maximum length the 90% most frequent length. When proceeding with the first approach, the main concern is the word index inserted to the sequences with a length smaller than the maximum length. The additional integers resulted from the expansion of the sequence are assigned to number zero.

More specifically the maxlen parameter per column is:

- Actors: Length of the sequences in the 90 percentile of the column is equal to **15** actors per sequence. Thus, sequences that include less that 15 word indexes will be expanded to 15 integers.

  Example: [2024, 3228, 451, 1615, 124, 7888, 9046, 4817, 3229, 3919, 1150, 6131, 18118, 0, 0]

  The sequence above has two additional 0 integer values indicating the sequence's successful expansion to 15 from 13 integers.

- Plot summary: Length of the sequences in the 90 percentile of the column is equal to **22** words per sequence. Thus, sequences that include less than 22 word indexes will be expanded to 22 integers.

  Example: [4, 216, 8, 1459, 112, 130, 209, 22, 9329, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

  The above plot summary sequence had a length significantly less than the 90

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

percentile of cases in the plot summary column. Thus, it should include many more 0 integers to balance with the rest of the plot summaries in the dataset.

- Movie features: Length of the sequences in the 90 percentile of the column is equal to **60** words per sequence. Thus, sequences that include less than 60 word indexes will be expanded to 60 integers.

  Example: [1656, 723, 780, 368, 12405, 162, 1589, 124, 1753, 136, 4508, 811, 222, 13, 11422, 573, 7655, 91, 566, 197, 5274, 222, 4, 190, 159, 732, 384, 765, 2817, 27, 424, 4, 1314, 99, 7656, 17, 667, 32, 3779, 344, 444, 640, 83, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

- User reviews: Length of the sequences in the 90 percentile of the column is equal to **328** words per sequence. Thus, sequences that include less than 328 word indexes will be expanded to 328 integers.

  Example: [2, 5723, 184, 48, 18, 177, 1138, 11246, 8717, 7065, 5269, 8718, 13360, 9814, 323, 228, 16652, 2, 142, 178, 955, 20001, 230, 18, 76, 336, 1522, 304, 2831, 10377, 6684, 733, 475, 6084, 2, 249, 10864, 4867, 61, 2, 24, 201, 5, 2495, 842, 861, 2463, 2463, 56, 566, 13649, 621, 4650, 3609, 475, 570, 1800, 5021, 261, 621, 20001, 24, 5, 11, 503, 61, 433, 16198, 12814, 182, 44, 1882, 3738, 367, 6874, 1101, 2390, 3723, 44, 5360, 330, 115, 1565, 4839, 24, 5, 4811, 19003, 1565, 4839, 14, 1480, 20001, 95, 87, 5334, 455, 44, 2770, 56, 419, 1642, 4248, 20001, 1058, 2296, 202, 495, 595, 224, 202, 4296, 2, 595, 224, 142, 87, 4675, 4445, 5487, 18, 566, 1062, 4651, 4580, 3043, 10212, 11050, 1730, 16199, 857, 190, 27, 307, 3214, 224, 42, 1315, 1448, 4867, 29, 160, 503, 805, 4, 499, 409, 1786, 384, 236, 20001, 2679, 1382, 2875, 247, 95, 612, 1642, 4248, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

0, 0, 0,

2.4.3 Prepare the input for deep learning experimentation

Note: This part creates the primary input for the next chapter where the training and prediction of different neural networks occur.

Data preprocessing is close to the end. Separation of input data into different sample for model experimentation is the third and final component of the data preprocessing. Neural networks and any kind of machine learning model is trained upon a sample dataset and then tested upon a different sample dataset. As a result, the second part of NLP Text Preprocessing will focus on splitting the data preprocessed to samples capable of training and testing a deep learning model. But before proceeding to what applied in the current classification experiment; first, start by explaining the importance of splitting the data into different samples for training and testing a predictive model classifier.

Data separation into different samples affects machine learning models' evaluation, a fundamental concept for both machine & deep learning fields. Someone could ask why not to evaluate a neural network on the same data initially trained. Because when a third user will use the same model to make predictions on data never addressed before would result in severe predictions and mistakes made by the model. A situation often called *overfit*. Overfit can happen when a machine-learning algorithm continues to correctly classify data that has already learned but will hustle to classify data never seen before. A model that is overfitting is capable of making classifications very close to the training data, which already seen, but will be very inaccurate on new data points.

Overfitting is the main hurdle that prevents a mathematical model trained for classification to achieve its original goal, a model that can *generalize*. Model generalization is the ability of a model to perform well on data never seen before. As a result, separating the data into different samples, one for training and one for testing the model, would potentially minimize the chance of getting a model that overfits. Although two different samples, one for training

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

the model and one for testing it, are enough to minimize the risk of overfitting? The answer is no. When developing a model always involve the process of tuning its hyper-parameters. Hyper-parameters are parameters such as the number of layers in a neural network or the batch size of a layer for parallel data training. The hyper-parameters should not be confused with the parameters of a neural network, such as its weights. The hyper-parameter tuning occurs when monitoring the model's performance on classifying data found on a third sample called *validation* data.

The validation sample originates from the training sample data points and is used to optimize the algorithm's learning. Optimize the learning process by changing the configuration of some parameter space. However, overfit can still exist when tuning the model's performance on the validation set, even though the model was never trained on it but rather evaluated. Root to this problem is the *information leaking*. Whenever a hyper-parameter of the model is tuned based on the model's performance on the validation set, information about the validation data leaks into the model. Executing the experiment many times and for many different hyper-parameters, a significant amount of information about the validation dataset leaks to the model. The information leakage will result in a model that performs extremely well on the data sample used to optimize the model in the first place. From that event, finds origin the need to have a third independent sample of totally new data to test the model's performance. The model should not have access to information about that data sample at all. If anything about the model has been tuned based on test dataset performance, then the model's measure of generalization will be flawed.

In conclusion, it is essential to separate the preprocessed dataset into three parts: training sample, validation sample, and test sample. Apart from using the validation sample for hyper-parameter tuning and performance optimization, it is common practice to use validation data to evaluate different models and use the never-seen-before test data to test the model's ability to generalize.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

It is important to specify the values of two critical components, before sampling the data points into train, validation and test sets.

Component 1: Split Ratio

Specify the split ratio. The splitting ratio will determine how much of the total data allocated for training and testing. Next, select the validation data points out of the training set. There are different combos of split ratios, like 70% training and 30% test data, 80% - 20%, or even 50%-50%. Typically, 50-50 is the least probable pair of values to select, and it fits situations where there is a high volume of extremely balanced data. In the current classification experiment, the number selected is 80% training and 20% test data points. In chapter three, from the 80% of data points, 30% is selected for validation and 70% for training. There are different methods to select a validation dataset out of the train set. The three most common approaches are (a) Out-of-sample validation, (b) K-fold validation, and (c) iterative k-fold validation. The current experiment proceeds with approach (a).

Component 2: How to split the data points

The second component is the split method of the data points. This component is very critical to the performance of a **classification** algorithm. One method recommends that the first N number of data points will go for training and the rest for testing. However, this approach has one and great throwback. Assuming the creation of a mathematical model that can identify the number shown in an image and the moment has come to test its performance to derive correct classifications. The *independent* features are values like the pixels, the width of the image, and the height of the image. The *dependent* variable, or usually called the target variable, is a single digit from 0 to 10. The dataset has one-thousand numbers ordered by digit and has ten occurrences of each digit. Splitting the dataset into 70% training and 30% test samples, will result to a training sample of 700 numbers from digits 0 to 7, and a test sample of 300 numbers from digits 8 to 10. This approach will lead the model to overfit; it will accurately



predict digits from 0 to 7, but will never accurately predict the digits from 8 to 10. Both training and test sets should contain all the different labels that exist in the dataset. Thus, it is essential to shuffle the data and then perform the split, the so-called random shuffle split.

However, what is the consequences when the dataset is imbalanced and someone applies random shuffle split. A dataset is characterized as imbalanced if the number of data points per label differs significantly. The example above is quite the opposite of an imbalance dataset because, as mentioned, each digit label has ten observation points. As a result, when dealing specifically with classification experiments, either they belong to binary classification, multi-class classification, or multi-label classification, the researcher should handle one more component. This component can make a significant difference when the dataset is imbalanced, rather than when the data is balanced. The component addresses the number of data points per label allocated between the train and the test set. The **percentage** of observations per label in the training sample has to be equal to the **percentage** of observations in the test sample. More precisely, if label A is present in the 20% of the total observations in the training sample, then the label A should also represent the 20% of the total observations in the test sample. The so-called **stratified shuffle split** was applied to address the issue of inaccurate classifications in imbalanced datasets.

In conclusion, the two issues addressed when splitting data for classification experiments is not only the number of data points (split ratio) allocated to each of the training and test samples but also the percentage per label to be equally present to both the training and the test sets (stratified shuffle split) in case of imbalanced datasets.

Having explained the concepts that can affect the data separation into train, validation and test samples, it is time to present the application of those approaches on the data of the current multi-label classification experiment.

(Concept 1) Split ratio selected: 80% training sample and 20% test sample.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Training sample: 39,192 movies for training.

Test sample: 9,799 movies for testing.

Both the training and the test samples are containing the *independent* and the *target* variable.

Independent variables (X): Actors, Plot summary, Movie Features, User Reviews and Movie Title.

dtype of independent variables: 2D Numpy arrays of integer numbers of shape (number_samples, maxlen).

Dependent variables (y): Movie genres.

dtype of dependent variable: 2D Array of genres of shape (number_of_samples, number_of_genres).

(Concept 2) Stratified shuffle split: The stratified shuffle split method is selected instead of the opposite approach, random shuffle split. Because the data is imbalanced, the distribution of the data points that belong to a specific label is not uniform. So to make sure that the algorithm will be trained upon all the labels and will not favor the dominant ones[15], the percentage allocation of genres between the training and the test samples should be identical to the percentage of the labels in the initial dataset. See below the results of the stratify split.

Label allocation among the movies

The figure above illustrates the percentages of movies belonging to each genre. They are total **17** distinct genres in the dataset. Needless to denote that GroupLens included movies from twenty-eight distinct genres, although the authors removed eleven of them because they were assigned to movies belonging to the 1% of the total dataset. The distribution shown in the figure should be the same in the training and test samples, respectively. Oppositely, applying random shuffle split, it would be highly probable to include none of the least popular movies

[15]The labels with the highest percentage of data points

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Figure 2.2 Percentage of genres per movie

Drama	25.382
Comedy	15.932
Action	7.581
Romance	7.210
Thriller	7.018
Horror	5.737
Crime	5.423
Documentary	4.356
Adventure	4.281
Sci-Fi	3.008
Mystery	2.721
Children	2.512
Animation	2.368
Fantasy	2.290
War	1.685
Western	1.350
Musical	1.146

in one of the samples.

Stratified shuffle split

Figure 2.3 Stratified shuffle split on training & test samples

(a) Training sample

Drama	25.375
Comedy	15.926
Action	7.587
Romance	7.216
Thriller	7.019
Horror	5.738
Crime	5.425
Documentary	4.359
Adventure	4.279
Sci-Fi	3.008
Mystery	2.722
Children	2.508
Animation	2.370
Fantasy	2.293
War	1.685
Western	1.348
Musical	1.142

(b) Test sample

Drama	25.410
Comedy	15.952
Action	7.554
Romance	7.186
Thriller	7.013
Horror	5.734
Crime	5.416
Documentary	4.344
Adventure	4.288
Sci-Fi	3.009
Mystery	2.719
Children	2.529
Animation	2.362
Fantasy	2.278
War	1.686
Western	1.357
Musical	1.161

As presented above, both the train and the test data samples include the same percentage per genre as the initial dataset. A goal accomplished with stratified shuffle split.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Concluding the second chapter, the reader reviewed a complete workflow presentation of the data text analysis. Firstly, the document stated the data requirements, the specific rules, the data should follow for the classification experiment. Next, the document described the data sources and the tools utilized to extract movie information based on the pre-defined data requirements. The downloaded data have been cleaned and prepared for language processing with the help of the NLP text preprocessing framework. Finally, the document explained the importance of separating the dataset into three different data samples to help the evaluation process of machine learning classification algorithms. Having cleaned and prepared the input data for natural language processing algorithms, the next chapter will extensively describe the deep learning models used to best estimate the classification pattern towards accurate genre predictions.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter Three

Neural Network Development

3.1 Introduction

The third chapter of the thesis is dedicated to engineering and estimating the optimal deep learning model that will identify as accurately as possible the genre(s) of a movie based on the movie's content information in textual format. Three subsections assemble the third chapter. The first subsection documents the custom model developed to predict the movie genres. In this experiment, a custom model refers to a deep neural network constructed layer by layer. Each component of the network plays a different part in its overall structure. The structure of a neural network is one of the essential elements of deep learning. The neural network structure plays a crucial role in the overall ability of the network to complete the task under development. Every neural network assembled fulfills a goal (e.g., predict the number of tickets in the next Superbowl game, predict the name of the next president of United States by analyzing the sentiment of tweeter messages, or classify movies by their genre). The documentation proceeds with the training and configuration of neural networks offered by TensorFlow and Stanford University. The latter are already built neural networks fitted on data derived from a variety of textual sources, like Wikipedia and Google News. The second section of the chapter presents the configuration and the use of those models. Finally, in the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

third subsection, the models trained from the previous two sections will be compared based on some defined metrics, rules, and reports. Among those different models, the model that will satisfy at the highest rate, the defined rules and metrics will proceed to the next chapter, where the development of the recommendation algorithm takes place. The weights extracted from the best performed neural network model are essential for the recommendation algorithm's performance.

3.2 Custom Neural Network

3.2.1 Development

This section will present the *development* of a **custom** neural network used to predict as accurately as possible the correct sequence of 17 different genres per movie. In general, a movie can have up to 6 or 7 genres on the IMDB website. Although most of the movies from the GroupLens Research dataset had at most three genres, thus the genres of all the movies have been pruned down to three. On a high level, the model developed assigns seventeen probabilities to all the available genres, and the three highest probabilities are the genres classified to the movie. The model classifier with the most correct classifications will be the one selected. However, apart from the correct classifications, a variety of additional measures and rules will also play an important role in which model to select. Please note that for movies assigned to an array of genres, the order of the genres inside the array is not under consideration. For example, a prediction for the movie A, which estimates that the movie belongs to genres ["Action", "Drama", "Thriller"], while the true sequence is ["Drama", "Action", "Thriller], is still a correct prediction even though the order of the genres is wrong. The current section will discuss the following subjects:

- **Neural Network Structure**. Since the neural network is a custom one, it is built based

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

on a *skeleton* that will describe the flow from the input (s) to the output(s).

- **Hyper-parameter** selection.
- **Optimization** algorithms and **performance** metrics selected.
- Thorough **selection plan** crafted to identify the best model classifier among the custom trained neural networks.

Before starting building the neural network, the authors remind to the reader the *independent* variables and the *target* feature of the classification experiment. The model will try to predict the movie's genre (target) based on textual data of five different columns (independent variables), the actors, the plot summary, the movie content, the user reviews, and the title of the movie. Those five variables are the so-called **inputs** of the neural network, whereas the target variable is the **output**.

The neural's network build follows the policies of the Keras Functional API. The Keras functional API is the way to define complex models, such as multi-output models, directed acyclic graphs, or models with shared layers. When creating a neural network, the following components should be specified:

1. Define the Input(s)
2. Define additional layers and connect them
3. Define the Output(s)
4. Compile the Model based on an optimization function, an optimization metric, and a performance metric

Keywords definition

Different types of layers initialized to assemble a complete neural network for classifying movies to a list of different genre(s). Each layer receives an *input* and provides an *output*. Each of those two elements has different shapes, dependent on the type of layer used. Below

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

are some indicative keywords used to configure most of the neural network layers.

- batch_size (or hidden units): The number of sequences injected into the model, per epoch, for training. A range of values assigned to this argument are derivatives of number eight [8, 16, 32, 64, 128, 256, 512, 1024]. *Argument specified in the input, Embedding, and Pooling layers.*
- number_of_samples: The total number of samples in the training dataset. For example, 39,192 sequences of vectorized movie content. *Referenced in the input layer.*
- sequence_length: The maximum length of a padded sequence. The padded sequences across each independent movie content column should have the same sequence value. For example, the actors' column includes sequences of fixed length equal to 15. *Referenced in the input, and Embedding layers.*
- num_timesteps: Same as the sequence_length value. *Referenced in the input layer.*
- steps: Same as the sequence_length value. *Referenced in the Pooling layer.*
- output_dim: The number of numerical weights per embedding layer. For example, a value equal to 50 means that in the first training iteration of the neural network, the embedding weights instantiated for a single word are 50 random floating numbers. *Referenced in the input, Embedding layers*
- features: Same as the output_dim value. *Referenced in the Pooling layer*:
- input_dim: The number of sequences injected into the model for training per iteration. Same as the batch_size. *Referenced in the Dropout layer.*
- units: Same as the batch_size. *Referenced in the Dropout layer.*

Define the Input(s)

Input Layer

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Input: 2D Numpy array of shape (number_of_samples, sequence_length)

Output: 3D tensor of shape (batch_size, sequence_length, output_dim)

The first layer, one of the core layers of a neural network, is the **input layer**. It is the layer that receives the information from the independent variables (called X features). A neural network is either (a) single-input or (b) multiple-input based on the number of input layers it utilizes. The input layer has the following arguments:

- shape: A shape tuple (integer). For instance, shape=(32,) indicates that the expected input will be batches of 32-dimensional vectors. In the current classification experiment, this is the maximum length of the sequence vector describing each textual sentence per column. For example, the column of actors is a 2D array of integer sequences of shape (number_of_samples, num_timesteps), where number_of_samples is equal to 39,192 and num_timesteps is equal to 15 actors per movie.
- batch_shape: A shape tuple (integer), including the batch size. For instance, batch_shape=(10, 32) indicates that the expected input will be batches of ten 32-dimensional vectors. While, batch_shape=(None, 32) indicates batches of an arbitrary number of 32-dimensional vectors. Argument not specified for the current experiment.
- name: Layer's optional name string. It should be unique in a model (do not use the same name twice). Autogenerated, if not provided.
- dtype: Input's data type given as string ("float32", "float64", "int32", "int64").

The input layer returns a tensor. For the current experiment, only the shape and the name arguments specified.

The number of input layers can be vary depending on the shapes and data types of different inputs. For example, suppose the five independent features had the same shape in the current



experiment. In that case, the authors could utilize a single input layer that would receive five columns of the same shape (e.g., all five sequences of movie content would have a maximum length of 20 integers). However, when dealing with features of various lengths, it is more efficient to initialize as many input layers as the independent variables. The current experiment follows the second approach, so the neural network is a multiple-input Keras model, with different dimensional shape per independent feature.

Define additional layers and connect them

Embedding Layer

Input: 2D tensor of shape (batch_size, sequence_length)

Output: 3D tensor of shape (batch_size, sequence_length, output_dim)

Between the input and the output layers, which are two layers that cannot be missing from a neural network, there is also an abundance of other layers to utilize with some of them to be case-specific. One such case-specific layer is the Embedding layer, not selected for every classification experiment. The Embedding layers exist because someone cannot feed a neural network with arrays of integers, which is the output after padding the dataset's textual feature (refer to the second chapter). The correct approach is to turn the arrays of integers into tensors. This transformation is a two-way approach:

- Pad the arrays so that they have the same length, turn the lists into tensors of shape (number of samples, maximum index length) and feed that tensor to a layer capable of handling such integer tensors, like an Embedding layer.
- Or, apply one-hot encoding to the arrays, so they turned into vectors of 0s and 1s. That would turn an array like [2,3] to a 30,000-dimensional vector that will contain 29,998 zeros except for the word indices 2, 3, which would be ones. Then, attach a Dense layer

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

(referenced later in the document), capable of handing float data.

The second approach that of one-hot encoding has two significant drawbacks. Firstly, when dealing with a high-dimensional dataset, one extra addition to the sample will add an extra number to the one-hot encoded vector. For example, representing the words of a dataset with 50,000 movies would need to construct a 50,000-dimensional vector. Such cases make the training of any machine learning model on this representation infeasible. Secondly, the mapping of words in one-hot encoded vectors is entirely uniform. Which means that similar words are not placed close to each other in the vector.

The input of the Embedding layer is a 2D tensor of integers of shape (sample, sequence_length), with each entry being a sequence of integer indices. The sequences are fed o the Embedding layer in batches, and each batch expects sequences of the same length. The arguments specified for a functional embedding layer are the following:

- input_dim: int > 0. Size of the vocabulary, e.g., maximum integer index + 1. The size of the vocabulary generated for each movie feature after the tokenization of the textual data. Each input layer has its embedding layer. The size of the vocabulary is the length of the tokenizer.word_index python method. In the current experiment, the vocabulary index per feature column is:
  - Actors' column: 249,141 (249,140 actor names plus the Out-of-vocabulary token).
  - Plot summary column: 36,908 (36,907 words plus the Out-of-vocabulary token).
  - Movie features column: 143,905 (143,904 words plus the Out-of-vocabulary token).
  - Movie reviews column: 107,182 (107,181 words plus the Out-of-vocabulary token).
  - Movie title column: 20,264 (20,263 words plus the Out-of-vocabulary token).

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- output_dim: int >= 0. The dimension of the dense embedding. This argument specifies the length of the numbers and weights, which will specify the place of a word in multidimensional space. Those numbers start randomly and change after one training iteration. In general, the higher the argument number, the more the available information for a neural network to identify similar words. However, since this argument is considered a hyper-parameter, it is better to test different values. The current experiment tests three values of this argument [50, 100, 150].
- embeddings_initializer: Initializer for the embeddings matrix. Initializations define how to initialize random weights of Keras layers. For the current experiment, the weights are initialized randomly from a uniform distribution.
- embeddings_regularizer: Regularizers allow the application of penalties on layer parameters or layer activity during optimization. If the loss function, optimized by the network, incorporates these penalties then this argument may not be specified.
- embeddings_constraint: Functions from the constraints module allow setting constraints (e.g., non-negativity) on network parameters during optimization—an argument not specified in this experiment.
- mask_zero: Masking in Keras is a technique used to understand and avoid padded values. In the previous chapter, the authors padded the data into the same length tensors, a pre-requisite measure for an input layer. Apart from balancing the lengths of each sentence in the relevant feature corpus, the padding technique generated extra zeros to expand the length of those sequences with smaller lengths. Those extra zeros represent the result of the padding, and they inform the neural network for padded data. When applying the masking mechanism, the neural network can understand that those zeros are a unique masked out "padding" value. Setting the argument to True, as a consequence, the vocabulary cannot contain index 0.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- input_length: The last argument of the Embedding layer, is relevant to the length of each sequence imported into the embedding layer. This value is constant for all the sequences of the same input and Embedding layer. In the current experiment, this value is equal to (15, Actors), (22, Plot summary), (60, Movie Features), (328, User reviews), (6, Movie's title).

GlobalMaxPooling1D layer

Input: 3D tensor with shape: (batch_size, steps, features)

Output: 2D tensor of shape: (batch_size, features)

Global Max Pooling is a type of layer that belongs to the general family of Pooling layers. Add pooling layers on embedded sequences in order to learn each sequence as a whole. In a different scenario, with the absence of a pooling layer, training the model on the presence of only one dense layer would lead to a neural network that treats each word in an input sequence separately without considering the inner-word relationship and sentence structure [6]. For example, a sentiment analysis model would likely treat both "this movie is a bomb" and "this movie is the bomb" as being negative reviews [19]. The max-pooling layer's attachment will optimize the computational power of the network since it gradually minimizes the network's size and will also help the model focus on those information sequences that matter the most into classifying movies to the correct list of the genre(s).
When attaching a *global max-pooling layer* in the neural network, someone must specify the data_format argument; the rest are optional arguments. The authors used the default value of the data_format argument in the current experiment.

Concatenate layer

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Input: 2D tensor with shape: (batch_size, output_dim)

Output: 2D tensor of shape: (batch_size, output_dim)

The layer concatenates a list of given inputs. It receives an array of tensors, and returns a single tensor, the concatenation of all inputs [33]. The concatenated layer includes all the information about the movie content of a single movie. Textual information about actors, plot summary, movie features, user reviews, and movie title, are all concatenated in a single layer describing describe a single movie.

Dense layer

Input: nD tensor with shape: (batch_size, input_dim)

Output: nD tensor of shape: (batch_size, units)

Dense layers also called the non-linear layers of a neural network, are responsible for applying the following *mathematical formula* to the input of a connected layer,

$$output = f(x \cdot w + b)$$

, the inner product between input (x) and the weight matrix, also referred to as the kernel (w) added to a bias factor (b), representing a biased value used to optimize the model. The weights contain the information learned by the network from exposure to training data [7]. The f() function that describes this relationship is non-linear, represents the activation function. Non-linear functions come in handy in situations where a mathematical model should learn more complex structures in the data. In general, the purpose of a machine learning model is to find the optimal function that maps inputs to appropriate outputs by minimizing the bias (error) of doing the mapping wrong. The arguments assembling a *Dense layer* are the following:

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- units: Positive integer, the dimensionality of the output space. Typically, represent the hidden units (e.g., 8, 16, 32, 64, 128, etc). In the current experiment is the number of batch_size units, hyper-parameter tuned using three to five different values [32, 64, 128] or [8, 16, 32, 64, 128].
- activation: Activation function to use. In the current classification experiment, the **Rectified Linear Unit (ReLU)** activation function is used.
- use_bias: Boolean, whether the layer uses a bias vector. Not specified in the current experiment.
- kernel_initializer: Initializer for the kernel weights matrix. An argument not defined for the current classification experiment.
- bias_initializer: Initializer for the bias vector. An argument not defined for the current classification experiment.
- kernel_regularizer: Regularization function applied to the kernel weights matrix. Regularization is a technique to reduce the probability of a model to over-fit and help the neural network distinguish the relevant data from the noise better. Noise is data points representing random chance. Regularization attempts to impose Occam's razor, which discourages the learning of complex models, to avoid the risk of overfitting. In the current classification experiment **l2_regularization** is deployed with 0.01 learning rate.
- bias_regularizer: Regularizer function applied to the bias vector. Since no bias initialized, no argument specified.
- activity_regularizer: Regularizer function applied to the output of the layer (its "activation"). An argument not defined for the current classification experiment.
- kernel_constraint: Constraint function applied to the kernel weights matrix. An argu-

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ment not defined for the current classification experiment.

- bias_constraint: Constraint function applied to the bias vector. An argument not defined for the current classification experiment.

The rectified linear activation function is among the best choices of activation function when developing most types of neural networks. Xavier Glorot et al. very well highlights the advantages of this activation function in their 2012 paper *Deep Sparse Rectifier Neural Networks* and presented in this article by Jason Brownlee A Gentle Introduction to the Rectified Linear Unit (ReLU) [16].

- Computational simplicity: Computations are also cheaper since there is no need for computing the exponential function in activations, with sparsity exploited.
- Representational sparsity. Sparsity is a concept described as a matrix that mainly contains zero values. The ReLU activation function, by definition, uses the expression

$$max(0.0, x)$$

, with positive inputs always returned in a linear function and negative inputs to be turned into zero outputs. The sparse representation is a concept that drastically accelerates the training time of an algorithm but also helps the algorithm to disentangle the information received. Highly dense representations that approximate zero values, are highly conserved by small changes of the input. [4]

- Linear Behavior: It is easier to optimize models of linear behavior [9]. In general, a neural network is easier to optimize when its behavior is linear or close to linear [16].

Dropout layer

Input: nD tensor with shape: (batch_size, input_dim)

Output: nD tensor of shape: (batch_size, units)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Dropout works by randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. The dropout rate is any decimal number in range (0.0, 1.0). For example, if someone sets the dropout rate to 0.3, then 30 percent of the inputs will be dropped. This layer has very effective applications in big data experiments where the computational time and the complexity of the data set a great hurdle in training and optimizing a machine learning model. The number of *dropout layers* used is often defined by the number of the *dense layers* attached to the network. In the current experiment, the drop out rate is equal to 0.1. Thus, 10% of trainable units drop per epoch.

Define the Output(s)

The target (y) of the dataset constructs the final layer of a neural network. The output neuron of the network is a *dense layer* that based on the feature map of weights and inputs created in the previously connected layers, either assigns the input sequence of the incoming batch to the classification label(s) or predicts the value of the output based on the input (regression problems). Specifically, in classification problems, the output can be either single or multi-valued. The former is observed in binary or multi-class classification problems while the latter output format is observed in multi-label classification experiments. The most important argument to specify in the *dense layer* of the output neuron is the value of the targets. In the current experiment, the number of targets is equal to seventeen different movie genres. The experiment is a multi-label classification problem; thus, a movie input can belong to more that one movie genre(s). The neural network is designed and optimized upon the prediction of at most three out of the seventeen genres. Before proceeding to a more elaborate explanation of the output layer, first document the arguments specified for the output neuron of the current experiment. *Please note that since this is a dense layer, the document presents only the respective value of each argument. For the rest of the configurable arguments, please refer to*

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

the previous reference about dense layers.

- units: The number of input data. This amount is equal to the number of unique values the target variable includes. In the current experiment, this is equal to **seventeen movie genres**. More precisely, the output of this layer is a probability of the input to belong to a specific target variable. As a result, by specifying this value equal to seventeen, the authors expect from the neural network to calculate seventeen different probabilities.
- activation: Recall that the activation function is either a non-linear or a linear function that calculates the output of the neuron. Similarly, the type of activation function used affects the probabilities assigned to the seventeen movie genres, given the set of five different inputs (cast, plot summary, movie features, reviews, and title). In the current experiment, the debate was among two activation functions, which, based on bibliography and online articles on various websites, mostly used for classification problems. Those two activation functions are the **sigmoid** and the **softmax**. The difference between the two activation functions boils down to the fact that the probability values calculated by the sigmoid function are not conditional probabilities. These are contrary to the probabilities calculated by the softmax activation function, which are conditional. Conditional probabilities are the kind of probabilities they add up to 1 when summed. Based on that, multi-class classification problems often use the softmax activation function. As a result, the prediction algorithm assigns the dependent feature to the target value with the highest probability. Since the experiment is a multi-label and not a multi-class classification problem, the sigmoid activation function is selected. Specifically, from the probabilities calculated by the sigmoid function, the three highest of those are assigned to a particular movie. Then, the authors test whether or not the result of the assignment is correct, and if the genres assigned are indeed the correct genres and not a random mistake of the model, then the chances to find the best model estimator increase.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Create the model

The structure of the neural network is now complete. The final step is to transform the network's structure into a machine learning model capable of assigning seventeen different probabilities as accurately as possible given the movie content input infused into the model. Keras library provides the method Model() that initializes the instance of a machine learning model. The keywords of the model () method are the input(s) and the output(s). The input(s) are the five movie-content textual data for which five *input layers* were created. The output(s) is the *output layer* defined at the end of the network's structure.

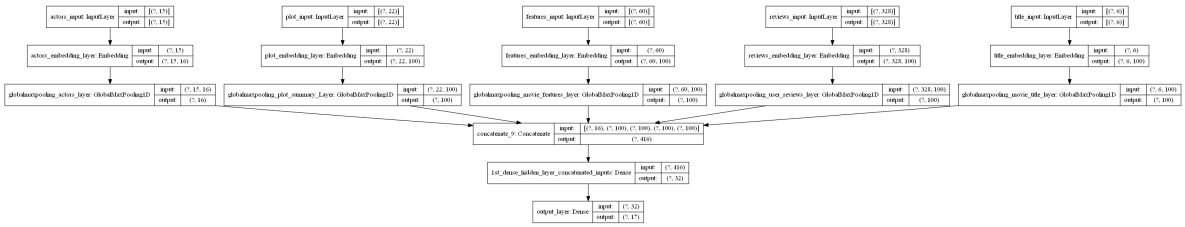


Figure 3.1 Neural network's structure

The first horizontal layer presents the five *input layers*. Each input layer corresponds to a different movie content column based on the vocabulary index and the length of the sequences. The second horizontal layer includes the *embedding layers* connected to each of the five inputs, respectively. Then, the third horizontal layer attaches the embedding layers to a single-dimension *global max-pooling layer*. The concatenated layer follows, which includes all the information received for a movie based on the movie's content information. Then the structure establishes a connecting link between the concatenated layer and a dense layer. This layer utilized the ReLU activation function to create a sparse feature map and downsample the concatenated layer's information. Finally, the model structure imports the sparsed feature map to an output layer that computes seventeen different probabilities, one per movie genre. Those probabilities denote the likelihood of a movie to belong to each different movie genre.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Now that the neural network is built and transformed into a machine learning model, it is time to put it into training.

3.2.2 Compiling & Training the neural network

The training phase of a machine learning model demands the prior compilation of the neural network. The compilation is achieved by specifying the following arguments:

- Optimization function: The optimization function is responsible for computing the gradient of the network's parameters per iterative batch and updating the weighted parameters a step forward or backward, helping the machine learning model converge to a local/optimal minimum for the monitored loss function. Update each step towards optimization concerning a learning rate, which defines the magnitude of the gradients' improvement forward or backward.
- Loss function: Each training iteration draws a batch of training samples X and target variables y. The network uses those inputs to estimate the corresponding predictions, referenced as y_pred. Next, the model estimator computes the loss, a mismatch among the actual values of the target y, and the predictions (y_pred). Concerning the loss, use the network's parameters to compute the gradient of the loss. The algorithm will reduce the batch loss by updating the network's parameters in the opposite direction of the gradient [8]. Given the correct values of the movie genre(s) of a single movie, how can the model estimator evaluate the predicted probabilities' goodness (or badness)? The answer to this question is the ground rule for specifying a loss function [24]. The document demonstrates an example of a loss function application on a multi-label experiment. Based on the weights and the gradients calculated through the execution of a neural network, the last layer calculates a numerical array with length equal to the number of unique target labels. In the current experiment, the target labels are seventeen.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

movie genres. So the final array would be something like:

$$[1.1, -.1, .33, .41, .98, .001, 1.5, -.003, -.33, .12, .65, .2, .0001, -2.1, .09, -.2, .15]$$

Applying the sigmoid activation function on that array, each element will fall to the range between 0, 1 inclusive

$$\downarrow$$

$$[1, 0, .33, .41, .98, .001, 1, 0, 0, .12, .65, .2, .0001, 0, .09, 0, .15]$$

Then, element-wise the optimization function selected, **binary crossentropy**, compares those seventeen outputs to the ground truth values (one-hot encoded),

$$[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$$

Furthermore, calculates the deviance from the truth. Then the errors are summed. The goal is to minimize, the more, the better, the deviance from the true values of each genre per movie.

- Metric: The metric(s) to be evaluated by the model during training and testing. For binary and multi-class classification problems, most researchers select the binary accuracy, although since this is a multi-label classification model, binary accuracy does not fit appropriately. Alternative measure(s) of performance is the hamming loss and the f1 score. The neural network developed uses both hamming loss and the f1 score to optimize its performance of the validation dataset and improve its predictions. Apart from the type of classification problem, one additional reason the authors preferred the hamming loss over the binary accuracy, is the number of zeros in the vector of the target variable. Recall, that a movie can have up to three genres from a pool of seventeen movie genres. Thus a vector of a random movie would look like [0, 0, 1, 1, 1, 0, 0, 0, 0,

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. In any case, the negative values outnumber the positive values. When someone selects the binary accuracy as a performance metric, it is highly probable that the neural network developed will achieve great accuracy score by correctly predicting the negative values. The document will later compare two trained model classifiers one that used the hamming loss metric and the other that used the binary accuracy.

For **optimization functions**, the authors used Support Gradient Descent (SDG), Adam, and RMSprop algorithms. The plan executed by the authors is to deploy the three optimizers during the training phase and compare their results afterward. Select the optimizer with the lowest loss value and the lowest hamming loss. The **loss function** selected is the binary_crossentropy due to the sigmoid activation function embedded on the last *dense layer*. The **metric** selected is the hamming loss and f1-score.

To complete the training stage, take into consideration two more parameters. The first parameter is the number of **epochs**, i.e., the number of training rounds. Each round calculates the weights and the validation loss of the neural network and adjusts them accordingly, aiming to find the weights framework to minimize the neural network's loss value. During a training epoch, the algorithm monitors the hamming loss, the f1-score, and the binary crossentropy of the neural network on both validation, and training samples. The second and the last parameter specified is the **early stopping** strategy. Early stopping is a strategy of ending the neural network training on a training epoch earlier than the number of epochs specified, taking into consideration the improvement in the loss function on validation samples. Suppose for a consecutive number of training epochs; the loss function is not improving (downgrading) below a certain threshold. The latter phenomenon is a signal that the algorithm has learned the data very well and cannot improve anymore. This non-improvement in the validation loss with the simultaneous improvement of the training loss, is a sign of overfitting, and the model should immediately stop its training. The early stopping applied saves the weights of

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

the model at any training iteration spotted with a validation loss lower than the validation loss of the previous training rounds. It is best practice to set a high value for the training epochs, such as 100, 150, or 200 epochs, and apply early stopping at 10-20 epochs. In such a way, the neural network will have time to learn the data and stop learning in case of no improvement in the validation loss for 10-20 consecutive epochs. The number of epochs in the current experiment is equal to 150, with ten early stopping rounds. Moreover, the monitored loss can have a threshold of improvement with values less than this threshold, not considered an improvement. This variable threshold, referred to as minimum delta, is set to 0.01, with decimal values between 0 and 1 being good choices.

Having settled the parameters as mentioned above, the neural network training with each of the three optimization functions is ready for initiation. Below there is a summary table of the variables configured to successfully train and optimize the custom neural network:

Parameters	Value	Description
Number of input layers	5	[actors, plot summary, movie features, movie reviews, title]
Number of output targets	17	-
Input length of embedding layer	(15, actors), (22, plot summary), (60, movie features, (328 movie reviews), (6, movie title)	
Actor names tokenized	249140	-
Plot words tokenized	36908	-
Movie features words tokenized	143905	-
Movie reviews words tokenized	107182	-
Movie title words tokenized	20264	-
Activation function of output layer	sigmoid	-
Loss function	binary crossentropy	-
Metric	[Hamming loss, f1 score]	F1 score with micro average, and None average
Optimization function(s)	[Adam, SGD, RMSprop]	-
Batch size (hyper parameter)	[32, 64, 128]	-
Embedding dimension size (hyper parameter)	[50, 100, 150]	-
Learning rate (hyper parameter)	[0.001, 0.01, 0,1]	-
Decay steps multiplier (hyper parameter)	[10, 20]	Every how many epochs the value of the learning rate is decreased
Validation split ratio	70% training, 30% validation dataset	-
Number of models trained	108	54 models with Adam optimizer, 27 models with SGD optimizer. 27 models with RMSprop optimizer
Train-test split ratio	80% training, 20% test dataset	
Epochs	150	-
Early stopping rounds	10	-
Minimum delta	0.009	If the validation loss drops below 0.009 units, it is not considered an improvement

Table 3.1 Summary table of training the custom neural network (Hamming Loss - F1 score performance metrics)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

The training phase of the neural network estimator generates four outputs related to the performance of the model estimator:

Output 1

Train the neural network for 150 epochs with ten epochs of early stopping. When the model detects an improvement in the selected validation loss function, at the same moment, the algorithm saves the weights of that model.

Output 2

Create & Save a pandas DataFrame object with the **training_loss, training_hamming_loss, training_f1_score_micro, training_f1_score_none, validation_loss, validation_hamming_loss, validation_f1_score_micro, validation_f1_score_none and epoch**. Later the authors will use this DataFrame for comparing and selecting the best estimator among each different neural model.

Output 3

Present three different figures. The first figure demonstrates the progress of the training hamming_loss and the validation hamming_loss learning curves through each epoch. The second figure portrays the progress of the training and validation losses learning curves per epoch. And the final figure presents the training and validation learning curves of the f1-score, which is the second performance metric monitored by the model classifier. All three figures are valuable measures for comparing different models and identifying an estimator's overfitting or underfitting behavior.

Output 4

Finally, during the training phase, the algorithm calculates the model's testing hamming_loss, f1 score and loss values advised as measures of the neural network's performance over never-seen-before movies.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

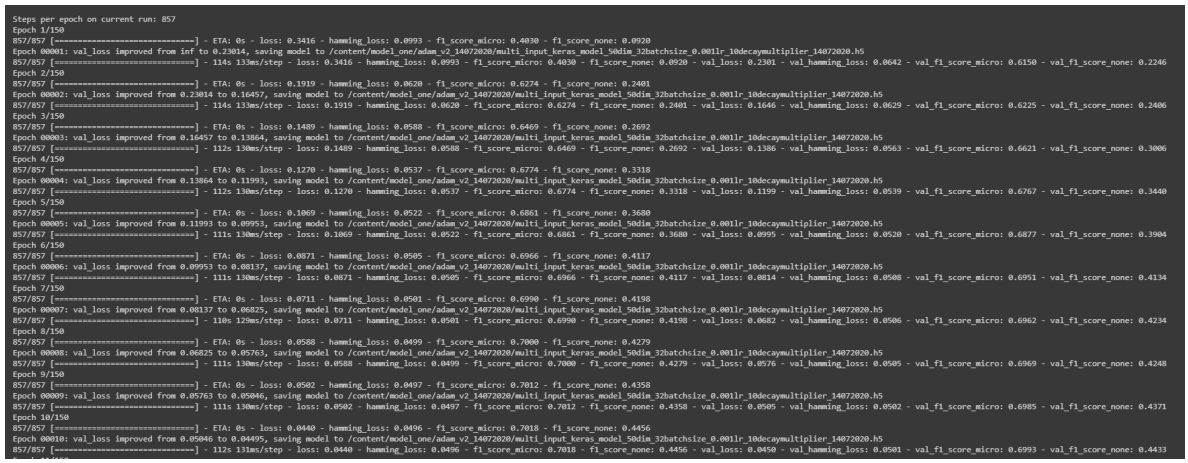


Figure 3.2 Output 1: Training per epoch

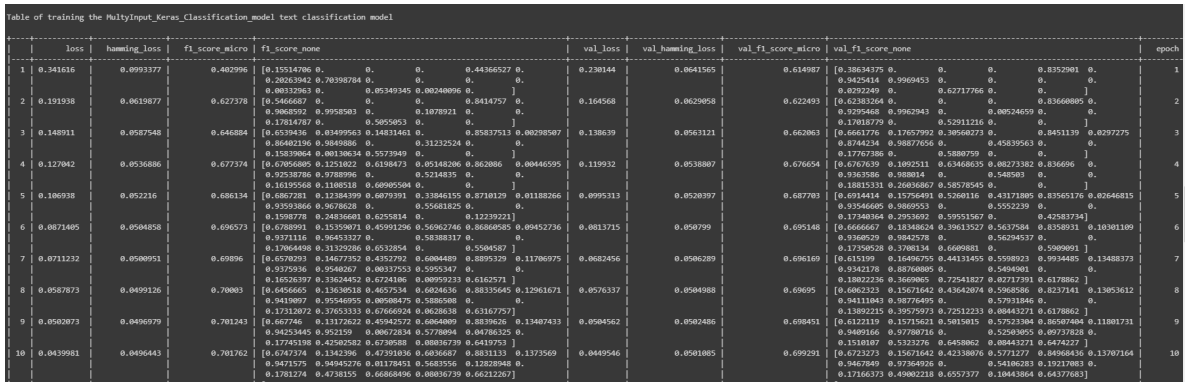


Figure 3.3 Output 2: Neural Network loss function and hamming loss, f1 score progress per epoch

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

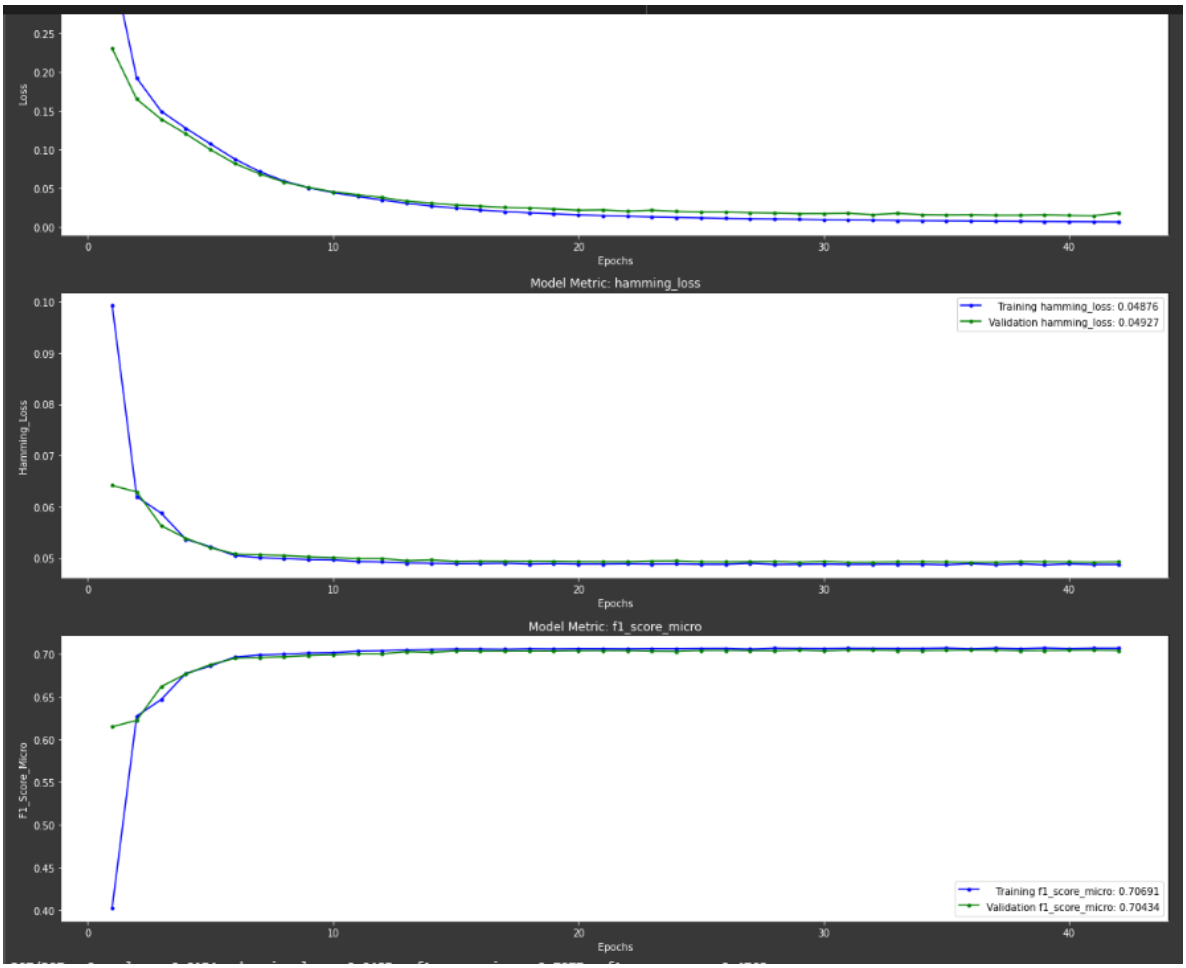


Figure 3.4 Output 3: Training-Validation Hamming loss, F1 score & Loss per epoch

```
307/307 - 2s - loss: 0.0154 - hamming_loss: 0.0485 - f1_score_micro: 0.7077 - f1_score_none: 0.4768

Test Score (evalution of the model's loss/error on the test sequences): 0.015434560365974903

Test model_metric (evalution of the hamming loss on the test sequences): 0.04849836975336075
```

Figure 3.5 Output 4: Hamming loss, Loss score performance of the neural network on never-seen-before movies

The four figures above are generated for all the 108 different neural network estimators trained using the custom developed neural network.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3.2.3 Select best model classifier (per optimizer)

After the 108 model estimators' training, select the one that demonstrated the best performance and evaluation scores among the validation and the test dataset. That is the main goal of this subsection of chapter 3. Then, in the next section of the chapter, the authors use the model estimator selected to compare it with the estimators calculated from the pre-trained algorithms. The metrics employed to select the neural network that will stand out from the rest of the trained estimators are the following:

- **Hamming loss & Zero-one loss**: Select the model estimator having the lowest of both values. Hamming loss and zero-one loss are loss functions, thus the lower their value, the better the performance indicator of the model. Recall that hamming loss also used as the performance metric while training the model estimators.
- **F1-score**: Select the estimator with the highest F1 score
- **Area Under the Curve (ROC-AUC) score**: Select the estimator with the highest ROC score.
- **Hamming loss - Loss** on never-seen-before data: Select the estimator with the lowest testing hamming loss and the lowest loss (error) predictions.
- **Classification report & Confusion matrix**: Select the best model estimator that illustrates a complete capability of predicting all the seventeen genres. Discard the neural network models that cannot observe all of the seventeen movie genres.
- **Predicted vs. Actual Genres**: Compare model predictions with the actual genres.
- **Training and validation learning curves**: Select the estimator that demonstrated the best progress of validation and training hamming loss & binary crossentropy through the training of 150 epochs.
- **Bias-Variance trade-off**: Select the least biased model estimator with the lowest

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

variance of predictions.

The authors combined the measures of comparison mentioned above to construct a compact and accurate plan of selecting the best model estimator.

Selection Plan

Support Gradient Descent optimizer (SGD): Out of the twenty-seven trained models, select the *one* with the lowest hamming & zero-one losses.

Adam optimizer: Out of the fifty-four trained models, select the *one* with the lowest hamming & zero-one losses.

RMSprop optimizer: Out of the twenty-seven trained models, select the *one* with the lowest hamming & zero-one losses.

From the first *cleaning* task, the authors end up with three model estimators, the best ones per optimizer. From this point forward, the comparison techniques will focus on distinguishing the best estimator out of those three models.

Next, the document will present a series of five comparison techniques to select the estimator that outperforms the other two classifier estimators.

Comparison 1 Classification Report and Confusion matrix comparison. From this comparison, it is essential to identify the model that can identify movies from all the seventeen different movie genres. Moreover, the model that correctly predicts most movies from each genre is another important outcome when examining the confusion matrix of a classifier.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Comparison 2 Hamming loss and Loss values on test data. Evaluate estimator's ability to predict data the three classifiers never seen before. Use unbiased movie content sample not used in training the classifiers, to evaluate the generalization and the predictive power of each of the three estimators.

Comparison 3 Select randomly twenty different movies from the test movie dataset, and for each of the twenty movies, use the classifiers to predict their movie genres. Assign to the movie the three genres with the highest probability predicted by the classifier. Compare the predicted genres to the actual genres of those twenty different movies; select the classifier with the least deviance from the true values.

Comparison 4 Compare the training and validation learning curves of the optimization and performance metrics. Those figures will point any sign of overfitting or underfitting existing on any of those three classifiers. The goal is to select the model estimator with good fit learning curves.

Comparison 5 Calculate the bias-variance trade-off and select the model estimator with the lowest bias and variance overall. The documentation includes the fifth comparison into the scoring table along with additional scoring metrics per model estimator (e.g., F1-score, ROC-AUC score, Hamming Loss, and Zero-one loss).

Before moving forward to implementing the above selection plan, the document will first describe the decision metrics referenced above. Those decision metrics are the *hamming loss, the zero-one loss, the f1-score and the area under the curve score.*

Hamming loss

Hamming loss [36] is the fraction of the of incorrectly predicted labels. Calculate the hamming

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

loss using the following mathematical formula:

$$\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \text{xor}(y_{i,j}, z_{i,j}) \tag{3.1}$$

N is the total number of sample observations,

L is the total number of different target labels,

$y_{i,j}$ is the target,

$z_{i,j}$ is the prediction

xor is the *exclusive or* operation.

XOR operation is bound to the following set of rules:

Table 3.2 XOR operations

Target	Operation	Prediction	Result
0	^	0	0
1	^	1	0
0	^	1	1
1	^	0	1

An application of the hamming loss formula in a multi-label experiment is demonstrated below,

Example 1

Actual values: [[0 1], [1 1]]

Predicted values: [[0 1], [1 1]]

Actual XOR Predicted (element-wise): [[0 0 0 0]]

Hamming loss:

$$\frac{0+0+0+0}{4 \cdot 1} = 0 \text{(totally accurate model classifier)} \tag{3.2}$$

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Example 2

Another dummy example,

Actual values: [[0 1], [1 1]]

Predicted values: [[0 0], [0 1]]

Actual XOR Predicted (element-wise): [[0 1 1 0]]

Hamming loss:

$$\frac{0+1+1+0}{4 \cdot 1} = 0.5(\text{non-accurate model classifier}) \tag{3.3}$$

Kindly note that the XOR operation applied by the hamming loss makes the latter non-differentiable. Someone cannot use non-differentiable functions in training neural networks. On each batch of training data the neural network computes how much the loss function will improve or degrade if the networks moves a particular trainable weight by a very small amount delta. This is also referenced as the concept of backpropagation and how it works on the gradients of a fitting neural network. Hamming loss is by definition non differentiable due to the XOR operation, so for small movements of trainable weights no change in the loss is experienced. This is the main reason why the hamming loss was not used as a loss function but rather as a performance metric, offered by the Tensorflow Addons library authored by Google.

Zero-one loss

The number of misclassifications over the total number of predicted values. The zero-one loss is either presented as a fraction, either as an integer number. The *zero-one loss* [39] considers the entire set of labels for a given sample incorrect if it does not entirely match the true set of labels. hamming loss is more forgiving in that, since it penalizes only the individual labels. The calculation formula to compute the zero-one loss is similar to the one used for computing the hamming loss. However, as already mentioned, in multi-label classification experiments, the zero-one loss is more strict in calculating the miss-classification since the entire array of

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

predicted values should exactly match the array of the true values. Hamming loss is illustrated as fraction and zero-one loss as an integer number.

F1-score

Interpret f1 score as a weighted average of the precision and recall, with best scores going closer to 1. There is an equal contribution of precision and recall to the f1 score. The mathematical formula is as follows:

$$2 \cdot \frac{(precision \cdot recall)}{(precision + recall)} \tag{3.4}$$

, with *precision* to be:

$$\frac{tp}{(tp + fp)} \tag{3.5}$$

, where *tp* is the number of true positives and *fp* the number of false positives. The *precision* [37] is intuitively the ability of the classifier not to label as positive a true negative sample. For example, precision is used for classification experiments in cyber security engines, where the identification of all the malware components of a program are of great importance to the estimator's credibility. On the other hand, calculate *recall* as:

$$\frac{tp}{(tp + fn)} \tag{3.6}$$

, where *tp* is the number of true positives and *fn* the number of false negatives. The *recall* [38] is intuitively the ability of the classifier to find all the positive samples. For example, recall is the most famous metric used in medical experiments where the fault of classifying a patient affected by pneumonia as healthy will probably have crucial negative effect on his health. Consider f1-score as a more balanced measure than precision and recall alone. Due to the trade-off between precision and recall, which often confuses readers about the results of the estimator trained, f1-score encapsulates information from both the precision and recall, making it a desirable option for measuring classification performance.

Confusion Matrix

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

		Actual Classes	
		P	N
Predicted Classes	P	TP	FP
	N	FN	TN

Area Under the Curve score (ROC-AUC)

Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. A commonly used measure of estimator performance for classification experiments. Prefer Values closer to 1.0.

The selected metrics, which will measure the prediction quality of a multi-label classifier, have been described. It is high time to reveal the best model estimator by applying the selection plan.

Both f1 score and ROC-AUC score have been calculated by computing the **micro** and **sample** average values. For the first average value, the f1 score is calculated globally by counting the total true positives, false negatives and false positives for each of the precision and recall values accordingly. The micro average value is an ideal option when class imbalance is present in the dataset. For example, for a set of data with four different sample rows, the relative micro average precision and recall values are calculated as:

$$precision = \frac{tp_1 + tp_2 + tp_3 + tp_4}{(tp_1 + tp_2 + tp_3 + tp_4 + fp_1 + fp_2 + fp_3 + fp_4)} \tag{3.7}$$

$$recall = \frac{tp_1 + tp_2 + tp_3 + tp_4}{(tp_1 + tp_2 + tp_3 + tp_4 + fn_1 + fn_2 + fn_3 + fn_4)} \tag{3.8}$$

On the other hand, the sample average of the f1-score is the calculated precision and recall for each data instance and then calculating their average. The same logic is also applicable to the computation of the ROC-AUC score.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Selection plan Step 1: Select the model estimator with the lowest Hamming & Zero-one loss (per optimizer)

For the selection of the best model estimator per optimization function, take into consideration the following three tables,

Tag Name	Embedding Dimension tag	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
multi_input_keras_model-1	50	32	0.001	10	0.023078	0.048498	0.004844	752.0	0.977308	0.978130	0.985796	0.989194	0.000591	0.095053
multi_input_keras_model-2	50	32	0.001	20	0.023070	0.048480	0.005481	849.0	0.974329	0.976750	0.984157	0.988494	0.000581	0.095063
multi_input_keras_model-3	50	32	0.010	10	0.033005	0.048722	0.005013	780.0	0.976594	0.977431	0.986738	0.989707	0.000061	0.095582
multi_input_keras_model-4	50	32	0.010	20	0.041583	0.048774	0.007708	1177.0	0.963479	0.963689	0.973318	0.978954	0.002520	0.093133
multi_input_keras_model-5	50	32	0.100	10	0.300758	0.054840	0.075212	7938.0	0.460306	0.389196	0.649620	0.667465	0.070047	0.031203
multi_input_keras_model-6	50	32	0.100	20	0.253460	0.060170	0.080098	8191.0	0.404516	0.340919	0.626885	0.645490	0.075358	0.026641
multi_input_keras_model-7	100	32	0.001	10	0.019068	0.058487	0.003620	566.0	0.983106	0.983534	0.990600	0.992834	-0.000024	0.095667
multi_input_keras_model-8	100	32	0.001	20	0.022374	0.057243	0.004070	635.0	0.980984	0.981173	0.988942	0.991414	0.000160	0.095483
multi_input_keras_model-9	100	32	0.010	10	0.041931	0.056338	0.006609	1031.0	0.968731	0.969315	0.976843	0.981891	0.002266	0.093385
multi_input_keras_model-10	100	32	0.010	20	0.037357	0.055571	0.004610	729.0	0.978491	0.979166	0.988171	0.990820	-0.000085	0.095728
multi_input_keras_model-11	100	32	0.100	10	0.292896	0.060655	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-12	100	32	0.100	20	0.292848	0.064891	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-13	150	32	0.001	10	0.023669	0.063635	0.004334	671.0	0.979770	0.980480	0.988696	0.991451	-0.000009	0.095653
multi_input_keras_model-14	150	32	0.001	20	0.020337	0.062574	0.003932	617.0	0.981635	0.981700	0.989439	0.991604	0.000109	0.095535
multi_input_keras_model-15	150	32	0.010	10	0.040651	0.061666	0.005283	825.0	0.975345	0.976595	0.986241	0.989431	-0.000019	0.095662
multi_input_keras_model-16	150	32	0.010	20	0.053480	0.060882	0.006375	939.0	0.970441	0.973368	0.986295	0.989602	-0.001128	0.096774
multi_input_keras_model-17	150	32	0.100	10	0.292648	0.063859	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-18	150	32	0.100	20	0.293460	0.066505	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-19	50	64	0.001	10	0.047707	0.065594	0.012738	1891.0	0.938949	0.941866	0.955237	0.965982	0.004449	0.091226
multi_input_keras_model-20	50	64	0.001	20	0.019794	0.064743	0.003998	629.0	0.981329	0.981824	0.989328	0.991867	0.000085	0.095559
multi_input_keras_model-21	50	64	0.010	10	0.042325	0.063992	0.006627	1018.0	0.968592	0.967960	0.976044	0.980890	0.002559	0.093095
multi_input_keras_model-22	50	64	0.010	20	0.037438	0.063322	0.005007	790.0	0.976628	0.977970	0.986864	0.989895	0.000019	0.095625
multi_input_keras_model-23	50	64	0.100	10	0.264272	0.063575	0.057131	5873.0	0.642554	0.611862	0.738925	0.779439	0.048670	0.049934
multi_input_keras_model-24	50	64	0.100	20	0.253114	0.064055	0.070097	6815.0	0.537050	0.512018	0.687747	0.728382	0.057256	0.042334
multi_input_keras_model-25	100	64	0.001	10	0.020341	0.063436	0.003998	630.0	0.981327	0.982393	0.989278	0.992043	0.000104	0.095540
multi_input_keras_model-26	100	64	0.001	20	0.022036	0.062867	0.004244	669.0	0.980167	0.980441	0.988401	0.990787	0.000194	0.095450
multi_input_keras_model-27	100	64	0.010	10	0.045093	0.062360	0.005379	834.0	0.974971	0.977354	0.987297	0.990863	-0.000518	0.096162
multi_input_keras_model-28	100	64	0.010	20	0.045117	0.061886	0.007246	1104.0	0.965617	0.964281	0.973873	0.978833	0.002774	0.092881
multi_input_keras_model-29	100	64	0.100	10	0.260276	0.062521	0.080206	7099.0	0.517740	0.527918	0.691925	0.733276	0.042248	0.055692
multi_input_keras_model-30	100	64	0.100	20	0.249638	0.063351	0.081119	8198.0	0.391224	0.330034	0.621629	0.641671	0.076754	0.025448
multi_input_keras_model-31	150	64	0.001	10	0.021180	0.062874	0.003914	613.0	0.981737	0.982012	0.989917	0.992242	-0.000057	0.095700
multi_input_keras_model-32	150	64	0.001	20	0.019501	0.062426	0.003626	561.0	0.983137	0.983898	0.992125	0.994212	-0.000612	0.096256
multi_input_keras_model-33	150	64	0.010	10	0.034596	0.062015	0.004712	731.0	0.977983	0.979514	0.987276	0.990530	0.000156	0.095488
multi_input_keras_model-34	150	64	0.010	20	0.039516	0.061627	0.004388	679.0	0.979634	0.980213	0.991057	0.993080	-0.000964	0.096609
multi_input_keras_model-35	150	64	0.100	10	0.292620	0.063052	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-36	150	64	0.100	20	0.292751	0.064398	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-37	50	128	0.001	10	0.026874	0.048510	0.005649	899.0	0.973353	0.974396	0.980611	0.985185	0.001779	0.093870
multi_input_keras_model-38	50	128	0.001	20	0.030346	0.048594	0.005895	874.0	0.972315	0.973377	0.981903	0.986269	0.001032	0.094614
multi_input_keras_model-39	50	128	0.010	10	0.037972	0.048694	0.005709	881.0	0.973334	0.974483	0.984770	0.988513	0.000118	0.095526
multi_input_keras_model-40	50	128	0.010	20	0.038123	0.048763	0.004892	756.0	0.977409	0.978931	0.992008	0.993702	-0.001824	0.097473
multi_input_keras_model-41	50	128	0.100	10	0.143329	0.049872	0.028382	3537.0	0.855501	0.849887	0.889218	0.907749	0.014635	0.081326
multi_input_keras_model-42	50	128	0.100	20	0.177307	0.050339	0.026143	3438.0	0.869011	0.867428	0.901569	0.919845	0.011944	0.083915
multi_input_keras_model-43	100	128	0.001	10	0.024191	0.050081	0.004280	674.0	0.980023	0.979751	0.988849	0.990923	-0.000014	0.095658
multi_input_keras_model-44	100	128	0.001	20	0.022159	0.049872	0.003986	622.0	0.981373	0.982116	0.989088	0.991761	0.000189	0.095455
multi_input_keras_model-45	100	128	0.010	10	0.037417	0.049742	0.005397	833.0	0.974579	0.976048	0.981911	0.986511	0.001531	0.094116
multi_input_keras_model-46	100	128	0.010	20	0.036798	0.049688	0.005709	886.0	0.973272	0.974680	0.983758	0.987926	0.000506	0.095138
multi_input_keras_model-47	100	128	0.100	10	0.197232	0.051040	0.050599	4935.0	0.730384	0.730660	0.813182	0.843249	0.022284	0.074065
multi_input_keras_model-48	100	128	0.100	20	0.235121	0.051644	0.044746	5040.0	0.760260	0.765235	0.826372	0.860927	0.023202	0.073203
multi_input_keras_model-49	150	128	0.001	10	0.022547	0.051398	0.004088	648.0	0.980870	0.980871	0.988217	0.990744	0.000420	0.095223
multi_input_keras_model-50	150	128	0.001	20	0.026666	0.051197	0.004316	677.0	0.979907	0.980200	0.989889	0.992031	-0.000447	0.096091
multi_input_keras_model-51	150	128	0.010	10	0.046663	0.051049	0.006051	912.0	0.972006	0.975033	0.988129	0.991183	-0.001502	0.097149
multi_input_keras_model-52	150	128	0.010	20	0.043143	0.050940	0.004436	692.0	0.979395	0.980513	0.990587	0.992581	-0.000833	0.096477
multi_input_keras_model-53	150	128	0.100	10	0.244658	0.052108	0.067954	7113.0	0.544540	0.512066	0.688799	0.724885	0.059563	0.040310
multi_input_keras_model-54	150	128	0.100	20	0.259525	0.053504	0.062353	6763.0	0.593289	0.533769	0.711884	0.738040	0.055296	0.044059

Table 3.3 Adam Optimization function Results - Hamming loss performance metric (best model highlighted)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Tag Name	Embedding Dimension tag	Batch tag	Learning Rate tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
multi_input_keras_model-1	50	32	0.001	0.298052	0.111494	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-2	50	32	0.010	0.212665	0.094722	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-3	50	32	0.100	0.071430	0.080362	0.020092	2764.0	0.900733	0.895532	0.923180	0.937286	0.009577	0.086206
multi_input_keras_model-4	100	32	0.001	0.295311	0.088145	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-5	100	32	0.010	0.208087	0.086056	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-6	100	32	0.100	0.050056	0.079954	0.011694	1810.0	0.943595	0.942425	0.955230	0.963869	0.005530	0.090162
multi_input_keras_model-7	150	32	0.001	0.297274	0.084460	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-8	150	32	0.010	0.192245	0.082664	0.056050	6595.0	0.646928	0.594574	0.739530	0.763374	0.049756	0.048966
multi_input_keras_model-9	150	32	0.100	0.066832	0.079098	0.015272	2225.0	0.926244	0.922085	0.945336	0.954278	0.005745	0.089951
multi_input_keras_model-10	50	64	0.001	0.302150	0.082337	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-11	50	64	0.010	0.215684	0.081422	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-12	50	64	0.100	0.070405	0.078802	0.021953	3023.0	0.891410	0.886439	0.917823	0.931043	0.009777	0.086012
multi_input_keras_model-13	100	64	0.001	0.307609	0.081317	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-14	100	64	0.010	0.214271	0.081092	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-15	100	64	0.100	0.046036	0.078987	0.009035	1363.0	0.957116	0.950964	0.969073	0.972617	0.002822	0.092834
multi_input_keras_model-16	150	64	0.001	0.309067	0.081019	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-17	150	64	0.010	0.212941	0.080838	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-18	150	64	0.100	0.054171	0.079086	0.016100	2473.0	0.921940	0.914813	0.941518	0.948758	0.006401	0.089306
multi_input_keras_model-19	50	128	0.001	0.374475	0.080792	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-20	50	128	0.010	0.230063	0.080664	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-21	50	128	0.100	0.151229	0.079565	0.044368	5523.0	0.740603	0.733548	0.795316	0.831316	0.037670	0.059840
multi_input_keras_model-22	100	128	0.001	0.388864	0.081016	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-23	100	128	0.010	0.230780	0.081268	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-24	100	128	0.100	0.084637	0.080011	0.021857	2973.0	0.889409	0.879536	0.908778	0.925100	0.013593	0.082326
multi_input_keras_model-25	150	128	0.001	0.394759	0.081270	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-26	150	128	0.010	0.224955	0.081151	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-27	150	128	0.100	0.085071	0.080093	0.020494	2879.0	0.896558	0.884273	0.913338	0.928266	0.013180	0.082724

Table 3.4 SGD Optimization function Results - Hamming loss performance metric (best model highlighted)

Tag Name	Embedding Dimension tag	Batch tag	Learning Rate tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
multi_input_keras_model-1	50	32	0.001	0.022421	0.048931	0.005853	893.0	0.972581	0.973750	0.983111	0.987126	0.000610	0.095034
multi_input_keras_model-2	50	32	0.010	0.065825	0.049177	0.015152	2192.0	0.926324	0.929936	0.942764	0.956719	0.006893	0.088825
multi_input_keras_model-3	50	32	0.100	0.313867	0.061737	0.086858	7310.0	0.476576	0.503470	0.673774	0.706148	0.042613	0.055363
multi_input_keras_model-4	100	32	0.001	0.035106	0.058622	0.008836	1295.0	0.958170	0.954860	0.970763	0.975454	0.002367	0.093286
multi_input_keras_model-5	100	32	0.010	0.082441	0.057299	0.024204	3417.0	0.880193	0.857991	0.911655	0.918797	0.009892	0.085900
multi_input_keras_model-6	100	32	0.100	0.375009	0.062219	0.079990	8191.0	0.404948	0.341038	0.626995	0.645586	0.075451	0.026562
multi_input_keras_model-7	150	32	0.001	0.028530	0.060338	0.006969	1109.0	0.967182	0.967235	0.977948	0.982866	0.001474	0.094173
multi_input_keras_model-8	150	32	0.010	0.065675	0.059075	0.017793	2509.0	0.912916	0.913544	0.933123	0.946800	0.007989	0.087753
multi_input_keras_model-9	150	32	0.100	0.470339	0.064899	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
multi_input_keras_model-10	50	64	0.001	0.025903	0.063285	0.006357	981.0	0.970486	0.972354	0.985763	0.988929	-0.000908	0.096553
multi_input_keras_model-11	50	64	0.010	0.053238	0.061999	0.012720	1901.0	0.938759	0.940597	0.953348	0.964406	0.005212	0.090474
multi_input_keras_model-12	50	64	0.100	0.334415	0.063298	0.062852	6904.0	0.585740	0.529925	0.707315	0.734320	0.056939	0.042613
multi_input_keras_model-13	100	64	0.001	0.047492	0.062266	0.013195	1848.0	0.936675	0.934273	0.953453	0.963095	0.004682	0.090996
multi_input_keras_model-14	100	64	0.010	0.084050	0.061564	0.024798	3162.0	0.872974	0.864597	0.896132	0.916732	0.015661	0.080344
multi_input_keras_model-15	100	64	0.100	0.483976	0.063250	0.086858	7310.0	0.476576	0.503470	0.673774	0.706148	0.042613	0.055363
multi_input_keras_model-16	150	64	0.001	0.020697	0.062347	0.004064	637.0	0.981206	0.982287	0.993508	0.994967	-0.001572	0.097220
multi_input_keras_model-17	150	64	0.010	0.083864	0.061801	0.027158	3486.0	0.863017	0.842786	0.896192	0.910368	0.013067	0.082832
multi_input_keras_model-18	150	64	0.100	0.625151	0.064240	0.098708	9599.0	0.248457	0.103317	0.571733	0.555944	0.078880	0.023635
multi_input_keras_model-19	50	128	0.001	0.056008	0.063453	0.016028	2465.0	0.922048	0.921728	0.940400	0.951443	0.006922	0.088796
multi_input_keras_model-20	50	128	0.010	0.046476	0.062759	0.011796	1738.0	0.943900	0.946271	0.961017	0.970684	0.003149	0.092510
multi_input_keras_model-21	50	128	0.100	0.483272	0.063905	0.079990	8191.0	0.404948	0.341038	0.626995	0.645586	0.075451	0.026562
multi_input_keras_model-22	100	128	0.001	0.046239	0.063255	0.013213	1869.0	0.936525	0.933138	0.952974	0.961332	0.004852	0.090828
multi_input_keras_model-23	100	128	0.010	0.128141	0.063174	0.041589	4655.0	0.766624	0.725195	0.817290	0.843002	0.030907	0.066034
multi_input_keras_model-24	100	128	0.100	0.699202	0.063795	0.077337	6973.0	0.536099	0.563166	0.700239	0.735321	0.041862	0.056041
multi_input_keras_model-25	150	128	0.001	0.030810	0.063205	0.006489	1030.0	0.969505	0.963067	0.980091	0.981384	0.001131	0.094514
multi_input_keras_model-26	150	128	0.010	0.099915	0.062740	0.033059	3988.0	0.828084	0.809225	0.868525	0.890400	0.018203	0.077922
multi_input_keras_model-27	150	128	0.100	0.932345	0.063632	0.079990	8191.0	0.404948	0.341038	0.626995	0.645586	0.075451	0.026562

Table 3.5 RMSprop Optimization function Results - Hamming loss performance metric (best model highlighted)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Finally, the table below summarizes the results presented above , including the best model neural network per optimization function:

Tag Name	Embedding Dimension tag	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
Adam model 7	100	32	0.001	10	0.019068	0.058487	0.003620	566.0	0.983106	0.983534	0.990600	0.992834	-0.000024	0.095667
SGD model 15	100	64	0.100	-	0.046036	0.078987	0.009035	1363.0	0.957116	0.950964	0.969073	0.972617	0.002822	0.092834
RMSprop model 16	150	64	0.001	-	0.020697	0.062347	0.004064	637.0	0.981206	0.982287	0.993508	0.994967	-0.001572	0.097220

Table 3.6 Summary table of the best model selected per optimizer (applying the first step of the comparison plan)

Based on all the four tables presented above, the model trained with Adam optimizer is the clear winner of the first comparison step. The three models selected per optimizer excelled among the other models of the same optimizer function. Adam's model estimator achieved the lowest hamming loss on the test dataset, with the RMSprop model estimator reaching very close. The model estimator of the SGD optimizer had the worse hamming loss and zero-one loss values among the three neural networks. Since the Adam optimizer and the RMSprop optimizer, both generated model estimators with very close score metrics, it is essential to proceed to the next five comparisons steps of the selection plan and decide whether or not Adam optimizer will remain the best estimator.

Selection plan Step 2: Classification Report - Confusion Matrix of the best estimators (per optimizer)

The first step of the selection plan is now complete. The model estimator optimized by Adam optimizer is the best one of the three models. The next step of the plan compares the relative classification reports and confusion matrices of the three best model classifiers. When using the tools of classification report and confusion matrices, two are the selection criteria for identifying the best of the three model estimators:

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- **Classification report:**
  1. The higher the metrics for each of the evaluation metrics of the classification report, the better the model estimator.
  2. The model estimator that can identify movies from all the seventeen different genres is a model classifier capable of understanding the semantics of each movie genre and classify the movies correctly.
- **Confusion Matrix:** Select the classifier with the most straight diagonal line. The diagonal line of a confusion matrix matches the predicted and the actual movie genres to each other.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Classification report for the best model estimator of the Adam optimizaion function:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1357
1	0.98	0.94	0.96	737
2	0.99	0.99	0.99	387
3	0.99	0.98	0.98	470
4	1.00	1.00	1.00	2869
5	0.98	0.94	0.96	986
6	0.98	0.99	0.98	854
7	1.00	1.00	1.00	4535
8	0.96	0.97	0.97	404
9	0.98	1.00	0.99	1019
10	0.93	0.94	0.93	201
11	0.96	0.98	0.97	483
12	0.98	1.00	0.99	1307
13	1.00	0.99	0.99	487
14	0.98	1.00	0.99	1210
15	0.76	0.70	0.72	312
16	0.97	0.92	0.95	226
micro avg	0.98	0.98	0.98	17844
macro avg	0.97	0.96	0.96	17844
weighted avg	0.98	0.98	0.98	17844
samples avg	0.99	0.99	0.98	17844

Figure 3.6 Adam optimizer Classification Report results

Classification report for the best model estimator of the SGD optimizaion function:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1357
1	0.90	0.98	0.94	737
2	0.80	0.75	0.77	387
3	0.97	0.98	0.98	470
4	1.00	1.00	1.00	2869
5	0.93	0.99	0.96	986
6	0.98	0.98	0.98	854
7	1.00	1.00	1.00	4535
8	0.93	0.80	0.86	404
9	0.98	0.99	0.99	1019
10	0.00	0.00	0.00	201
11	0.94	0.92	0.93	483
12	0.97	1.00	0.98	1307
13	1.00	0.96	0.98	487
14	0.98	1.00	0.99	1210
15	0.00	0.00	0.00	312
16	0.00	0.00	0.00	226
micro avg	0.97	0.94	0.96	17844
macro avg	0.79	0.78	0.78	17844
weighted avg	0.93	0.94	0.94	17844
samples avg	0.97	0.95	0.95	17844

Figure 3.7 SGD optimizer Classification Report results

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Classification report for the best model estimator of the SGD optimizaion function:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	1357
1	0.90	1.00	0.95	737
2	0.99	0.99	0.99	387
3	0.97	0.98	0.98	470
4	1.00	1.00	1.00	2869
5	0.93	1.00	0.96	986
6	0.98	1.00	0.99	854
7	1.00	1.00	1.00	4535
8	0.95	0.98	0.96	404
9	0.98	1.00	0.99	1019
10	0.89	0.93	0.91	201
11	0.93	0.97	0.95	483
12	0.97	1.00	0.98	1307
13	1.00	1.00	1.00	487
14	0.98	1.00	0.99	1210
15	0.73	0.68	0.70	312
16	0.93	1.00	0.96	226
micro avg	0.97	0.99	0.98	17844
macro avg	0.95	0.97	0.96	17844
weighted avg	0.97	0.99	0.98	17844
samples avg	0.98	0.99	0.98	17844

Figure 3.8 RMSprop optimizer Classification Report results

Based on the three classification reports, those that belong to the Adam Optimization function and the RMSprop optimization function seem to be closer to each other with more accurate results than the classification report of the model selected by the SGD optimization function. SGD is out of the competition. Looking closer between the first and the third classification reports, for the same number of movies per genre, Adam's optimizer model performed slightly better in F1-score values from its counterpart, the best model of the RMSprop optimizer. For example, this is noticeable for the War movies (index 15) with Adam optimizer hitting a score of 0.72 or 72%, and RMSprop 0.70 or 70%, with two points of difference. The reader should also expect to see the difference in the relative confusion matrices of the models selected by Adam and RMSprop optimizers. The latter optimizer will not predict as many movies from the war genre as the former optimizer, due to their difference in precision, recall, and f1-score for the genre mentioned above. However, there are cases where the RMSprop optimizer outperformed Adam optimizer. One such case is the sci-fi genre (index 13). The model estimator of the RMSprop optimizer classified correctly more movies than the model

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

estimator of the Adam optimizer, by 1% difference. The 1% difference means that the Adam optimizer classified incorrectly four movies of the sci-fi genre, while the RMSprop optimizer correctly classified all the 487 sci-fi movies. It is clear that for slight, although significant differences, the best model trained with the Adam optimizer is better in classifying the movies in the current dataset. However, it is also vital to compare the confusion matrices related to the generated classification reports to drive the conclusion of the better-performed classifier.

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	1342	0	0	1	4	1	1	4	0	1	0	0	1	0	0	0	2
Adventure	1	377	10	3	14	1	9	2	0	1	2	0	1	0	1	0	0
Animation	0	1	258	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Children	0	5	1	212	6	0	0	0	2	2	0	0	0	0	0	0	0
Comedy	3	1	0	0	2365	0	0	0	0	0	0	0	0	0	0	0	0
Crime	5	0	0	0	0	517	0	28	1	1	0	3	0	0	6	0	0
Documentary	3	0	1	1	3	0	768	2	0	1	0	0	0	0	0	0	0
Drama	0	3	0	1	7	8	11	2676	0	0	0	0	0	0	0	0	0
Fantasy	0	1	0	0	0	0	0	0	43	2	0	1	0	0	0	0	0
Horror	0	0	0	2	0	0	2	1	1	585	0	0	0	0	0	0	0
Musical	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0
Mystery	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0
Romance	0	0	0	0	2	0	0	1	0	0	1	0	55	0	0	0	0
Sci-Fi	0	2	0	0	0	0	1	0	1	0	0	0	0	60	0	0	0
Thriller	2	0	0	0	0	1	1	0	0	0	0	2	1	0	147	0	0
War	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	12	0
Western	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	90

Table 3.7 Adam optimizer Confusion Matrix

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	1350	0	0	0	3	0	1	3	0	0	0	0	0	0	0	0	0
Adventure	2	405	0	1	10	2	0	2	0	0	0	0	0	0	0	0	0
Animation	5	7	177	0	42	2	5	17	3	2	0	0	0	0	1	0	0
Children	3	13	47	159	1	0	0	2	0	2	0	0	0	0	1	0	0
Comedy	8	34	0	1	2326	0	0	0	0	0	0	0	0	0	0	0	0
Crime	2	0	0	0	0	554	0	2	0	3	0	0	0	0	0	0	0
Documentary	0	6	0	2	3	7	756	3	0	2	0	0	0	0	0	0	0
Drama	9	11	0	9	7	28	9	2633	0	0	0	0	0	0	0	0	0
Fantasy	0	1	3	0	0	0	0	0	40	2	0	0	1	0	0	0	0
Horror	8	0	0	0	0	4	0	1	1	577	0	0	0	0	0	0	0
Musical	17	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0
Mystery	0	0	0	0	0	2	0	0	0	0	0	68	0	0	0	0	0
Romance	1	0	0	0	2	0	0	1	0	0	0	0	55	0	0	0	0
Sci-Fi	0	3	0	0	0	0	1	0	1	0	0	0	0	59	0	0	0
Thriller	2	0	0	0	0	4	1	0	0	0	0	1	1	0	145	0	0
War	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Western	92	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.8 SGD optimizer Confusion Matrix

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	1347	0	0	1	3	0	4	1	0	0	0	0	1	0	0	0	0
Adventure	1	419	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
Animation	5	7	247	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Children	0	13	1	206	0	0	1	2	1	3	0	0	0	0	1	0	0
Comedy	9	28	0	1	2331	0	0	0	0	0	0	0	0	0	0	0	0
Crime	2	4	0	0	0	554	0	0	0	1	0	0	0	0	0	0	0
Documentary	0	7	1	0	3	7	757	3	0	1	0	0	0	0	0	0	0
Drama	9	11	0	8	7	28	10	2633	0	0	0	0	0	0	0	0	0
Fantasy	0	1	0	0	0	0	0	0	44	2	0	0	0	0	0	0	0
Horror	4	0	0	0	0	0	2	1	1	583	0	0	0	0	0	0	0
Musical	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0
Mystery	0	0	0	0	0	2	0	0	0	0	0	68	0	0	0	0	0
Romance	1	0	0	0	2	0	0	1	0	0	1	0	54	0	0	0	0
Sci-Fi	0	3	0	0	0	0	1	0	1	0	0	0	0	59	0	0	0
Thriller	2	0	0	0	0	4	1	0	0	0	0	1	1	0	145	0	0
War	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	13	1
Western	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	92

Table 3.9 RMSprop optimizer Confusion Matrix

From the three confusion matrices illustrated above, it is clear that the one belonging to the model optimized by the SGD optimizer classifies most of the movies incorrectly. Specifically, it assigns most movie texts per batch to either Action, Comedy, or Drama movie genres, the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

most data populated movie genres. Moreover, the model estimator of the SGD optimizer failed to identify movies from the Musical, War, and Western genres. The algorithm trained on movies coming mostly from the three dominant genres fails to identify particular details of the movies belonging to the rest of the genres. However, this is not the case for the best models optimized by the Adam and the RMSprop optimizers. From the first confusion matrix, the model estimator can identify most of the movies' correct genre, apart from very few cases in action, comedy, and drama movies, the most dominant movie genres. The best model of the counterpart optimizer also seems to follow accurate results, however, with some more miss-classified movies, especially in the adventure genre. The Adam optimizer presented the best results in both the classification report and the confusion matrix. Thus, the best model trained on 100 word-embedding dimensional space, 32 batch-size, 0.001 learning rate, ten decay steps multiplier, and optimized by the Adam algorithm is selected.

The third step in the selection plan involves comparisons between the test hamming loss and score (loss[1]) between the three selected models. Briefly, to summarize, the two previous steps of the selection plan, chose the model trained by the Adam optimizer. As already mentioned, the trained neural network with the most votes will proceed to the next section of comparisons between custom trained and pre-trained neural networks. The answer to the third step lies in the initial scoring table of the first step, referenced on page 90. The model estimator trained with the Adam optimizer achieved the lowest test hamming loss and test loss on never-seen-before data. All the three models had very close metric results on movies the neural networks never seen before. However, the Adam optimizer model stood out.

[1]binary crossentropy

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Selection plan Step 3: Dataframe of the movie title, the predicted genre(s) and the actual movie genre(s)

The next step of the selection plan is the random selection of 20 different movies and based on their content data, like actors, plot summary, and reviews select the model that could predict most accurately the genres of those movies.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

	Movie Title	Predicted Genre tags (top 3)	Real Genre tags
1	Devoured	[Drama, Horror, Thriller]	[Drama, Horror, Thriller]
2	Re Loca	[Comedy]	[Comedy]
3	Hero Wanted	[Action, Crime, Drama]	[Action, Crime, Drama]
4	Love and Lies	[Children, Romance]	[Children, Romance]
5	Oliver Twist	[Children, Crime, Drama]	[Children, Crime, Drama]
6	Hamlet	[Drama]	[Drama]
7	To Stay Alive - A Method	[Documentary]	[Documentary]
8	My Best Girl	[Comedy, Romance]	[Comedy, Romance]
9	Baby's Day Out	[Comedy]	[Comedy]
10	John Lennon - Plastic Ono Band	[Documentary]	[Documentary]
11	Social Lion	[Animation]	[Animation]
12	Love from a Stranger	[Drama, Mystery, Thriller]	[Drama, Mystery, Thriller]
13	Secrets of the Tribe	[Documentary]	[Documentary]
14	The House That Screamed	[Horror, Thriller]	[Horror, Thriller]
15	Bailey's Billion$	[Children, Comedy]	[Children, Comedy]
16	Tomorrow We Move	[Comedy]	[Comedy]
17	Belly 2: Millionaire Boyz Club	[Drama]	[Drama]
18	A Wedding	[Drama]	[Drama]
19	Martian Child	[Comedy, Drama]	[Comedy, Drama]
20	The Psycho Legacy	[Documentary]	[Documentary]

Table 3.10 Movie Predictions (20 random movies) | Adam optimizer

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

	Movie Title	Predicted Genre tags (top 3)	Real Genre tags
1	Devoured	[Drama, Horror, Thriller]	[Drama, Horror, Thriller]
2	Re Loca	[Comedy]	[Comedy]
3	Hero Wanted	[Action, Crime, Drama]	[Action, Crime, Drama]
4	Love and Lies	[Children, Romance]	[Children, Romance]
5	Oliver Twist	[Children, Crime, Drama]	[Children, Crime, Drama]
6	Hamlet	[Drama]	[Drama]
7	To Stay Alive - A Method	[Documentary]	[Documentary]
8	My Best Girl	[Comedy, Romance]	[Comedy, Romance]
9	Baby's Day Out	[Comedy]	[Comedy]
10	John Lennon - Plastic Ono Band	[Documentary]	[Documentary]
11	Social Lion	[Animation]	[Animation]
12	Love from a Stranger	[Drama, Mystery, Thriller]	[Drama, Mystery, Thriller]
13	Secrets of the Tribe	[Documentary]	[Documentary]
14	The House That Screamed	[Horror, Thriller]	[Horror, Thriller]
15	Bailey's Billion$	[Children, Comedy]	[Children, Comedy]
16	Tomorrow We Move	[Comedy]	[Comedy]
17	Belly 2: Millionaire Boyz Club	[Drama]	[Drama]
18	A Wedding	[Drama]	[Drama]
19	Martian Child	[Comedy, Drama]	[Comedy, Drama]
20	The Psycho Legacy	[Documentary]	[Documentary]

Table 3.11 Movie Predictions (20 random movies) | SGD optimizer

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

	Movie Title	Predicted Genre tags (top 3)	Real Genre tags
1	Devoured	[Drama, Horror, Thriller]	[Drama, Horror, Thriller]
2	Re Loca	[Comedy]	[Comedy]
3	Hero Wanted	[Action, Crime, Drama]	[Action, Crime, Drama]
4	Love and Lies	[Children, Romance]	[Children, Romance]
5	Oliver Twist	[Children, Crime, Drama]	[Children, Crime, Drama]
6	Hamlet	[Drama]	[Drama]
7	To Stay Alive - A Method	[Documentary]	[Documentary]
8	My Best Girl	[Comedy, Romance]	[Comedy, Romance]
9	Baby's Day Out	[Comedy]	[Comedy]
10	John Lennon - Plastic Ono Band	[Documentary]	[Documentary]
11	Social Lion	[Animation]	[Animation]
12	Love from a Stranger	[Drama, Mystery, Thriller]	[Drama, Mystery, Thriller]
13	Secrets of the Tribe	[Documentary]	[Documentary]
14	The House That Screamed	[Horror, Thriller]	[Horror, Thriller]
15	Bailey's Billion$	[Children, Comedy]	[Children, Comedy]
16	Tomorrow We Move	[Comedy]	[Comedy]
17	Belly 2: Millionaire Boyz Club	[Drama]	[Drama]
18	A Wedding	[Drama]	[Drama]
19	Martian Child	[Comedy, Drama]	[Comedy, Drama]
20	The Psycho Legacy	[Horror]	[Documentary]

Table 3.12 Movie Predictions (20 random movies) | RMSprop optimizer

The three tables above illustrate the three best classification models' predictions, one per optimization function, on data that each neural network has never seen before. The more accurate the results, the closer a model is to generalization. The latter describes a model's

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ability to make correct and accurate predictions based on data not used in its training. From the results of the above three tables, there is no clear winner. That is because all the three classification model estimators had very few misclassified movies as revealed by their respective zero-one loss. More precisely, the first model correctly classified 9233 movies, the second model 8,436, and the third model 9,162 movies out of the total 9,799 movies of the test sample. Thus, it is highly probable the 20 random movies to fall into those ranges of correctly classified movies by each estimator. There is no clear winner out of this comparison step. However, considering the zero-one loss value of each model estimator, Adam had the most movies correctly classified.

Selection plan Step 4: Learning Curves

The pre-final step, the fourth comparison presented by the selection plan, is to inspect the train and validation learning curves per training epoch and try to identify overfitting or underfitting classifiers and the goodness of fit per model estimator. The overfitting is a case scenario where the training error is continuously dropping, while the validation error is increasing. Underfitting is the opposite case; the model estimator is over-simplified and cannot recognize any particular pattern on the training dataset. One such scenario would be allocating all movies to the Action, Drama, or Comedy genre since those are the most movie-populated genres. *A model that is under-fitted will have high training and testing errors while an overfit model will have deficient training error but high testing error.* A learning curve is a kind of plot that demonstrates the time in the x-axis and the learning improvement on the y-axis. The reader may refer to the very well explained article of Jason Brownlee [18]. *Learning curves (LCs) are useful tools for monitoring the performance of workers exposed to a new task. Learning curves provide a mathematical representation of the learning process that takes place as task repetition occurs.* [2] Two are the different learning curves used to evaluate the performance of a machine learning model estimator and to drive the conclusion

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

of its overall goodness of fit. The first learning curve is the one graphed with the training dataset, **training learning curve**. The learning curve related to the training dataset can reveal how well the model estimator learned the dataset and its different semantics. On the other hand, **validation learning curve** is plotted based on the validation sample and can provide a reasonable estimation of how well the model estimator can generalize. Those two learning curves are plotted together in graphs that either measure the progress of the optimization metric (e.g., binary crossentropy), either on graphs that measure the performance metric (e.g., hamming loss) of the model estimator. The pre-final step of the selection plan, portrays both performance and optimization learning curves. Below the document presents some examples of underfitting, overfitting, and good-fitting curves.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

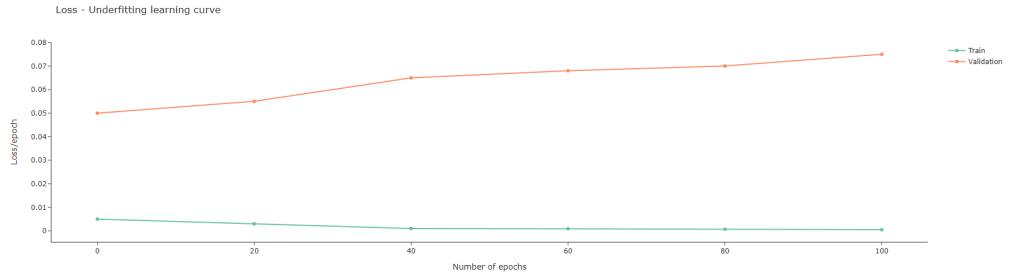


Figure 3.9 Underfitting Learning Curve - Part 1

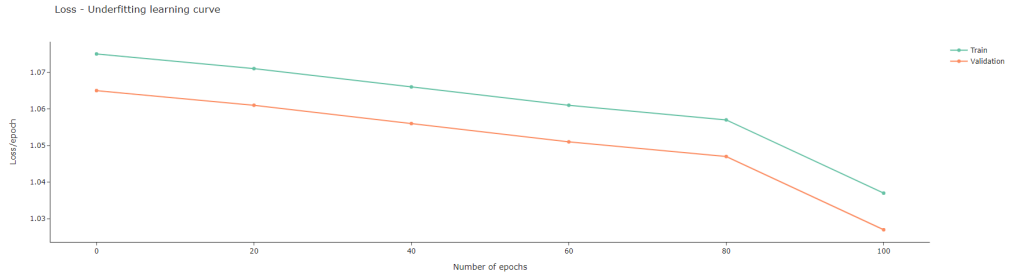


Figure 3.10 Underfitting Learning Curve - Part 2

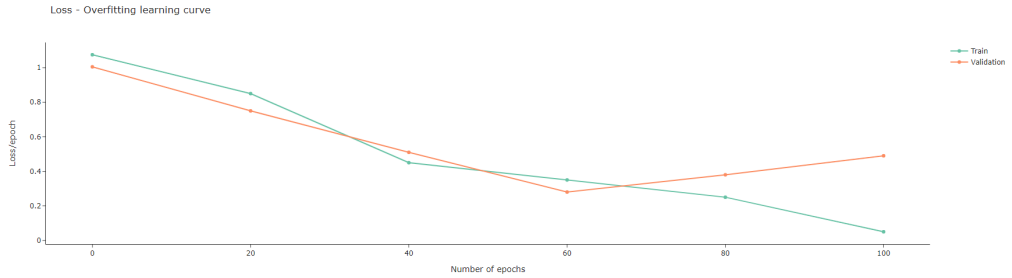


Figure 3.11 Overfitting Learning Curve

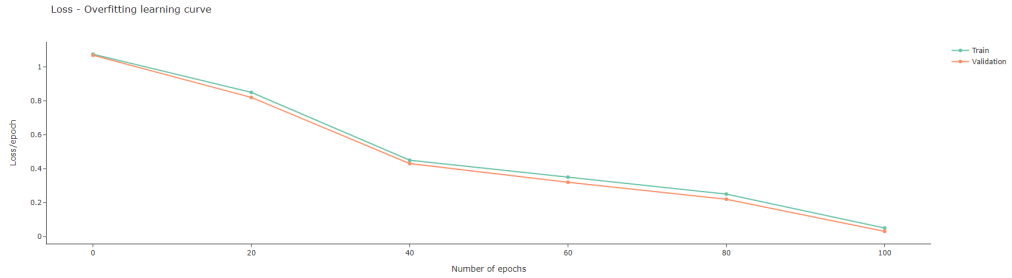


Figure 3.12 Good fitting Learning Curve

Underfitting: Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set [10]. Based on the plots above, two are the different occasions of an underfitting estimator; either the model estimator did not train for enough epochs, while

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

its loss could further improve. Either on cases were due to lots of noisy data, the model did not manage to learn the training data whatsoever; creating an important generalization gap between the training and the validation learning curves. Moreover, underfitting is a sign that the model classifier is oversimplified. A learning curve plot shows underfitting when: (1) The training loss remains flat regardless of training and there is a great generalization gap between the two learning curves (figure 3.9). (2) A continuous decrease in the training and validation losses until the end of training (figure 3.10) [18].

Overfitting: The overfitting occurs when the validation error is increasing while the training error, the error on the training dataset, keeps improving (decreasing per training epoch). This continuous decrease is a sign that the model estimator is more complicated than it should, and it gained too much flexibility with losing its ability to generalize. Use the learning curve of the validation dataset to identify overfitting model estimators. Another sign of the overfitting model is when training the model classifier for a considerable number of epochs. *The inflection point in validation loss may be the point at which training could be halted as the learning curve of the validation sample after that point shows the dynamics of overfitting.* [18].

Good fit: A right fit learning curve is the desired outcome for every model estimator. The model algorithms that have a well-fitted curve upon the training data are capable of generalizing, making good predictions on never-seen-before data, and can decrease the loss at a satisfying point without invoking overfit or underfit. *The training dataset will almost always have lower loss compared to the validation dataset. The difference in the loss will generate a gap between the two learning curves. This gap is the so-called "generalization gap.". A plot of learning curves shows a good fit if: (1) The plot of training loss decreases to a point of stability. (2) The plot of validation loss decreases to a point of stability and has a small gap with the training loss* [18]

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Below the document presents the learning performance curves for each of the three model estimators.

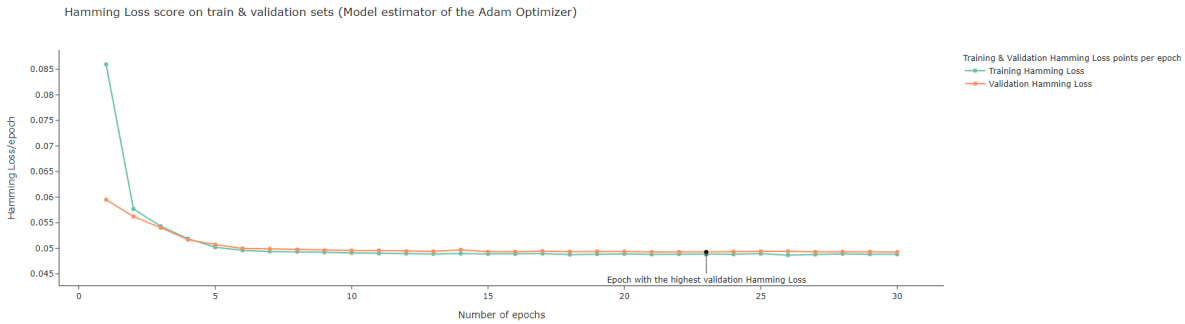


Figure 3.13 Adam optimizer Training - Hamming loss per training epoch

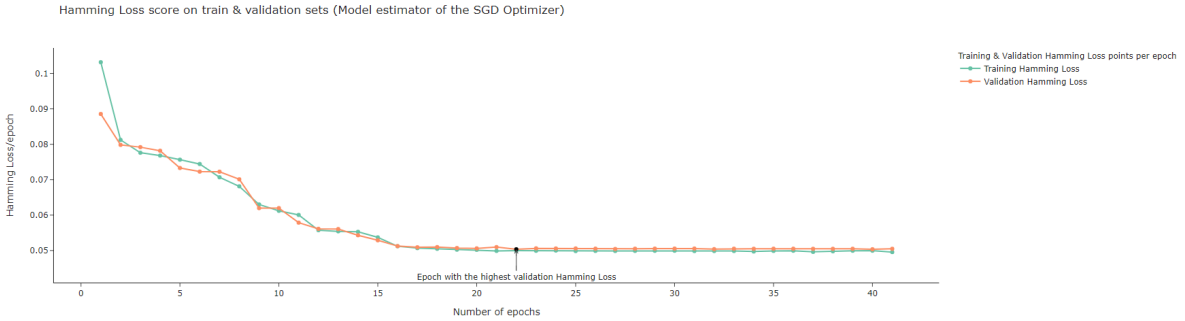


Figure 3.14 SGD optimizer Training - Hamming loss per training epoch

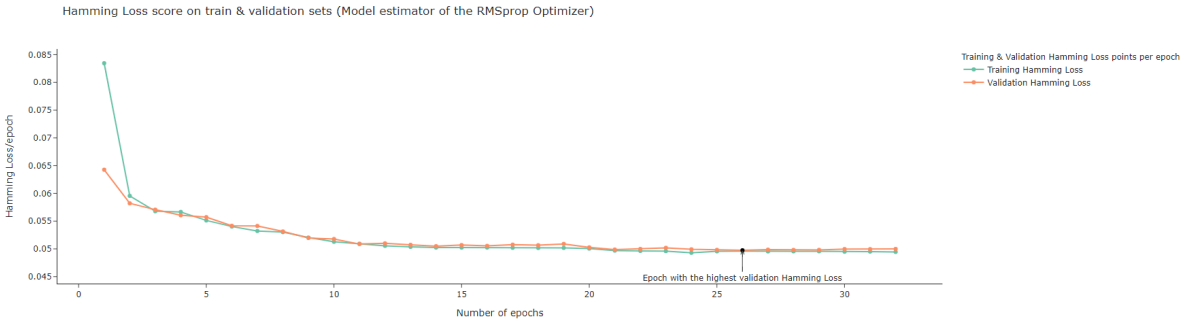


Figure 3.15 RMSprop optimizer Training - Hamming loss per training epoch

The above three plots present the hamming loss, which is the selected performance metric, and the relative learning curves of the Training & the Validation data. The relative learning curves of the first and the third models have the least number of spikes compared to the second graph of learning curves. The lack of spikes indicates a smoother training of the model estimator.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Moreover, it is crystal clear that Adam optimizer and the RMSprop optimizer learned the training data faster. The number of epochs in the x-axis can verify this accusation. The first and third models' training stopped at thirty epochs, while the training of the second model stopped after approximately forty epochs.

The second group of learning curves are those which monitor the lifetime progress (improvement) of the optimization metric, based on which the model parameters are updated. Those type of metrics are typically loss metrics. Recall that for the optimization of the model weights the *binary_crossentropy* loss was used.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

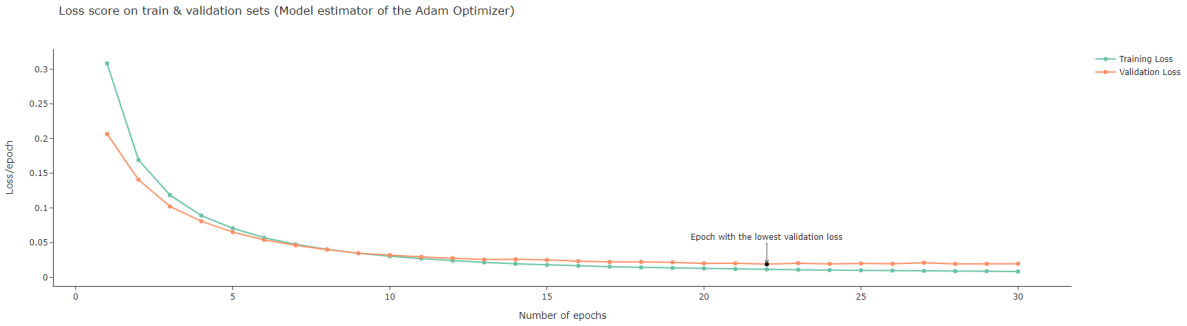


Figure 3.16 Adam optimizer Training - Loss per training epoch

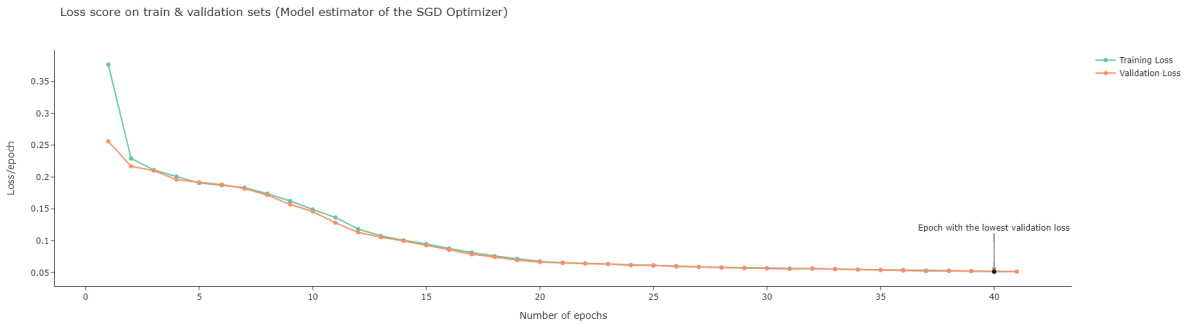


Figure 3.17 SGD optimizer Training - Loss per training epoch

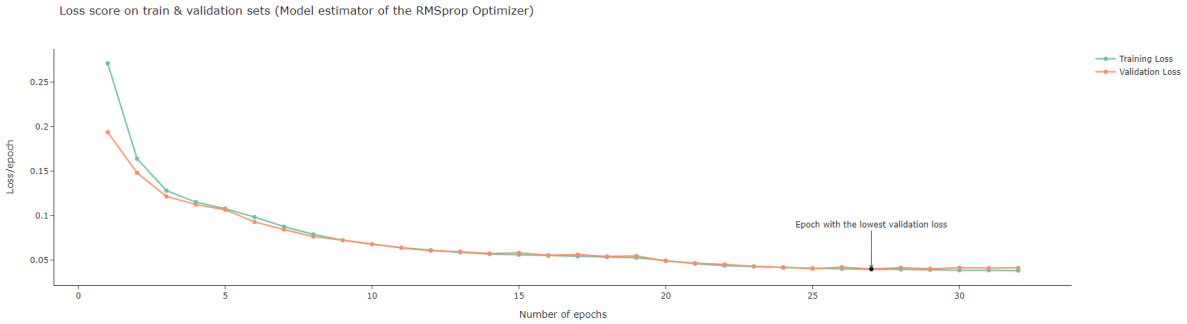


Figure 3.18 RMSprop optimizer Training - Loss per training epoch

Based on the three figures above, the model estimator of Adam optimizer has the best good fitted learning curves above all three model estimators. The reader can also observe that the training phase for the SGD and the RMSprop optimizers was longer than the training phase of the Adam optimizer model. Fortunately, the three figures present no significant gaps of generalization error. The first figure is an excellent example of right fitting learning curves.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

The second and the third figures present right fitting learning curves, but with a more unstable start on the first 4 to 10 epochs. Needless to say, that none of the three figures present signs of over-fitted or under-fitted model estimators. The first figure seems to have found the ideal spot where both the training and the validation loss are low. Based on the thorough analysis of the pre-final step of the selection plan, the model estimator of the Adam optimization algorithm presented the most well-fitted learning curves (both in performance and optimization metrics).

Selection plan Step 5: Bias-Variance trade-off

The final step of the selection plan compares the three model estimators, one chosen per optimization algorithm, based on the respective bias and variance.

Bias: The difference between the predictions of the neural network and the actual values of the target variable. A high biased model cannot correctly identify any special pattern or noise in the training data and leads to an oversimplified model estimator. Under-fitting machine learning models demonstrate high bias. Models unable to determine the mathematical function that fits the training data. The closer the bias is to 0, the more unbiased the model is.

$$Bias = \underbrace{(\mathbb{E}[\mathbb{E}[\hat{Y}] - Y]^2)}_{\text{SSE}} - \underbrace{\mathbb{E}[(X - m)^2]}_{\text{Variance}} \tag{3.9}$$

Variance: A measure of spread among the true and the predicted values. The higher the spread, the more uncertain the model is about the predictions made. High variance hurdles the model's ability to generalize on never-seen-before data because, with a high spread, all the never-seen data points will fall into the same target category. Even though the model will perform extremely well on the training data, the results will differ on first-seen data points. Model estimators with high variance are over bound to overfit.

$$Variance = \mathbb{E}[(X - m)^2] \tag{3.10}$$

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

The bias and the variance are two values commonly used together to describe a model estimator. The framework under which those two values interact is called the bias-variance trade-off. The below figure is regularly referenced in the bibliography and describes exactly their relationship.

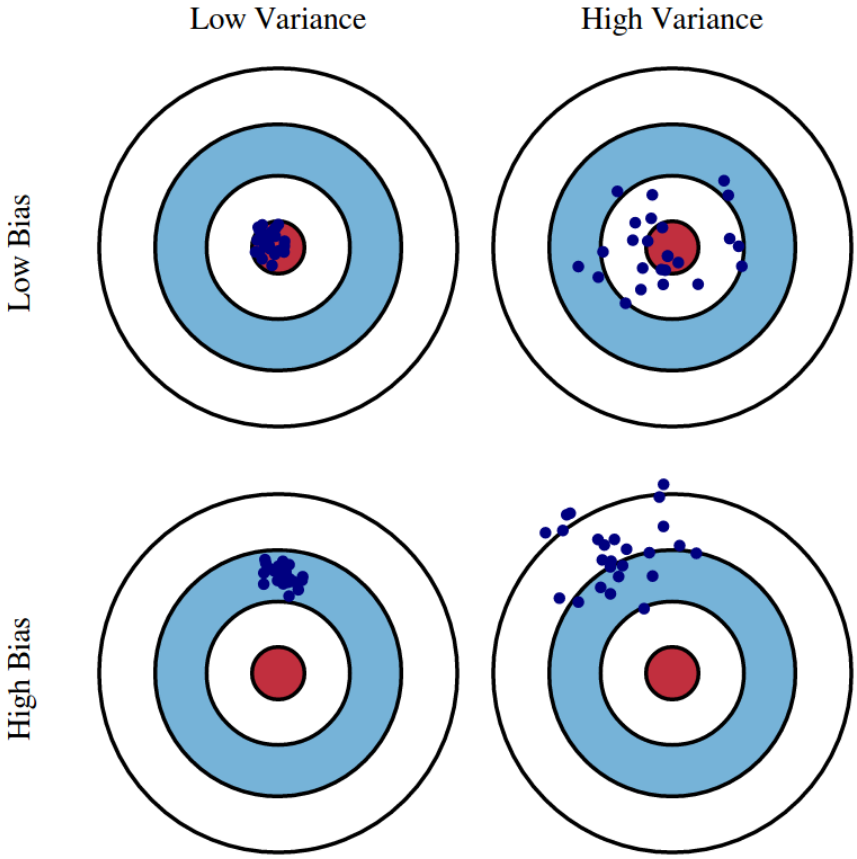


Figure 3.19 Bias-Variance trade-off | Bullseye diagram
Source: BiasVariance [22]

Based on the bulls-eye graph, a model of low variance and low bias reaches the ideal state. All the other three cases describe model estimators that either have been overfitting during the training phase (high variance - low variance), either have been underfitting (high bias - low variance). There is also the high bias and high variance case, which describes model

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

estimators that are non-representative to the data. Use the testing dataset to calculate the values of the variance and bias. Someone can potentially limit phenomena like overfitting and underfitting during the training phase. For the first phenomenon, the solution is either use **cross-validation scheme** with a hold-out dataset, *validation dataset*. Additionally, **reduce the model's complexity** up to a point where both training and validation learning curves are satisfyingly low. Whereas potential solutions to resolve or prevent underfitting could be the **replacement of the model algorithm** with another one. For example, if the dataset's distribution is polynomial, then an algorithm that tries to fit linear function on data would seriously underperform over machine learning algorithms that are robust in polynomial estimations. Moreover, if the model replacement is time-consuming, then the **increase of the current model's complexity** could substantially improve the prediction results.

The reader can review the values of the bias and variance per model estimator in the table already presented on page 90. The bias of all three models is very close to zero, and the bias for the first and third models is negative. Additionally, the variance is very close to 0.01, another good sign of well-fitted models. Out of the three model estimators, the first model stands out with the lowest bias and variance among all the others.

The results of the table indicate that all three model estimators are unbiased, biased very close to zero, and their spread of the prediction values is not high whatsoever. Thus, all three model estimators achieved a sufficiently good fit upon the training and the test data. In conclusion, the document presents a summary of the outcome of each step executed in the selection plan.

Step 1: Among the 108 different models trained, select the three models, one per optimizer, with the lowest hamming loss & zero-one loss values. The outcome was three different model estimators, one per optimization algorithm.

- Adam optimizer: 7th model, with word embeddings dimension 100, batch size value equal to 32, learning rate equal to 0.001 and 10 decay steps multiplier.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- SGD optimizer: 15th model, with 100-word embeddings dimension, 64 hidden units on the dense layer(s), and 0.1 learning rate.
- RMSprop optimizer: 16th model, with 150-word embeddings dimension, 64 hidden units on the dense layer(s), and 0.001 learning rate.

Step 2: Execute a round of five different comparisons. Each round highlights one model. The first round compared the classification reports and the relative confusion matrices per neural network. From the second step, the model trained with Adam optimizer stood out from the other two.

Step 3: The second comparison round compares the hamming loss and loss values on content about movies never seen before by the three classification model estimators. From the third step of the execution plan, the first model classifier, optimized with Adam stood out from the other two. With the model estimator from the RMSprop optimization function reaching very close to the results of the first model estimator.

Step 4: The third comparison round, representing the fourth step in the execution plan sequence, brings the comparison down to the actual predictions on movies never-seen-before by the model estimators. Twenty random movies compared. The results were outstanding and very accurate for all three classification models.

Step 5: The fourth comparison round, representing the fifth step in the execution plan sequence, compared the learning curves, one for the training dataset, and the other for the validation dataset, for the performance and optimization metrics accordingly. The graphs presented well-fitted and smooth learning curves, with the model estimator of Adam's optimizer being the one with the least spikes. The gap of the generalization error between the training and the validation learning curves determines the goodness of fit. In contrast, the smoothness is determined by the steady improvement of the optimization metric through each training epoch. The graphs of the second model estimator, especially the performance learning graph,

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

presented lots of spikes (non-smoothness) but rather a small generalization error gap. The outcome of the pre-final step in the execution plan proposed that the first model estimator had the best-fitted learning curves, without implying that the other two model estimators were under-fitted or over-fitted. The examination of the learning curves is a tool for judging the performance of a model estimator.

Step 6: The final step of the execution plan, and the fifth comparison round selected the best model estimator by comparing the bias-variance trade-off of each model estimator. Even though the first and the third model estimators had very close results, the first model estimator trained with Adam optimizer had the lowest bias and variance values overall. Based on the conclusion and facts written above, the neural network which optimized the binary crossentropy loss using the Adam algorithm is selected to proceed in the next chapter.

3.2.4 Hamming loss model estimator vs. Binary accuracy model estimator

During the documentation of the model compilation strategy, referenced on page 74, the authors chose the hamming loss as the performance metric after explaining the reason why binary accuracy is not appropriate for the current classification experiment. However, the authors trained a model estimator after setting the binary accuracy as the performance metrics. The goal is to compare those two model estimators, the one trained with hamming loss and the other trained with binary accuracy.

The two summary tables have many differences, review the first table on page 77. Specifically, the inputs were four and not five, movie title was absent from the training, and the words tokenized per independent variable were 20,000 tokens. Additionally, the early stopping patience was much higher, equal to 75 epochs, a value that allowed the Adam optimizer train for a high number of epochs, approximately 80 epochs of training. Last but not least, the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Parameters	Value	Description
Number of input layers	4	[actors, plot summary, movie features, movie reviews]
Number of output targets	17	-
Input length of embedding layer	(17, actors), (20, plot summary), (58, movie features, (255 movie reviews)	
Actor names tokenized	20000	-
Plot words tokenized	20000	-
Movie features words tokenized	20000	-
Movie reviews words tokenized	20000	-
Activation function of output layer	sigmoid	-
Loss function	binary crossentropy	-
Metric	binary accuracy	
Optimization function(s)	[Adam, SGD, RMSprop]	-
Batch size (hyper parameter)	[16, 32, 64, 128]	-
Embedding dimension size (hyper parameter)	[50, 100, 200, 300]	-
Learning rate	0.01	-
Decay steps multiplier	1000	Every how many epochs the value of the learning rate is decreased
Validation split ratio	80% training, 20% validation dataset	-
Number of models trained	48	16 models with Adam optimizer, 16 models with SGD optimizer. 16 models with RMSprop optimizer
Train-test split ratio	80% training, 20% test dataset	
Epochs	150	-
Early stopping rounds	75	-
Minimum delta	0.01	If the validation loss drops below 0.009 units, it is not considered an improvement

Table 3.13 Summary table of training the custom neural network with binary accuracy

hyper-parameters of the neural network were two instead of four, tuned on the hamming loss performance metric model. Below the document presents the scoring table and the learning curves of the model trained with the binary accuracy performance metric.

Tag Name	Embedding tag	Batch tag	Keras Model	Test Loss	Test Accuracy	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
model sixteen adam	300	128	multi_input_model_300dim_128batchsize	0.036827	0.996062	0.003938	623.0	0.981737	0.981821	0.990968	0.992668	-0.00047	0.096426

Table 3.14 Scoring table of the model estimator trained on binary accuracy metric

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

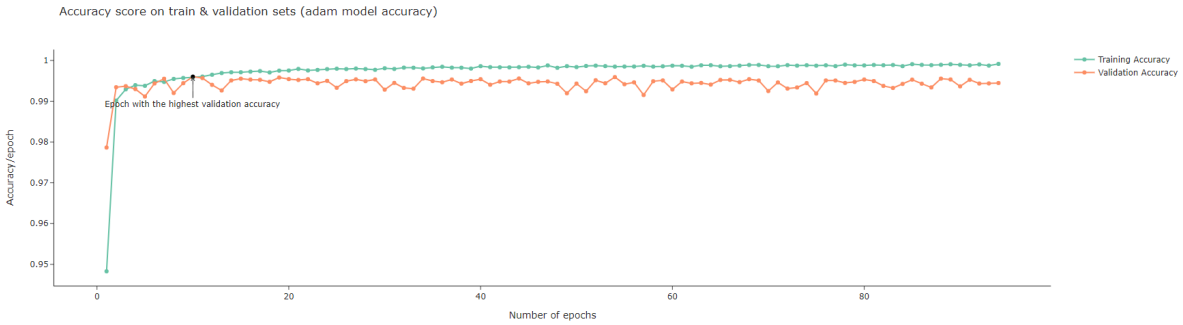


Figure 3.20 Performance Learning curves - Binary accuracy per training epoch

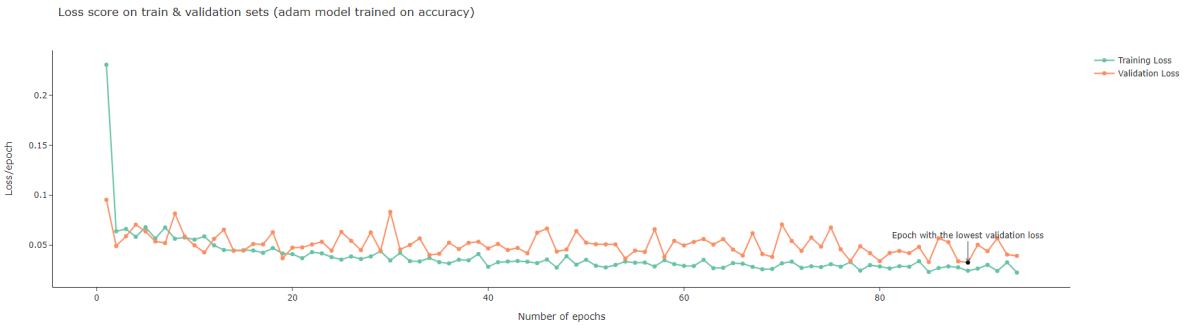


Figure 3.21 Optimization Learning curves - Loss per training epoch

The table and the two figures above present a well-fitted model estimator with high accuracy and low binary crossentropy loss value. The zero-one loss is very close to the model estimator of the hamming loss performance metric, 623.0 against 566.0. The model classifier of the hamming loss performance metric predicted correctly 57 more movies. Moreover, it is crystal clear that the second model estimator's learning curves have more spikes and less smoothness than the learning curves of the first model estimator, as presented in pages 105, 107. A fact that indicates how well fitted is the model estimator trained on the hamming loss performance metric. The lack of smoothness and the spikes of the learning curves, along with the relative numbers of the score metrics accomplished with the second model estimator, indicate that the imbalance of negative and positive values of the target variable y has affected the performance of the binary accuracy metric. The authors cannot ignore that even with the binary accuracy metric, the model classifier can still make correct predictions. However, someone should always check for imbalance issues among the labels of the target variable and not use the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

binary accuracy if extreme imbalance is observed. A model classifier, trying to correctly classify an imbalanced target variable and measure the classification performance with binary accuracy, is very susceptible to learn and correctly predict only the dominant values of the target feature y.

3.3 Neural networks of pre-trained word embeddings

This section of the third chapter documents the training of different neural networks of either pre-trained word embeddings, or saved network structures and selects the best model estimator for each pre-trained classifier. The next subsection compares the model estimators of those neural networks with the model estimator selected from the previous chapter. The selection is made by following a shortened version of the selection plan mentioned above. The assembling of the conversational agent that will take place in the next section will exploit the word embeddings of the best model classifier chosen at the end of this chapter.

3.3.1 GloVe pre-trained word embeddings

GloVe embeddings stand for Global Vectors of Word Representation. *Perform the training on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space* [3]. The pre-trained word vectors used in the current experiment, contain 42 billion tokens, 1.9 million words in the vocabulary, and 300-dimensional vectors. The neural network assembled with the pre-trained GloVe embeddings has the same structure with the neural network trained on custom word-embeddings—five separate Embedding layers of 300 dimensions each. The GloVe embeddings neural network's training process is the same as the training process already documented in section Compiling & Training the neural network.The execution of

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

the GloVe embeddings training process yielded eighteen different neural network model estimators.

Tag Name	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
glove_embeddings-1	32	0.001	10	0.195429	0.065127	0.060756	6356.0	0.649513	0.639765	0.757214	0.792683	0.035471	0.061845
glove_embeddings-2	32	0.001	20	0.207555	0.066411	0.065031	6633.0	0.612678	0.599929	0.734847	0.771462	0.040704	0.057086
glove_embeddings-3	32	0.010	10	0.296977	0.081439	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-4	32	0.010	20	0.293504	0.088953	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-5	32	0.100	10	0.292655	0.093461	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-6	32	0.100	20	0.292095	0.096466	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-7	64	0.001	10	0.193746	0.091701	0.058992	6169.0	0.649549	0.638848	0.751520	0.789985	0.040285	0.057466
glove_embeddings-8	64	0.001	20	0.190848	0.087977	0.057641	6178.0	0.669535	0.660289	0.767565	0.802285	0.034453	0.062776
glove_embeddings-9	64	0.010	10	0.296258	0.090590	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-10	64	0.010	20	0.294165	0.092680	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-11	128	0.001	10	0.209676	0.068224	0.065421	6722.0	0.594960	0.567999	0.720721	0.754117	0.046983	0.051440
glove_embeddings-12	128	0.001	20	0.207239	0.066868	0.063968	6590.0	0.608379	0.584024	0.728266	0.763978	0.045171	0.053063
glove_embeddings-13	128	0.010	10	0.293262	0.081739	0.102249	8346.0	0.325025	0.308569	0.603853	0.620729	0.057181	0.042400
glove_embeddings-14	128	0.010	20	0.291706	0.088962	0.107130	9797.0	0.000896	0.000561	0.500191	0.500195	0.106987	0.000108
glove_embeddings-15	128	0.100	10	0.299538	0.093468	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-16	128	0.100	20	0.296394	0.096472	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-17	64	0.100	10	0.292272	0.098618	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
glove_embeddings-18	64	0.100	20	0.292156	0.100228	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000

Table 3.15 Trained Neural Networks on pre-trained GloVe word embeddings

The seventh and the eighth trained models accomplished the best results compared to the others. The two models managed to achieve the lowest hamming loss and zero-one loss on test data. Even though the seventh model correctly classified nine more movies than the eighth model, the latter had better performance on the rest of the score metrics of f1-score, ROC-AUC score and bias. As a result the eighth model is a better model classifier and is the one selected among the eighteen total trained models.

From now on, the document will present a set of saved models hosted in Tensorflow Hub. The latter is an online cloud community with available pre-trained neural networks of various authors. It supports an alternative way for text representation, such as to convert sentences into embeddings vectors using pre-trained text embeddings as the first layer. The differences, and at the same time, the advantages of using such models for making predictions, against a neural network with custom word embeddings are the following:

- The first layer that stores the neural network takes care of the text preprocessing, by

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

mapping a corpus of sentences to an embedding vector.

- The benefit of transfer learning.

Transfer learning - Definition 1:

The classic supervised machine learning paradigm is based on learning in isolation, a single predictive model for a task using a single dataset. This approach requires a large number of training examples and performs best for well-defined and narrow tasks. Transfer learning refers to a set of methods that extend this approach by leveraging data from additional domains or tasks to train a model with better generalization properties. [14]

Transfer learning - Definition 2:

Pre-training allows a model to capture and learn a variety of linguistic phenomena, such as long-term dependencies and negation, from a large-scale corpus. Then this knowledge is used (transferred) to initialize and then train another model to perform well on a specific NLP task, such as sentiment classification [21]

Those two are the reasons that make pre-trained saved models so commonly used and an excellent opportunity to compare the performance of custom neural networks with the performance of pre-trained models. Please note that the results presented from now on bound to the specific experiment and dataset. Someone should not use the results and conclusions derived from this documentation to drive general decision-making. Besides, the thesis presents the results of only four out of many more pre-trained neural networks found on the Tensorflow Hub. Thus, the results apply only to the current experiment and study.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3.3.2 English Google News 130GB corpus | 20 dimensions & without OOV token

This is the first neural network, which represents a saved model stored in Tensorflow hub and the second pre-trained model estimator after the neural network developed on GloVe embeddings, with both models to contribute in the comparison with the custom estimator's performance. In those kind of models, the first layer is a TensorFlow Hub layer. This layer uses a pre-trained Saved Model to map a sentence into an embedding vector. The model [12], which is trained on English Google News corpus, splits the sentence into tokens, embeds each token and then combines the embedding. The resulting dimensions are: (num_examples, embedding_dimension). The embedding dimension of the vector space has a fix value equal to 20. The neural network structure of the second model estimator,

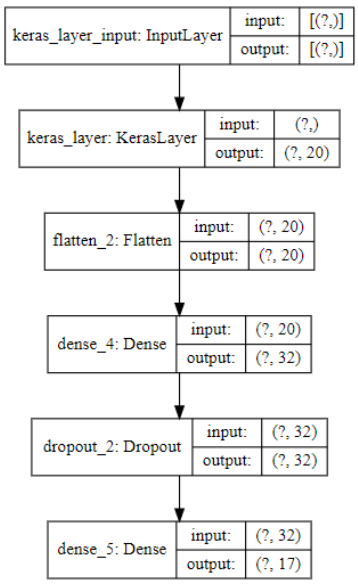


Figure 3.22 Neural Network Structure | English Google News 130Gb corpus with 20 dim & without OOV token

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Parameters	Value	Description
Loss function	binary crossentropy	
Metric	[hamming loss, f1 score]	
Optimization function(s)	Adam	
Batch size (hyper parameter)	[32, 64, 128]	
Embedding dimension size	20	
Learning rate (hyper parameter)	[0.001, 0.01, 0,1]	
Decay steps multiplier (hyper parameter)	[10, 20]	
Number of models trained	18	

Table 3.16 Summary table of English Google News 20dim without OOV token

Tag Name	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
English_Google_News_130GB_20dim_without_OOV-1	32	0.001	10	0.244004	0.095754	0.093869	8009.0	0.387289	0.359360	0.629284	0.656543	0.055407	0.043961
English_Google_News_130GB_20dim_without_OOV-2	32	0.001	20	0.240941	0.094614	0.092542	7911.0	0.418571	0.390544	0.644995	0.671417	0.049339	0.049337
English_Google_News_130GB_20dim_without_OOV-3	32	0.010	10	0.248332	0.094409	0.092434	7761.0	0.443231	0.427533	0.659358	0.687790	0.042536	0.055432
English_Google_News_130GB_20dim_without_OOV-4	32	0.010	20	0.251306	0.094406	0.094205	7846.0	0.450852	0.433057	0.666084	0.693607	0.037187	0.060279
English_Google_News_130GB_20dim_without_OOV-5	32	0.100	10	0.286514	0.097091	0.102706	9011.0	0.214931	0.167367	0.560222	0.575266	0.079457	0.023144
English_Google_News_130GB_20dim_without_OOV-6	32	0.100	20	0.282675	0.098707	0.105119	9281.0	0.145847	0.105837	0.537985	0.549373	0.088260	0.015696
English_Google_News_130GB_20dim_without_OOV-7	64	0.001	10	0.254082	0.098690	0.095856	8053.0	0.372278	0.347672	0.623067	0.649612	0.055921	0.043509
English_Google_News_130GB_20dim_without_OOV-8	64	0.001	20	0.250829	0.098499	0.095034	7856.0	0.410281	0.392241	0.642564	0.670968	0.047348	0.051114
English_Google_News_130GB_20dim_without_OOV-9	64	0.010	10	0.265364	0.098485	0.096270	7952.0	0.401225	0.380535	0.638567	0.667143	0.047720	0.050781
English_Google_News_130GB_20dim_without_OOV-10	64	0.010	20	0.261058	0.098401	0.095934	7909.0	0.417538	0.406960	0.647510	0.674964	0.043826	0.054271
English_Google_News_130GB_20dim_without_OOV-11	64	0.100	10	0.285952	0.098879	0.098882	8408.0	0.313781	0.273496	0.597478	0.619243	0.064952	0.035611
English_Google_News_130GB_20dim_without_OOV-12	64	0.100	20	0.278998	0.099263	0.099548	8526.0	0.271621	0.236779	0.580485	0.601343	0.072980	0.028679
English_Google_News_130GB_20dim_without_OOV-13	128	0.001	10	0.264633	0.099265	0.097345	8007.0	0.407743	0.386479	0.643119	0.669649	0.044163	0.053968
English_Google_News_130GB_20dim_without_OOV-14	128	0.001	20	0.265379	0.099254	0.097105	8009.0	0.410324	0.391520	0.644388	0.671035	0.043856	0.054244
English_Google_News_130GB_20dim_without_OOV-15	128	0.010	10	0.281693	0.099206	0.098227	8058.0	0.421168	0.403613	0.651774	0.680234	0.038962	0.058665
English_Google_News_130GB_20dim_without_OOV-16	128	0.010	20	0.278330	0.099231	0.099128	8060.0	0.404013	0.380169	0.642491	0.667984	0.042237	0.055702
English_Google_News_130GB_20dim_without_OOV-17	128	0.100	10	0.301855	0.099447	0.100514	8260.0	0.375829	0.347333	0.628004	0.652266	0.047462	0.051012
English_Google_News_130GB_20dim_without_OOV-18	128	0.100	20	0.300186	0.099504	0.099170	8169.0	0.382245	0.351336	0.630483	0.655725	0.047966	0.050562

Table 3.17 Trained Neural Networks on pre-trained English Google news 20dim without OOV token

The third model trained presented the best performance compared to all the other eighteen models. Thus, it is the one selected to continue in the next chapter. The overall insight gained by reviewing the table is that all the trained neural networks ill-performed, misclassifying most movies. Additionally, all the models have high testing losses, with values indicating the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

models' inability to classify 1 in 4 movies correctly.

3.3.3 English Google News 130GB corpus | 20 dimensions & OOV token

The only difference to the above pre-trained saved model is the conversion of 2,5% vocabulary to OOV buckets. The Out-Of-Vocabulary tokens are useful in situations where there is a gap between the vocabulary size of the corpus and the vocabulary size of the saved model. The vocabulary size of the current saved model is 20,000 tokens. So there is a very high chance that 20,000 tokens from the English Google News corpus do not contain many case-specific words related to movie content. The saved model used can be found here [12].

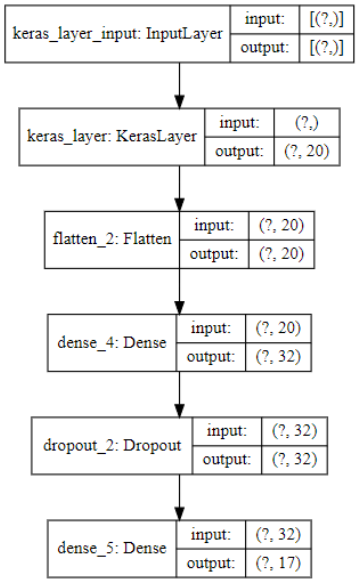


Figure 3.23 Neural Network Structure | English Google News 130Gb corpus with 20 dim & with OOV token

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Parameters	Value	Description
Loss function	binary crossentropy	
Metric	[hamming loss, f1 score]	
Optimization function(s)	Adam	
Batch size (hyper parameter)	[8, 16, 32, 64, 128]	
Embedding dimension size	20	
Learning rate (hyper parameter)	[0.001, 0.01, 0,1]	
Decay steps multiplier (hyper parameter)	[10, 20]	
Number of models trained	30	

Table 3.18 Summary table of English Google News 20dim with OOV token

Tag Name	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
model_english_google_news_130GB_20dim_with_oov-1	8	0.001	10	0.241678	0.095646	0.095256	8114.0	0.359179	0.330955	0.616301	0.642992	0.060141	0.039804
model_english_google_news_130GB_20dim_with_oov-2	8	0.001	20	0.246518	0.095760	0.095082	8114.0	0.366617	0.337048	0.619777	0.645735	0.058604	0.041151
model_english_google_news_130GB_20dim_with_oov-3	8	0.010	10	0.249060	0.096049	0.095238	8013.0	0.386575	0.363092	0.629923	0.655536	0.053301	0.045821
model_english_google_news_130GB_20dim_with_oov-4	8	0.010	20	0.253008	0.096109	0.095304	8107.0	0.379844	0.351454	0.626508	0.654513	0.054919	0.044392
model_english_google_news_130GB_20dim_with_oov-5	8	0.100	10	0.295467	0.099186	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
model_english_google_news_130GB_20dim_with_oov-6	8	0.100	20	0.294864	0.101238	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
model_english_google_news_130GB_20dim_with_oov-7	16	0.001	10	0.248089	0.100614	0.094595	8005.0	0.394529	0.370111	0.633612	0.660212	0.052303	0.046704
model_english_google_news_130GB_20dim_with_oov-8	16	0.001	20	0.244693	0.099933	0.093995	7910.0	0.400077	0.379538	0.636093	0.663331	0.051852	0.047105
model_english_google_news_130GB_20dim_with_oov-9	16	0.010	10	0.259078	0.099609	0.095766	7836.0	0.431245	0.425091	0.655495	0.685294	0.040239	0.057508
model_english_google_news_130GB_20dim_with_oov-10	16	0.010	20	0.260606	0.099279	0.095760	7831.0	0.439966	0.430694	0.660874	0.689186	0.037721	0.059792
model_english_google_news_130GB_20dim_with_oov-11	16	0.100	10	0.297577	0.100389	0.104194	9291.0	0.119872	0.093309	0.530784	0.538612	0.093690	0.011141
model_english_google_news_130GB_20dim_with_oov-12	16	0.100	20	0.293139	0.101315	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
model_english_google_news_130GB_20dim_with_oov-13	32	0.001	10	0.255035	0.101144	0.096979	8052.0	0.392692	0.368949	0.634472	0.660689	0.048814	0.049805
model_english_google_news_130GB_20dim_with_oov-14	32	0.001	20	0.248896	0.100824	0.094998	7829.0	0.429956	0.414838	0.653952	0.681687	0.041920	0.055988
model_english_google_news_130GB_20dim_with_oov-15	32	0.010	10	0.261797	0.100585	0.095868	7927.0	0.408299	0.386278	0.642171	0.669472	0.046481	0.051889
model_english_google_news_130GB_20dim_with_oov-16	32	0.010	20	0.270825	0.100415	0.096450	7878.0	0.398262	0.387066	0.637086	0.664222	0.048212	0.050342
model_english_google_news_130GB_20dim_with_oov-17	32	0.100	10	0.282603	0.100908	0.101559	8782.0	0.226075	0.191506	0.564045	0.582313	0.079007	0.023527
model_english_google_news_130GB_20dim_with_oov-18	32	0.100	20	0.292240	0.101349	0.101373	8706.0	0.225758	0.196074	0.563928	0.581392	0.079336	0.023247
model_english_google_news_130GB_20dim_with_oov-19	64	0.001	10	0.257157	0.101117	0.095142	7907.0	0.419599	0.400248	0.647978	0.675308	0.044595	0.053580
model_english_google_news_130GB_20dim_with_oov-20	64	0.001	20	0.262654	0.100953	0.095310	7905.0	0.410413	0.393142	0.642878	0.670915	0.046845	0.051563
model_english_google_news_130GB_20dim_with_oov-21	64	0.010	10	0.266375	0.100772	0.096865	7914.0	0.438201	0.421239	0.660921	0.687624	0.036356	0.061037
model_english_google_news_130GB_20dim_with_oov-22	64	0.010	20	0.273109	0.100668	0.098329	7992.0	0.421855	0.398967	0.652284	0.677744	0.038598	0.058996
model_english_google_news_130GB_20dim_with_oov-23	64	0.100	10	0.292396	0.100782	0.098275	8428.0	0.311362	0.273926	0.596215	0.621295	0.066434	0.034325
model_english_google_news_130GB_20dim_with_oov-24	64	0.100	20	0.291094	0.100839	0.099578	8357.0	0.315451	0.283354	0.598469	0.620313	0.063496	0.036877
model_english_google_news_130GB_20dim_with_oov-25	128	0.001	10	0.265040	0.100745	0.096829	7962.0	0.415791	0.396643	0.647305	0.673679	0.042807	0.055188
model_english_google_news_130GB_20dim_with_oov-26	128	0.001	20	0.266817	0.100657	0.096679	7955.0	0.421287	0.404050	0.650397	0.677542	0.041522	0.056347
model_english_google_news_130GB_20dim_with_oov-27	128	0.010	10	0.276336	0.100599	0.098419	8018.0	0.425039	0.404483	0.654305	0.679934	0.037543	0.059955
model_english_google_news_130GB_20dim_with_oov-28	128	0.010	20	0.277498	0.100538	0.097681	7932.0	0.416816	0.402146	0.648677	0.675203	0.041095	0.056733
model_english_google_news_130GB_20dim_with_oov-29	128	0.100	10	0.292564	0.100577	0.100851	8188.0	0.391348	0.362819	0.636693	0.660524	0.042854	0.055146
model_english_google_news_130GB_20dim_with_oov-30	128	0.100	20	0.303202	0.100598	0.100809	8232.0	0.350443	0.323633	0.615238	0.638629	0.053363	0.045767

Table 3.19 Trained Neural Networks on pre-trained English Google news corpus 20dim with OOV token

The eighth and the fourteenth model classifiers demonstrated the best performance out of the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

thirty total models. However, the overall score metrics monitored do not achieve favorable results. For example, the ROC-AUC score cannot climb above the 0.65 units in the majority of the models.ROC-AUC score values around 0.5 and 0.65 units indicate a poor-fitted model classifier. Additionally, test loss follows the same pattern with the previous pre-trained model estimator. Nonetheless, the table's overall results do not present a promising chance to choose one of those models instead of the custom-trained neural network of the previous chapter. Out of the two best performed models, the eighth is the one to continue in the next chapter, as the one having the lowest hamming loss value.

3.3.4 English Google News 7B corpus | 50 dimensions & OOV token

The fourth and the last saved pre-trained model that will be compared to the custom model estimator created on the previous section, is a token based text embedding model trained on English Google News of 7 billion tokens. Each word in the corpus is embedded to 50-dimensional word vector space. The use of a higher dimensional word representation space will substantially increase both the trainable parameters of the neural network and also the information learned from a 50-vector space of a token would be more helpful in distinguishing and better learning the patterns between different movie genres. This saved model also makes use of Out-Of-Vocabulary token, so as to help the neural network to derive the semantics of the text, if the latter contains tokens that do not overlap with the tokens in the vocabulary of the saved model estimator. The saved model used can be found here [1].



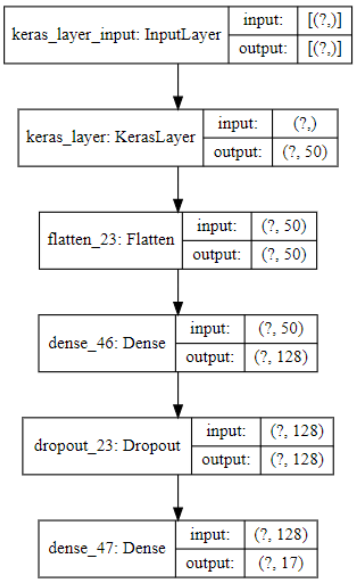


Figure 3.24 Neural Network Structure | English Google News 7B corpus with 50 dim & with OOV token

Parameters	Value	Description
Loss function	binary crossentropy	
Metric	[hamming loss, f1 score]	
Optimization function(s)	Adam	
Batch size (hyper parameter)	[16, 32, 64, 128]	
Embedding dimension size	50	
Learning rate (hyper parameter)	[0.001, 0.01, 0,1]	
Decay steps multiplier (hyper parameter)	[10, 20]	
Number of models trained	24	

Table 3.20 Summary table of English Google News 50dim 7B with OOV token

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Tag Name	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
model_english_google_news_7B_50dim-1	16	0.001	10	0.250967	0.096559	0.095646	8318.0	0.365042	0.321783	0.619362	0.642709	0.058067	0.041622
model_english_google_news_7B_50dim-2	16	0.001	20	0.257931	0.096757	0.095646	8356.0	0.354390	0.311635	0.614258	0.637092	0.060664	0.039347
model_english_google_news_7B_50dim-3	16	0.010	10	0.255495	0.096382	0.094013	7838.0	0.415525	0.399308	0.644615	0.672656	0.047648	0.050846
model_english_google_news_7B_50dim-4	16	0.010	20	0.273763	0.096555	0.098077	7970.0	0.432235	0.408075	0.658417	0.682393	0.036047	0.061318
model_english_google_news_7B_50dim-5	16	0.100	10	0.293484	0.099543	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
model_english_google_news_7B_50dim-6	16	0.100	20	0.294172	0.101535	0.107118	9799.0	0.000000	0.000000	0.500000	0.500000	0.107118	0.000000
model_english_google_news_7B_50dim-7	32	0.001	10	0.256035	0.100958	0.095190	7938.0	0.394702	0.375106	0.634167	0.661738	0.051261	0.047629
model_english_google_news_7B_50dim-8	32	0.001	20	0.256646	0.100489	0.094980	7965.0	0.401226	0.381866	0.637515	0.664745	0.049883	0.048853
model_english_google_news_7B_50dim-9	32	0.010	10	0.276216	0.100067	0.095550	7834.0	0.417365	0.406632	0.647059	0.676170	0.044524	0.053643
model_english_google_news_7B_50dim-10	32	0.010	20	0.273565	0.099578	0.093491	7756.0	0.433879	0.421547	0.654796	0.682846	0.043395	0.054658
model_english_google_news_7B_50dim-11	32	0.100	10	0.291494	0.100665	0.106758	9347.0	0.124113	0.093292	0.531272	0.537959	0.089623	0.014549
model_english_google_news_7B_50dim-12	32	0.100	20	0.295328	0.101417	0.105929	9621.0	0.047295	0.034527	0.511466	0.514259	0.102209	0.004053
model_english_google_news_7B_50dim-13	64	0.001	10	0.271966	0.101122	0.095604	7975.0	0.409580	0.386424	0.642664	0.671436	0.046576	0.051804
model_english_google_news_7B_50dim-14	64	0.001	20	0.274506	0.100904	0.096757	7972.0	0.409813	0.388370	0.643794	0.672825	0.044577	0.053595
model_english_google_news_7B_50dim-15	64	0.010	10	0.288847	0.100621	0.094668	7861.0	0.416401	0.400442	0.645704	0.675563	0.046290	0.052060
model_english_google_news_7B_50dim-16	64	0.010	20	0.296013	0.100544	0.098948	8067.0	0.420897	0.391273	0.652283	0.677774	0.037842	0.059682
model_english_google_news_7B_50dim-17	64	0.100	10	0.320761	0.100848	0.098972	8560.0	0.262624	0.228125	0.576961	0.598745	0.075677	0.026369
model_english_google_news_7B_50dim-18	64	0.100	20	0.327471	0.101128	0.100749	8677.0	0.265996	0.220941	0.578555	0.601937	0.072336	0.029233
model_english_google_news_7B_50dim-19	128	0.001	10	0.308700	0.101070	0.098690	8045.0	0.397670	0.377055	0.638544	0.666659	0.044672	0.053510
model_english_google_news_7B_50dim-20	128	0.001	20	0.305446	0.101015	0.098173	7995.0	0.407163	0.389346	0.643494	0.672288	0.042948	0.055061
model_english_google_news_7B_50dim-21	128	0.010	10	0.341578	0.101031	0.099764	8197.0	0.391847	0.362370	0.636142	0.663572	0.044477	0.053686
model_english_google_news_7B_50dim-22	128	0.010	20	0.336297	0.101070	0.101691	8086.0	0.408850	0.395240	0.647491	0.671682	0.036734	0.060692
model_english_google_news_7B_50dim-23	128	0.100	10	0.364466	0.101333	0.101157	8394.0	0.317967	0.282349	0.600199	0.623944	0.060488	0.039501
model_english_google_news_7B_50dim-24	128	0.1	20	0.41063	0.107598	0.103126	8434.0	0.34807	0.3072	0.615322	0.640013	0.050325	0.04846

Table 3.21 Trained Neural Networks on pre-trained English Google news 50dim 7B corpus with OOV token

The tenth model trained achieved the lowest hamming loss value among the twenty-four models trained. The section is now complete. Each pre-trained neural network has a comparison candidate, demonstrating the lowest hamming loss and zero-one loss value upon the test data. The chapter to come compares the five model classifiers to select the one best classifier. The weights of the classifier selected are the main component of developing an accurate recommendation algorithm.

3.3.5 Select best model classifier (out of the five neural networks)

In the final section of the chapter takes place the selection of the best model classifier out of the five neural networks trained on the previous subsections. The model estimator selected will continue on the next chapter. The chapter that will thoroughly present the development and the intuition behind the interactive conversational agent. For the moment, the focus is on the selection plan that will help to decide the best model estimator among those trained

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

and saved. The selection plan represents a shorter version of the selection plan executed on chapter 3.2.3.

The selection plan followed in this chapter includes the following three comparisons:

Comparison 1 Create the scoring dataframes per model estimator and compare the values of Hamming loss, Zero-one loss, Test Hamming loss, Test Binary Crossentropy (loss), ROC-AUC, variance and bias-variance tradeoff.

Similar to the previously used scoring dataframe the model scoring metrics illustrated are:

- Test Loss: The binary crossentropy of the model estimator on the data never-seen before by the neural network. The lower the loss the closer the movie genres predictions to the actual movie genres of a movie. An indication of the model's potential to generalize.
- Test Hamming loss: The classifier's ability to correctly predict the movie genre(s) of new content. The lower the hamming loss, the lower the misclassifications of the model estimator.
- Hamming Loss: The fraction of wrong labels to the total number of labels.
- Zero-one Loss: The number of misclassifications. The best performance is 0, because this will indicate that all the movies have been allocated to the correct genre sequence.
- F1_score: The f1 score value as described in section 3.2.3
- ROC_score: The AUC score as referenced in section 3.2.3
- Variance: The variance value of the model predictions. A closer to zero variance indicates both minimum spread of model predictions on new testing data and secondly the model estimator is good at picking out hidden patterns between the trainable parameters.
- Bias: The bias of a model estimator stands for the assumption made by the neural network to approximate the target function. The more biased a neural network is, the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

more sensitive is to drive conclusions based on noisy data and patterns that do not help distinguish the difference between the movie genres. In general, the high bias reduces the flexibility of a model, but amplifies its ability to learn fast. However, it affects negatively the predictive performance on more complex problems, because such problems fail to understand the simplifying assumptions of the model bias [17] (the reader may refer to this link)

The model estimator that will present the best performance on those various metrics will be the one selected for chapter four.

Comparison 2 Compare the confusion matrices for each model estimator. The confusion matrix will point to the model classifier that allocated correctly the Predicted Genres to the True target values. Confusion matrix is an appropriate tool for a scientist to evaluate the classifier's performance. The proper use of a confusion matrix has been documented here.

Comparison 3 Evaluate the goodness of fit of the target function estimated by each classifier accordingly on training and validation data, from checking both the performance and the optimization learning curves.

Comparison 1: Scoring results per model classifier

Tag Name	Embedding Dimension tag	Batch tag	Learning Rate tag	Decay Multiplier tag	Test Loss	Test Hamming Loss	Hamming Loss	Zero_one Loss	F1_score	F1_score_samples	ROC_score	ROC_score_samples	Bias	Variance
multi_input_keras_model-7	100.0	32	0.001	10	0.019068	0.058487	0.003620	566.0	0.983106	0.983534	0.990600	0.992834	-0.000024	0.095667
glove_embeddings-8	NaN	64	0.001	20	0.190848	0.087977	0.057641	6178.0	0.669535	0.660289	0.767565	0.802285	0.034453	0.062776
English_Google_News_130GB_20dim_without_OOV-3	NaN	32	0.010	10	0.248332	0.094409	0.092434	7761.0	0.443231	0.427533	0.659358	0.687790	0.042536	0.055432
model_english_google_news_130GB_20dim_with_oov-8	NaN	16	0.001	20	0.244693	0.099933	0.093995	7910.0	0.400077	0.379538	0.636093	0.663331	0.051852	0.047105
model_english_google_news_7B_50dim-10	NaN	32	0.010	20	0.273565	0.099578	0.093491	7756.0	0.433879	0.421547	0.654796	0.682846	0.043395	0.054658

Table 3.22 Scoring table of the selected model classifiers

From the table presented above, it is clear that the custom trained neural network by far surpass the model classifiers of pre-trained word embedding models. The difference is not only seen in the relative hamming loss and zero-one loss but also by the other relevant metrics, which describe the quality of the classification predictions and learning. More specifically, the first

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

model has by far the lowest testing loss and the highest f1-score and ROC-AUC score with at least 0.25 units of difference from the other model classifiers. Last but not least, the first model is the least biased classifier and the one with the lowest spread between the true and the predicted values.

Comparison 2: Confusion matrices

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	1342	0	0	1	4	1	1	4	0	1	0	0	1	0	0	0	2
Adventure	1	377	10	3	14	1	9	2	0	1	2	0	1	0	1	0	0
Animation	0	1	258	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Children	0	5	1	212	6	0	0	0	2	2	0	0	0	0	0	0	0
Comedy	3	1	0	0	2365	0	0	0	0	0	0	0	0	0	0	0	0
Crime	5	0	0	0	0	517	0	28	1	1	0	3	0	0	6	0	0
Documentary	3	0	1	1	3	0	768	2	0	1	0	0	0	0	0	0	0
Drama	0	3	0	1	7	8	11	2676	0	0	0	0	0	0	0	0	0
Fantasy	0	1	0	0	0	0	0	0	43	2	0	1	0	0	0	0	0
Horror	0	0	0	2	0	0	2	1	1	585	0	0	0	0	0	0	0
Musical	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0
Mystery	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0
Romance	0	0	0	0	2	0	0	1	0	0	1	0	55	0	0	0	0
Sci-Fi	0	2	0	0	0	0	1	0	1	0	0	0	0	60	0	0	0
Thriller	2	0	0	0	0	1	1	0	0	0	0	2	1	0	147	0	0
War	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	12	0
Western	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	90

Table 3.23 Confusion matrix - model 1 (Custom Keras model)

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	643	28	3	0	198	42	1	335	0	81	0	0	0	7	19	0	0
Adventure	151	51	12	3	90	0	3	97	0	13	0	0	0	1	1	0	0
Animation	84	17	67	3	65	0	5	14	0	6	0	0	0	0	0	0	0
Children	42	4	8	8	105	0	0	57	0	3	0	0	0	1	0	0	0
Comedy	102	1	5	1	2061	0	1	176	0	20	0	0	0	1	1	0	0
Crime	88	0	1	0	6	87	2	334	0	28	0	0	0	0	15	0	0
Documentary	254	0	0	0	7	0	456	54	0	7	0	0	0	1	0	0	0
Drama	60	1	0	0	27	16	2	2551	0	43	0	0	0	1	5	0	0
Fantasy	17	2	1	0	1	0	0	3	0	23	0	0	0	0	0	0	0
Horror	21	0	0	0	1	2	0	70	0	496	0	0	0	0	1	0	0
Musical	12	0	0	0	4	0	1	7	0	0	0	0	0	0	0	0	0
Mystery	36	0	0	0	1	5	0	9	0	14	0	0	0	0	5	0	0
Romance	16	0	0	0	19	0	0	21	0	0	0	0	0	0	3	0	0
Sci-Fi	35	0	2	0	1	0	1	0	0	14	0	0	0	8	3	0	0
Thriller	52	0	0	0	0	12	0	30	0	43	0	0	0	0	17	0	0
War	9	0	0	0	0	0	1	5	0	0	0	0	0	0	0	0	0
Western	84	0	0	0	3	0	2	3	0	0	0	0	0	0	0	0	0

Table 3.24 Confusion matrix - model 2 (GloVe embeddings)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	582	16	0	0	106	99	9	443	0	56	0	0	0	39	7	0	0
Adventure	163	28	2	1	47	2	4	149	0	18	0	0	0	8	0	0	0
Animation	103	10	4	1	71	0	4	48	0	10	0	0	0	8	2	0	0
Children	55	6	2	1	73	1	1	80	0	6	0	0	0	3	0	0	0
Comedy	370	2	1	1	981	28	35	886	0	49	0	0	0	13	3	0	0
Crime	103	0	0	0	48	118	1	255	0	19	0	0	0	6	11	0	0
Documentary	133	0	0	0	36	1	374	223	0	5	0	0	0	7	0	0	0
Drama	327	1	0	0	344	29	37	1892	0	59	0	0	0	11	6	0	0
Fantasy	14	1	0	1	4	0	3	16	0	5	0	0	0	3	0	0	0
Horror	158	0	0	0	28	2	4	169	0	207	0	0	0	15	8	0	0
Musical	3	0	0	0	11	0	2	8	0	0	0	0	0	0	0	0	0
Mystery	16	0	0	0	11	7	0	26	0	4	0	0	0	1	5	0	0
Romance	6	0	0	0	10	1	1	40	0	1	0	0	0	0	0	0	0
Sci-Fi	30	0	0	0	2	0	1	10	0	6	0	0	0	13	2	0	0
Thriller	27	0	0	0	19	15	1	75	0	13	0	0	0	1	3	0	0
War	2	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0
Western	44	0	0	0	7	2	0	39	0	0	0	0	0	0	0	0	0

Table 3.25 Confusion matrix - model 3 (English Google News 20dim 130GB no OOV)

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	740	31	4	0	111	66	14	319	0	50	0	0	0	13	9	0	0
Adventure	224	15	1	2	61	1	8	98	0	9	0	0	0	3	0	0	0
Animation	118	5	9	1	71	2	7	39	0	9	0	0	0	0	0	0	0
Children	75	5	3	0	79	1	1	61	0	3	0	0	0	0	0	0	0
Comedy	572	1	0	2	901	34	50	748	0	52	0	0	0	6	3	0	0
Crime	139	0	0	0	41	87	7	244	0	34	0	0	0	0	9	0	0
Documentary	205	0	0	0	35	0	362	168	0	8	0	0	0	1	0	0	0
Drama	574	1	1	0	316	19	58	1664	0	65	0	0	0	2	6	0	0
Fantasy	22	1	0	0	5	1	2	9	0	7	0	0	0	0	0	0	0
Horror	209	0	0	0	23	3	5	130	0	207	0	0	0	7	7	0	0
Musical	4	0	0	0	10	0	1	9	0	0	0	0	0	0	0	0	0
Mystery	21	0	0	0	5	9	0	25	0	7	0	0	0	0	3	0	0
Romance	13	0	0	0	12	0	0	33	0	1	0	0	0	0	0	0	0
Sci-Fi	33	1	0	0	2	1	1	7	0	10	0	0	0	7	2	0	0
Thriller	43	0	0	0	14	12	1	64	0	16	0	0	0	0	4	0	0
War	4	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0
Western	56	1	0	0	6	1	0	28	0	0	0	0	0	0	0	0	0

Table 3.26 Confusion matrix - model 4 (English Google News 20dim 130GB with OOV)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
Action	613	13	2	1	133	125	30	358	0	63	0	0	0	6	12	0	1
Adventure	170	4	6	10	65	1	9	147	0	10	0	0	0	0	0	0	0
Animation	87	5	15	11	69	2	10	53	0	9	0	0	0	0	0	0	0
Children	57	2	2	10	72	0	5	76	0	4	0	0	0	0	0	0	0
Comedy	367	0	1	8	1019	35	61	818	0	52	0	0	0	3	5	0	0
Crime	143	0	0	0	68	114	3	190	0	36	0	0	0	0	7	0	0
Documentary	123	0	0	1	39	1	399	209	0	7	0	0	0	0	0	0	0
Drama	385	0	0	1	398	38	69	1734	0	69	0	0	0	1	10	0	1
Fantasy	14	0	0	1	6	0	3	15	0	8	0	0	0	0	0	0	0
Horror	170	0	0	0	50	3	4	108	0	245	0	0	0	6	5	0	0
Musical	6	0	0	0	10	0	1	7	0	0	0	0	0	0	0	0	0
Mystery	20	0	0	0	9	6	1	21	0	8	0	0	0	0	5	0	0
Romance	6	0	0	0	19	3	1	29	0	1	0	0	0	0	0	0	0
Sci-Fi	30	1	0	0	2	0	3	11	0	11	0	0	0	4	2	0	0
Thriller	42	0	0	1	22	13	2	56	0	13	0	0	0	0	5	0	0
War	4	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0
Western	34	0	0	0	14	3	0	33	0	0	0	0	0	0	0	0	8

Table 3.27 Confusion matrix - model 5 (English Google News 50dim 7B with OOV)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

The classification matrix of the first model stands out compared to the other four matrices. The second matrix, which belongs to the GloVe embeddings model presents more promising results than the third, the fourth, and the fifth matrix. However, the second model classified most of the movies into the comedy and the drama genre, the two most populated movie genres in the dataset. The high numbers of misclassified movies on wrong genres indicate under-fitting, a behavior observed in all the four model classifiers but the first one. Under-fitting classifiers cannot correctly spot the semantics and the tokens that separate the target values. As a result, they classify movies to only the genre they better learned. Overall, from the second comparison, the first model is also the winner of the duel.

Comparison 3: Performance & Optimization Learning curves

Performance learning curves (training and validation) monitoring the hamming loss metric

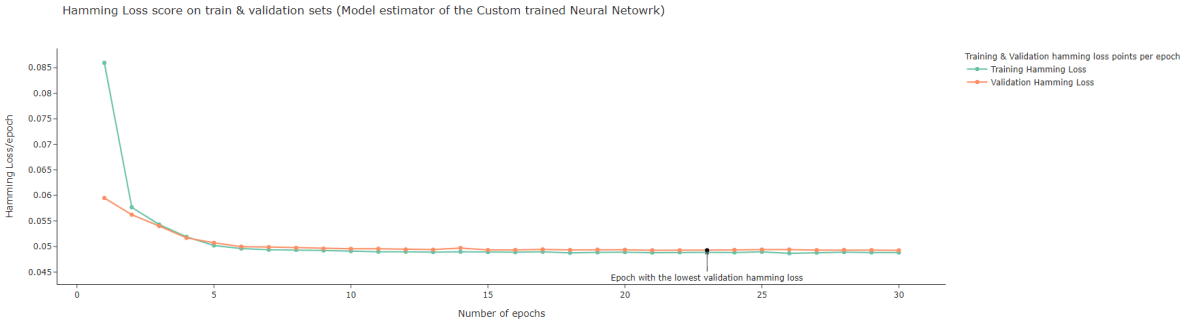


Figure 3.25 Custom Keras model - Hamming loss learning curves

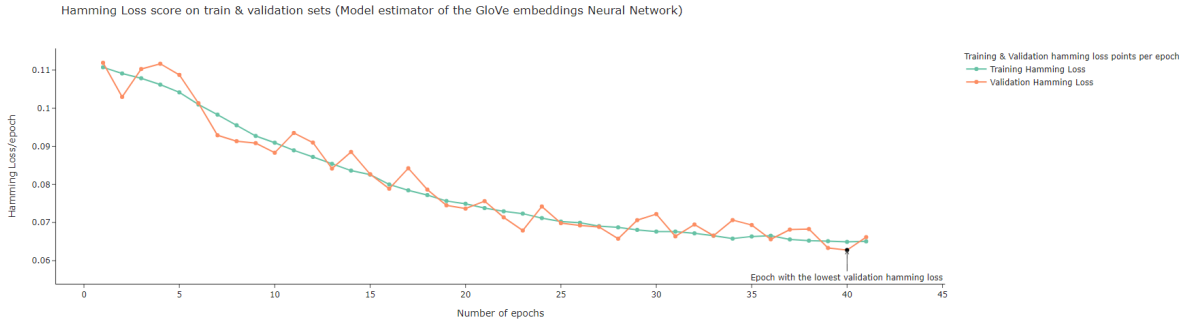


Figure 3.26 GloVe embeddings - Hamming loss learning curves

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

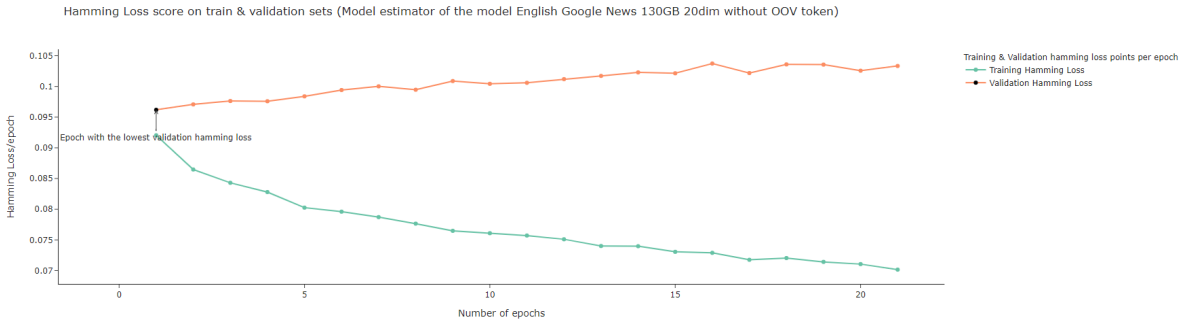


Figure 3.27 English Google News 20dim 130GB no OOV - Hamming loss learning curves

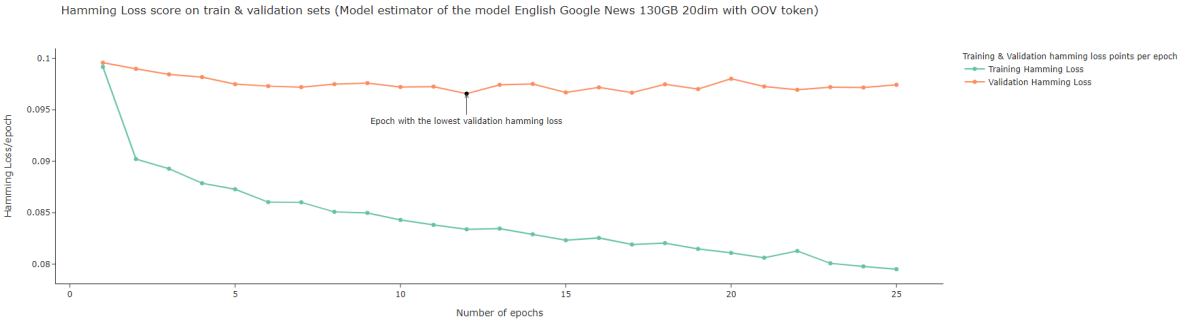


Figure 3.28 English Google News 20dim 130GB with OOV - Hamming loss learning curves

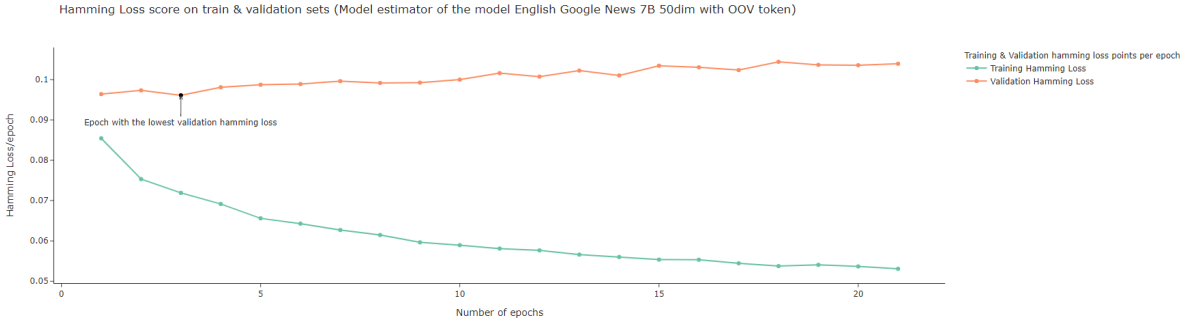


Figure 3.29 English Google News 50dim 7B with OOV - Hamming loss learning curves

The bottom three plots validate the under-fitting behavior of the three model estimators trained on the English Google News models. The training dataset learning curve keeps decreasing without reaching a point of stability even after twenty training epochs with the Adam optimizer.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

On the contrary, the first two performance learning curves exhibit more stable and smooth learning. The first learning curve, of the custom trained neural network, reaches a sufficiently low point of stability very early, at around the fifth training epoch. The second pair of learning curves, of the GloVe embeddings model, present a stable training curve and a spiky (non-smooth) validation curve. That phenomenon indicates that the validation dataset was probably not sufficient for the training—the so-called unrepresented validation dataset. A solution to this would be an increase in the validation split ratio, selecting more data for performance validation—for example, 60% training and 40% validation data instead of 70-30%. Last but not least, the significant gap between the validation and training learning curves observed in the last three models represents the under-fit generalization error.

Optimization learning curves (training and validation) monitoring the loss function

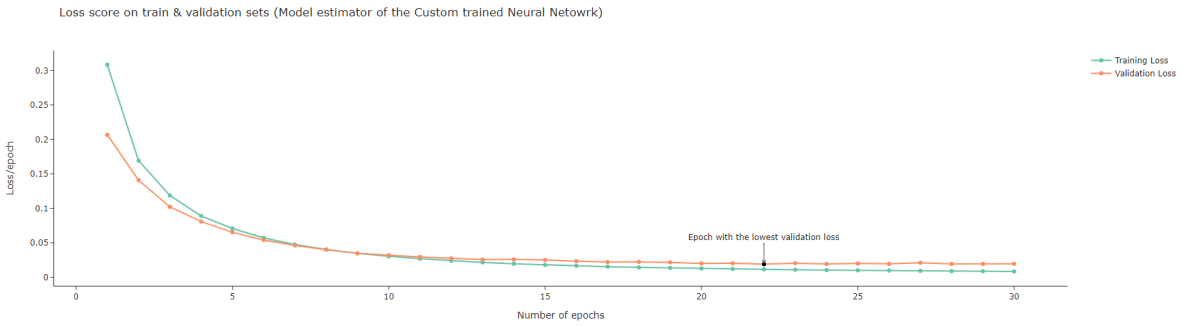


Figure 3.30 Custom Keras model - Loss learning curves

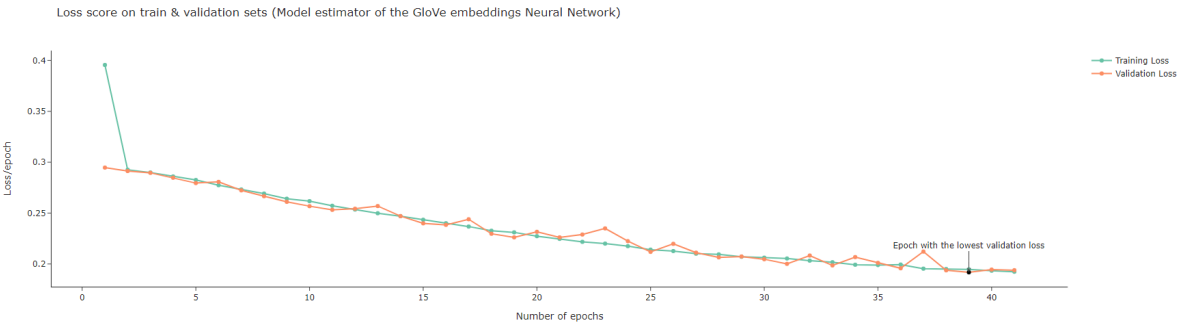


Figure 3.31 GloVe embeddings - Loss learning curves

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

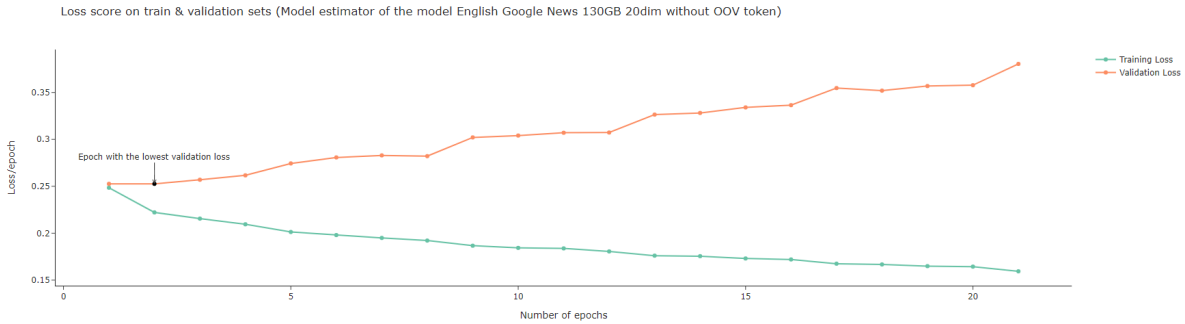


Figure 3.32 English Google News 20dim 130GB no OOV - Loss learning curves

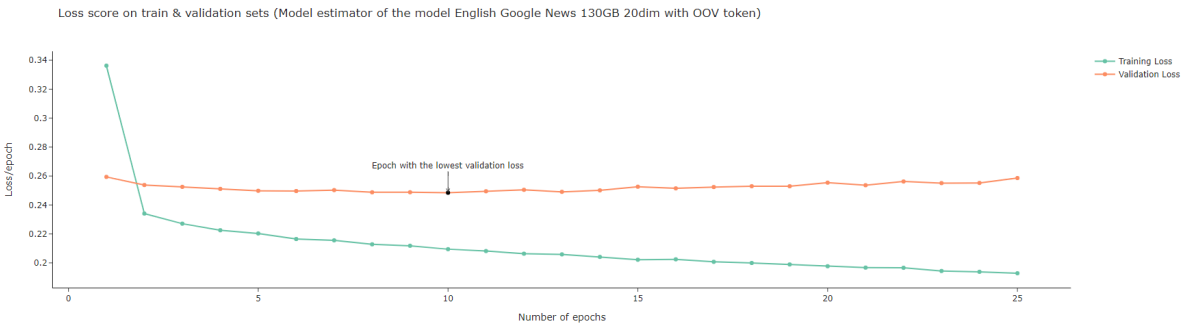


Figure 3.33 English Google News 20dim 130GB with OOV - Loss learning curves

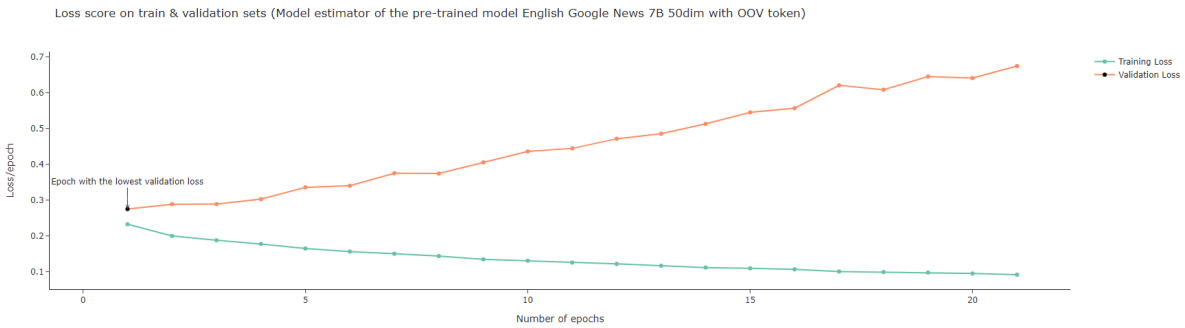


Figure 3.34 English Google News 50dim 7B with OOV - Loss learning curves

The optimization learning curves are more appropriate to identify over-fitting and under-fitting classifiers. The last three models present the same pattern with the relative performance learning curves. While the first and the second models present well-fitted optimization learning curves. The first model classifier also achieved the smoothest learning curve with a deficient generalization error. Based on all the three comparisons of the selection plan, the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

custom-trained neural network, with custom-trained word embedding vectors, demonstrated the best performance. For this reason, it is the one selected for the next chapter for building the recommendation algorithm.

Concluding, in the third chapter of the thesis, the documentation addressed many different issues. Initially, the document presented the structure of the neural network, developed to tackle the classification problem under review. Four are the main components to assemble a functional neural network, (1) define the input layer(s), (2) define the hidden layer(s), (3) define the output layer and (4) compile the model by specifying the optimization function, the optimization metric, and the performance metric. The documentation thoroughly explained the intuition behind the configuration of its of the four mentioned components. Moreover, the document presented the configurations made during the classifier's training phase and the useful outputs produced during the model training that can assist the decision-making of a good fitting or a bad fitting classifier, referenced in page 78. After the completion of the training phase, the document assembled a five-step selection plan, presented on page 82 to compare and to identify the classifier that outperformed all the others. Up to this point, the authors trained only one classifier; a custom-made neural network built upon the movie dataset. In the last section, four additional model classifiers trained using saved neural networks with pre-trained word embeddings. The five in total classifiers compared to each other using a shorter version of the initial selection plan. The comparison phase selected the custom neural network as the best model classifier out of all model estimators trained. The model selected has the following hyper-parameters and optimization function:

- Embeddings dimension: 100
- Batch size: 32
- Learning rate: 0.001
- Decay steps multiplier: 10

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- Optimizer: Adam

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter Four

Engineering an Interactive Conversational Agent

4.1 Introduction

The last chapter of the thesis documentation gives a thorough description of how a researcher can exploit the *output(s)* of a text-classification neural network. The document will refer both to the output(s) of a neural network, and to one of the many options of utilizing those output(s). Three significant sections compose the final chapter. The first section presents which is the output of a neural network and how to extract it. The second section illustrates the intuition behind creating a functional conversational agent that combines the information given by a third-party user, the movie content, and the weights of the best neural network to recommend movies similar to what the user has requested. The final section demonstrates a survey conducted to compare the conversational agent created by the research team of this thesis against similar conversational agent applications. The survey will evaluate different aspects of the application and the quality of recommendations proposed.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

4.2 Neural Network Weights | Word Vector Embeddings

The most valuable asset of a neural network that makes it easily portable and reproducible is its weights. Even though the weights initialization is random at first, at the end of each training epoch and based on the optimization metric and the optimizer selected, those weights are updating towards to achieve a generalized prediction algorithm. The better the model estimator's predictions between the true target values and the validation target values, the more probable to find the weights of a mathematical algorithm that can generalize and predict new movie content.

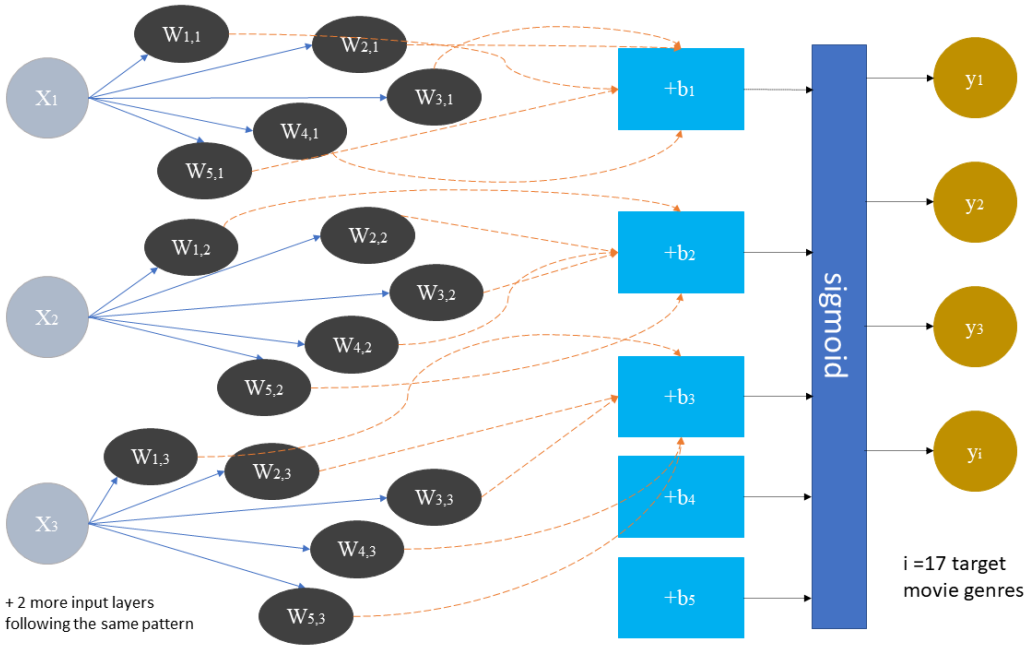


Figure 4.1 Overview of how the weights are updated during a training epoch

The goal is to estimate the weights that would accurately predict the actual values of the target label and best fit a line across the actual target values. Also, note that the model is a multi-output neural network, with each output to represent a movie genre. At each training iteration (epoch) the weights generated from the embedding layers, and the weights estimated by each hidden layer are updated continuously to minimize as much as possible the loss function and

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

improve the model's performance.

In a text classification neural network, such as the one used in this scientific experiment, the essential weights belong to each embedding layer. The embedding layer produces a vector of numbers that map similar words together in an n-dimensional space. The number of dimensions serves as a hyper-parameter of the neural network, so different dimensions produce different classification results. The more the dimensions the more the information used by the neural network to map correctly the words. However, it is not suggested to use a very high number of dimensions, because this would probably drive the model estimator towards overfit and to learn very well only the semantics of the trained content.

The output of this section is essential for the recommendation algorithm, the authors will built in the next section. To properly extract the output three steps should be followed.

1. Extract the embedding layer weights belonging to each trainable feature. The trainable features with movie content are five. The actors, the plot summary, the movie content, the movie reviews and the movie title. All of those five features are assigned to one embedding layer of which the weights are very important and useful for the conversational agent.

2. Having extracted the weights of each embedding layer, first **map** each word/token of the five trainable columns to their corresponding vector array and then **replace** the word from each textual feature to the corresponding embedding vector. So, initially create a map dictionary that will assign each word of a column's vocabulary to a 100-dimensional array, and then use that dictionary to replace for example the word *adventure* or the actor name *Tobey Maguire* with that 100-dimensional array or 16-dimensional array respectively. Recall, that for the actors column, the authors used 16-dimensional embedding vectors, while for the rest of the four columns 100-dimensional vectors. As a result, instead of a sequence of words/tokens, the output will be a sequence of 100-dimensional and 16-dimensional arrays.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

3. The third and the final step is to concatenate the number of different 100 dimensional vectors into one vector that will contain the *information* from the whole sentence of words. Continuing the example of actor names. Replace the 15 actor names of 16-dimensional vectors each, by one vector with sixteen float numbers. Those sixteen float numbers, they may represent an average of the fifteen vector arrays, a minimum array, a maximum array or even their concatenation. Basically, *summarize* (shrink) the information from fifteen actor names to one vector pointing to the specific movie and its sequence of genres.

Before moving on, it's important to demonstrate the above steps with a simple example, using dummy numbers.

Weights of the embedding layer

Layer Weights

$$= \begin{bmatrix} -0.42056742 & -0.3540595 & -0.25417486 & -0.50596726 & -0.29918054 \\ -0.23971583 & -0.39325562 & -0.35581827 & -0.3175518 & -0.2992685 \\ -0.26149312 & -0.3268542 & -0.34264958 & -0.50005287 & -0.41450888 \\ \dots (m = 60) & & & & \end{bmatrix} \tag{4.1}$$

$= \text{shape}(5 \times 60)$= 300-dimensional vector

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Assign (Map) each array of weights to the relative actor name

Tobey Maguire

$$= \begin{bmatrix} -0.42056742 & -0.3540595 & -0.25417486 & -0.50596726 & -0.29918054 \\ -0.23971583 & -0.39325562 & -0.35581827 & -0.3175518 & -0.2992685 \\ -0.26149312 & -0.3268542 & -0.34264958 & -0.50005287 & -0.41450888 \\ \dots (m = 60) & & & & \end{bmatrix} \tag{4.2}$$

$= \text{shape}(5 \times 60)$

Daniel Craig

$$= \begin{bmatrix} -0.30834767 & -0.26681098 & -0.2173222 & -0.11151562 & -0.27951762 \\ -0.1721798 & -0.25406063 & -0.38693774 & -0.19798501 & -0.257399 \\ -0.05970115 & -0.2399106 & -0.21202469 & -0.28024384 & -0.2577843] \\ \dots (m = 60) & & & & \end{bmatrix} \tag{4.3}$$

$= \text{shape}(5 \times 60)$

Replace the sequence of tokens per trainable column by the relative array of embeddings, using as map the output of the previous step.

For example the column actors, which has the following structure:

['tom hanks', 'tim allen', 'don rickles', 'jim varney', 'wallace shawn', 'john ratzenberger', 'annie potts', 'john morris', 'erik von detten', 'laurie metcalf', 'r. lee ermey', 'sarah freeman', 'penn jillette', 'jack angel', 'spencer aste']

$$\downarrow$$

$$\begin{bmatrix} [-0.3083 & -0.2668 & -0.2173 & -0.1115 & -0.2795 & \dots & x_i = 300] \\ [-0.1722 & -0.2541 & -0.3869 & -0.1980 & -0.2574 & \dots & x_i = 300\ ] \\ [-0.2399 & -0.2120 & -0.2802 & -0.2578 & -0.2574 & \dots & x_i = 300] \\ \vdots (m = 15) & & & & & & \end{bmatrix}$$

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Please note that numbers presented are fictitious.

The first array corresponds to Tom Hanks, the second array to Tim Allen, the third array to Don Rickles and so on.

Result of the example: Replace each actor name with their corresponding array of 300 float numbers. Use the array to map a token, in this example an actor name, in 300-dimensional space and accurately locate the actor names played in similar movie genres. Implement the mapping in the same way to any token from the other four content columns, the plot summary, the movie features, the movie reviews, and the movie title. Also, note that any token, either this is a word or an actor name, not mapped in the relevant dictionary, gets the value of the Out-of-Vocabulary token. Recall that the total number of actors in 48991 is equal to approximately 260,000 names. However, only 240,000 of those tokenized. The rest assigned to the OOV token. The use of an OOV token is significant because it allows mapping not so frequent words to an N-dimensional vector instead of dropping them.

Before starting the creation of the conversational agent, the final step is to aggregate the information included in each of the five movie content columns. More specifically, the result of the previous step was to replace tokens by numerical arrays in each of the five columns. So, the first row of the actors' column now has fifteen numerical arrays instead of fifteen actor names, and the plot summary of the same movie now has twelve numerical arrays instead of twelve words. The final goal is to aggregate the information per row and per column to get numerical arrays of shape (100,1) for the plot summary, movie features, movie reviews, movie title or (16,1) for the actor names. To aggregate the information the metrics applied are: the minimum, the maximum, and the average. The example below demonstrates how those three aggregations will be applied.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

$$\begin{bmatrix} [1, & 2, & 3] \\ [4, & 5, & 6] \\ [7, & 8, & 9] \\ [10, & 11, & 12] \end{bmatrix} \rightarrow [5.5, 6.5, 7.5], \quad [1, 2, 3], \quad [10, 11, 12]$$

$$\text{shape}(4 \times 3) \qquad \text{shape}(3,) \quad \text{shape}(3,) \quad \text{shape}(3,)$$

Replace tokens by arrays | Average | Minimum | Maximum

Aggregating each sequence of arrays by the average, the minimum, and the maximum measures, creates additional information that the conversational agent can exploit to find closely similar movies. A researcher can use either one of those aggregation measures, either create each own or use more than three aggregation measures to generate more information about one movie. Next, concatenate those three aggregation arrays into one array of shape (9,). Below is the final array of the aggregated information.

[5.5, 6.5, 7.5, 1., 2., 3., 10., 11., 12.]

Note that the numbers presented in the earlier examples are dummy and do not match the real numbers generated from the experiments. They have been used only to demonstrate the intuition of extracting and exploiting the output of a neural network.

Applying the transformations presented in the examples above, the authors end up with a unique 1248-dimensional array that describes a movie. The array has the following structure:

- Actors column: 16-dimensional vector * 3 aggregation measures = 48 float numbers
- Plot summary column: 100-dimensional vector * 3 aggregation measures = 300 float numbers
- Movie features column: 100-dimensional vector * 3 aggregation measures = 300 float numbers
- Movie reviews column: 100-dimensional vector * 3 aggregation measures = 300 float numbers
- Movie title column: 100-dimensional vector * 3 aggregation measures = 300 float numbers overall total = 1248 float numbers to represent a single movie with the above movie content

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

The next part presents the conversational agent's architecture, the host environment, and the proper usage and functionality of the recommendation engine.

4.3 Assembling an Interactive Conversational Agent

The word embeddings generated from the previous chapter will be the main component used by the chatbot to propose similar movies to the one preferred by the user. The chapter will thoroughly describe the chatbot's intuition, the chatbot's architecture, the chatbot's technique to find similarities among movies, and which platform hosts the chatbot.

4.3.1 What is a chatbot

Chatbots are a kind of software application that tries to simulate online conversations via text or speech. Chatbots are programmed to behave correctly as a human would during a professional or day-to-day conversation with other human beings. Recall throughout the whole thesis documentation the reference of the word interactive conversational agent. The chatbot responses are changing based on the input it receives from the other side of the conversation. A chatbot is programmed to receive an input of information, understand, digest, and decipher the message and give back the appropriate response. A chatbot is often programmed to trigger a specific response when a keyword links to a specific course of action. Typical cases of chatbots that understand and respond to text information are those used as support agents on many websites. Apart from chatbots that understand a text, chatbots that can understand speech and response back in audio also exist. An example of those conversational agents is Siri used by Apple in many of its iPhone devices or Alexa developed by Amazon.

Conversational agents are part of the Artificial Intelligence domain, and they received an outburst of attention with the arrival of Natural Language Processing algorithms becoming a top trend nowadays. Chatbots can receive information from various APIs, combining both the input of a human and the input of another algorithm to identify an action and respond to the appropriate result. In this experiment, the chatbot crafted can understand and respond to textual messages and signals.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

4.3.2 Architecture of the chatbot

The chatbot in the current application serves as a conversational agent that receives from a human being three inputs of movie related content, uses those inputs to calculate the most similar movies, and responds four movie recommendations. The chatbot does no inner calculation or computation but rather receives the results of the recommendations from a third API, generated from a recommendation algorithm program running in Python.

Before describing the chatbot's actions and decisions, the document will first present the movie recommendation algorithm.

Step 1: The algorithm receives at first two textual inputs. So it demands two inputs in the form of plain written text. The first input is related to the genre of the movie; the user would like to watch. For example, the user would like to watch an *Action* movie. The second input in written format is a movie title related explicitly to the movie genre given as first input. For example an *action* movie is the movie *Skyfall*.

Step 2: Filter the rows of the initial dataset of 48,900 movies, and isolate only the movies that belong to the selected genre saved from step 1. Thus, import into the algorithm only the action movies (e.g., the 12,000 action movies). Now the algorithm will work with those 12,000 movies. This action can enormously minimize the processing time to come. Moreover, the movie recommendations should only include Action movies, so this step is a safe-lock that the recommendations will include action movies and not drama movies, for example.

Step 3: The algorithm receives the third and the final written input. This input describes the reasons why the user selected the action movie Skyfall. In this step, the user can write textual information about the movie's plot, the cast, a quick review of the movie, or anything that made him/her like the movie. The more the information the user writes, the more accurate the four movie recommendations. For example, the user can write *James Bond is the best secret agent, and Daniel Craig played very well the 007 MI6 agent.*

Step 4: If the movie title written by the user, belongs to a movie from the 12,000 action movies list,

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

then the movie features column is located for that movie. Recall that the movie features column contains text information about the plot, the cast, the director(s), the title, and the movie genre(s). Then the algorithm splits the sentence into separate words, and the algorithm concatenates those words with the words given in the third input. The output is a bag of words assembled by the movie features and the user's input. This bag of words list must contain as many words as possible. In the previous step, the documentation mentioned that more the written text from the user, the better. Continue to step 6.

Step 5: Else if the movie is not in the movies' list of the selected genre, either because the movie does not belong to the movie genre selected, or the dataset does not have any information about the selected movie title whatsoever. Then the bag of words is composed only of the words written by the user as the third input since no additional information about movie content exists. Continue to step 8.

Step 6: (only if the movie exists in the dataset): Collect the word embeddings of the movie selected by the user. Recall that the first section documented the creation of the word embedding vectors per movie. In this step, the algorithm imports all the word embedding vectors of all the movies of the selected genre (e.g., of all the action movies) and next isolates the word embedding vector of the index that links to the selected movie title (e.g. of the movie *Skyfall*). Specifically, the 1248-dimensional vectors for the movie *Skyfall* and all the other action movies are imported.

Step 7: (only if the movie exists in the dataset): Calculate the *cosine distance* between the word embeddings of the selected movie with all the other movies of the same genre. Based on the current example, calculate the cosine distance between the embeddings (1248-dimensional vector) of the movie *Skyfall* with all the other action movies. Below, the documentation presents a thorough explanation of the cosine similarity and an example of this technique. The output of the cosine similarity is a table with two columns. The first column is about the movie title, and the second column is about the similarity of the movies. The cosine similarity takes values between (0,1), with values closer to 1 indicating strong similarity, while values closer to 0 to indicate distant similarity or dissimilarity. From the 12,000 cosine distances select the highest

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

16 most similar movies to the one selected by the user. For the example, select the 16 most similar movies to the movie *Skyfall*.

Step 8: Isolate the columns of the movie features and the movie reviews of the most similar movie titles. Create a second bag of words list with the unique words from those two columns. This bag of words list is different from the bag of words created in step 4. Based on the second bag of words list, the algorithm creates a new column under the name *unique words*. The algorithm calculates how many words from the bag of words of Step 4 are found in the column of the unique words for each of the 16 most similar movies. So, per similar movie the algorithm returns a number which indicates how many words are found in the intersection of the two bag of words lists. The higher the number of common words, the closer the movie to the movie selected from the user because this indicates that they share common content. For example, movies related to James Bond, MI6, secret agents, and 007 should have many similar words to the movie title *Skyfall*. In the opposite scenario, in which the movie title is not part of the dataset, only the third input written from the user is utilized by the algorithm to calculate the number of unique words.

Step 9: Create the **movie score**. Movie-score is a metric that measures the overall similarity of the 16 movies with the movie selected by the user. It is a metric that combines information from four columns related to each movie. The algorithm orders the 16 movies based on that score, and recommends the top 4 movies to the user. The movie score combines information from the following columns:

- **imdb_rating_range**: This is a value assigned to movies based on their IMDB rating value given by the viewers of the movie. The IMDB rating takes values between the range [0, 10), and it is a valuable metric to measure the positive or negative opinion of the viewers. The algorithm applies the following transformation on the IMDB rating:
  - For IMDB values between [8, 10) -> assign the value 0.2
  - For IMDB values between [6, 8) -> assign the value 0.4
  - For IMDB values between [4, 6) -> assign the value 0.6

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- For IMDB values between [2, 4) -> assign the value 0.8
- For IMDB values between [0, 2) -> assign the value 1.0

The algorithm gives higher movie scores on movies with less IMDB rating because, in the opposite scenario, the movies with a very high IMDB rating would always get recommended. The algorithm should not spoil the recommendation results based only on a good review of the movie.

- **number_of_words**: The number of words value calculated on *Step 8*.
- **sentiment_value**: The sentiment value takes the value 1 if the movie has a rating greater than 2.5 else takes the value 0. The rating, as already documented in the second chapter, is a different value than the movie's IMDB rating. It is also a measure of the positive and the negative influence of viewers' opinions over the movie; however, it takes values on a 5-star scale. Likewise, the higher the rating value, the better.
- **rating**: The actual 5-star scale rating value monitored by the Grouplens movie dataset.

The movie score is calculated with the following formula:
movie_score = (imdb_rating_range*1.0) + (number_of_words*0.5) + (sentiment_value*0.5) + (rating *1.0)

Step 10: Propose the four movies with the highest movie score. The recommendation includes the *movie title*, the *IMDB rating* and the *URL* of the movie.

Above is the overall architecture of the recommendation algorithm. The architecture is programmed in Python, and it runs on a local-host server. The interactive conversation agent receives the result of the recommendation algorithm through the *ngrok* [29] application. Ngrok gives access to a public HTTPS URL for a program running locally on a development machine. It allows the exposure of a web server running on a local machine to the internet, just by specifying the web server's listening port. Ngrok enables access to any program output through that HTTP(s) URL from anywhere. Those URLs are also easily accessible by third-party applications like chatbots.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Before moving on with the rest of the conversational agent functionality, it is essential to reference the **cosine similarity** used by the recommendation algorithm to find similar movies to the one selected by the user. Cosine similarity is a measure that estimates how similar a pair of written documents are despite their size. The invention of the cosine similarity replaced the old-fashioned method of counting the common words between two written texts. When implementing the latter approach while the text's size was gradually increasing, it was almost probable that the common words between the text would increase even though the text would refer to totally different topics. Mathematically, the cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space [30]. Each word (token) in the written text represents one dimension, and the cosine similarity metric represents the angle of the written text based on its tokens. The smaller the angle between two documents, the higher the similarity. In this experiment, the multi-dimensional vectors are word embeddings belonging to each separate movie. Recall that in the previous section of this chapter, the authors created one multi-dimensional vector per movie (1248 float numbers), so the recommendation algorithm uses it. With the cosine similarity and the word embeddings, the algorithm programmed can successfully identify the similar movies.

The cosine distance formula:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}} \tag{4.4}$$

, where x, y are two multi-dimensional vectors of word embeddings and $\|\mathbf{x}\|$ the Euclidean norm of a vector.

An example of calculating the cosine similarity between two vectors.

$x : [1, 1, 0]$

$y : [1, 0, 1]$

Euclidean norm of x($\|\mathbf{x}\|$) = $\sqrt{1^2 + 1^2 + 0^2} = \sqrt{2}$

Euclidean norm of y($\|\mathbf{y}\|$) = $\sqrt{1^2 + 0^2 + 1^2} = \sqrt{2}$

$x \cdot y = x_1 y_1 + x_2 y_2 + x_3 y_3 = 1 * 1 + 1 * 0 + 0 * 1 = 1$

$\cos(x, y) = \frac{1}{\sqrt{2} \times \sqrt{2}} = \frac{1}{2} = 0.5$ this represents the similarity between the two vectors

The angle of the similarity is calculated as: $\theta = \cos^{-1} * 0.5 = 60°$

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Everything about the architecture of the recommendation algorithm documented and described. The next component to document is the hosting environment of the interactive conversational agent, the Dialogflow.

4.3.3 Dialogflow-Hosting the chatbot

Based on the Dialogflow's documentation page, the latter is defined as

A natural language understanding platform that makes it easy to design and integrate a conversational user interface into a mobile app, web application, device, bot, or interactive voice response system. Using Dialogflow, the product manager can provide new and engaging ways to interact with the product. Dialogflow can analyze multiple input types from customers, including text or audio inputs (like from a phone or voice recording). It can also respond to customers in a couple of ways, either through text or with synthetic speech.

For a conversation round to start between the chatbot and the user, an *intent* is matched to a keyword written or said by the user. Based on the relative documentation, the intent is the triggering point of the user's intention to get a response. For example, when the user will give as input the movie genre, the movie title, and the reason (s)he liked the movie title written, an intent is triggered to return the recommendation algorithm results. The components of an *intent* are the following:

- **Context:** Used to store parameters shared between intents. Additionally, context can indicate the arrival of upcoming or the completion of previous intent. Intent(s) are executed sequentially or in any order specified in the context.
- **Event:** A free way of triggering an intent without the obligation to match a specific keyword.
- **Training phases:** The written keywords that, when given by the user, trigger a specific intent. It is important to give as many training phases as possible to keep up with shorter versions of the same word (e.g., thanks, thank you, thnx -> same meaning but different written format).
- **Action and parameters**: When an intent is triggered, all the specified actions of that intent will

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

initiate. A so-called parameter value can store the result of an action, and that value is used by third-party programs or by the chatbot to generate pre-defined responses.

- **Responses:** Defined text, speech, or visual responses returned to the end-user as an output of an action. These may provide the end-user with answers, ask the end-user for more information, or terminate the conversation [25].
- **Fulfillment:** Enable communication between the chatbot and third party HTTP protocol to collect data and information.

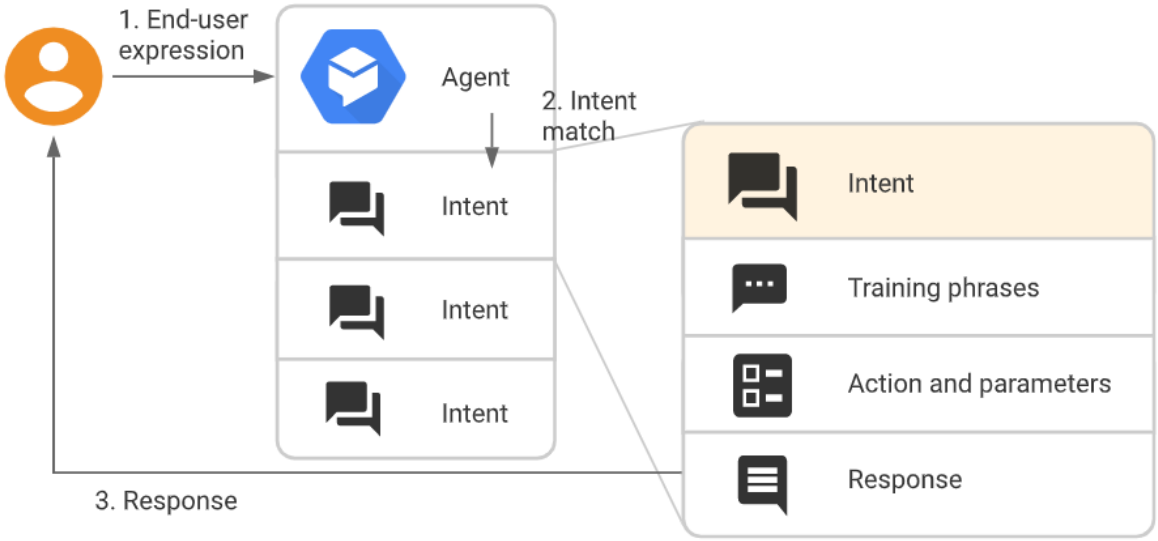


Figure 4.2 Basic flow for intent matching and responding to the end-user
Source: Dialogflow-Intent Overview

Apart from the component of *intents* also the *fulfilment* component is utilized. This component is configured separately from the *intents* component. The *Fulfilment* attribute connects Dialogflow with a third-party application, like the recommendation algorithm, through an HTTP(s) URL issued by ngrok. The third-party application receives a POST request from Dialogflow with a response written by the user. For example, when the application asks the user to provide a movie genre, this written information is passed to a parameter and then to the recommendation algorithm that uses the movie genre to filter out from the dataset the movies of different genre.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

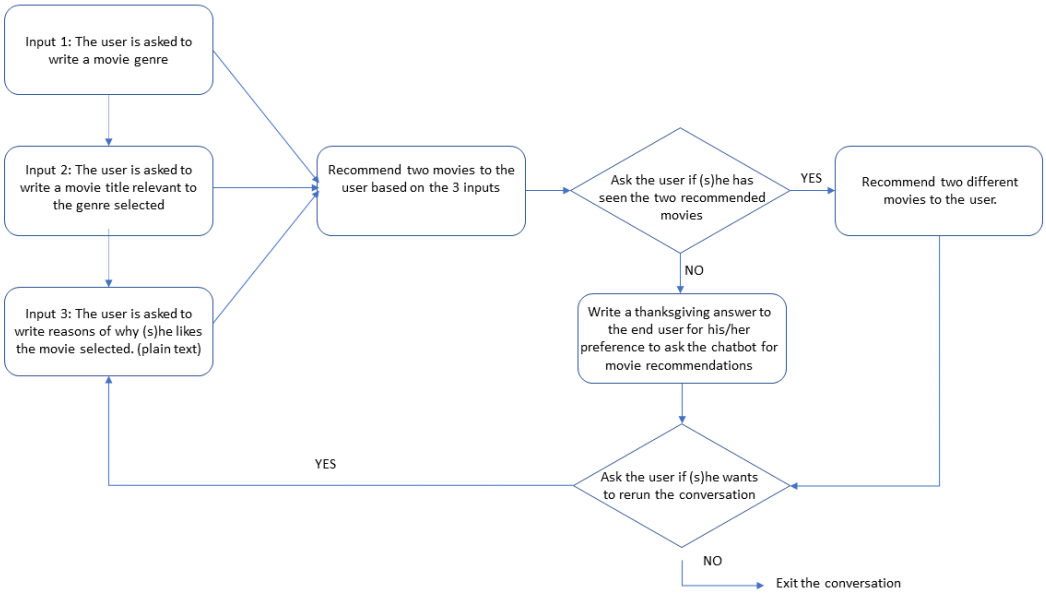


Figure 4.3 The implementation of the chatbot on Dialogflow

Each orthogonal shape represents a different intent and, thus, different actions. While the rhombus shape indicate if/else statements that trigger different actions.

Even though the Dialogflow is a service that hosts the chatbot, it does not support the appropriate user-friendly Graphical User Interface (GUI) for an end-user to interact with the chatbot. Dialogflow provides GUI interactions with various online services such as the Google Assistant and Facebook Messenger. The latter is utilized in this case, since speech interactions often use the former. The Facebook Messenger service of the conversational agent is fully functional, and users can access it under the name Kelly MovieBot.

With everything settled and documented, it is time to test and compare the recommendation algorithm and the chatbot with other similar chatbot services hosted in the Facebook Messenger environment. The authors conduct a small research and ask the public's opinion about the recommendation algorithm developed in the context of this thesis and derive insights about other similar conversational agents.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

4.4 Conversational Agent-Opinion based research

In the last section of the chapter, the authors run question-based research asking the public's opinion to evaluate the functionality and recommendations proposed among three conversational agents, chatbots, hosted on Facebook Messenger. The questionnaire successfully gathered 25 unbiased responses. The section organizes the research into three sub-sections. The first sub-section presents the relevant questions. The second section presents the results and critical insights. Finally, the last section refers to the conclusions made from the comparison.

4.4.1 Opinion-based questionnaire

The questions answered by the users, evaluate the chatbot application around three different topics:

1. The relevance of the recommendations proposed by each of the three chatbots.
2. The evaluation of the chatbot application from the perspective of the human-computer interaction and the graphical user interface.
3. The features of the chatbot, created by the authors, that satisfied the most users; and the features of the other two chatbots, the users preferred to experience in a feature version of *Kelly MovieBot*.

The reader can find the questionnaire and the chatbots under comparison in the following links:

- Questionnaire
- Chatbots: Kelly MovieBot, And Chill, Movie Bot

For the first topic, the authors selected three movies of three different genres. The first movie is the *action* movie **Skyfall**, the second movie is the *romantic* movie **Titanic**, and the third movie is the *thriller* movie **Psycho**. The questionnaire asks the users to rate the relevance of the recommendations proposed from each chatbot.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Question 1: Evaluate the recommendations of the action movie "Skyfall."

Question 2: Evaluate the recommendations of the romance movie "Titanic."

Question 3: Evaluate the recommendations of the thriller movie "Psycho."

Rating:

1. Irrelevant recommendation results
2. Partially irrelevant recommendation results
3. Neutral satisfaction with the results
4. Somewhat relevant movie recommendation results
5. Relevant movie recommendation results

For the second topic, the questionnaire asked the users to evaluate the interactive conversational agents based on the variety of recommendations propose, evaluate the Graphical User Interface (GUI) of the chatbot, and finally rate the level of human-computer interaction between the chatbot and the user. Specifically, the questions and their relative ranking scale provided as follows:

Question 4: How do you evaluate the number of recommendations proposed by the chatbot?

Rating:

1. Too many options than expected, I cannot choose
2. More than expected but I can decide
3. Few
4. Very Few
5. The ideal number of options to choose from

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Question 5: Do you find sufficient the Graphical User Interface (GUI) representation of the recommendations? Assess the recommendation(s) based on their representation (e.g., Images, URLs, videos, and interactive buttons)

Rating:

1. Strongly Disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly Agree

Question 6: During your conversation with the chatbot, do you get the feeling that you interact with a robot or a human agent?

Rating:

1. I get a strong feeling that I interact with a robot and not a human agent
2. I get a slight feeling that I interact with a robot and not a human agent
3. The conversation did not give me the feeling that I interact with a robot
4. I get a slight feeling that I interact with a human agent and not a robot
5. I get a strong feeling that I interact with a human agent and not a robot

The third and final topic provides two questions relevant to the features offered by the three chatbots. The first question is about the features the users are fond of about Kelly MovieBot, the authors' custom-created conversational agent. While the second question, draw insights about the features of the other two chatbots the users liked the most, and Kelly MovieBot should adopt.

Question 7: Which feature(s) did you like the most in Kelly MovieBot?

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Rating:

1. Re-run the whole conversation with only one response
2. The proposal of not so popular movies, but the promotion of both high and low IMDB rating movies
3. The proposal of 2 more movies if the user has already seen the first two movies
4. The friendly responses and clear instructions
5. Other

Question 8: Which feature(s) the other two chatbots had and would like to see on Kelly MovieBot?

Rating:

1. (And Chill + Movie Bot) Less Recommendations
2. (And Chill) Image per response
3. (And Chill) YouTube trailer link per recommendation
4. (Movie Bot) Propose movies based on favorite Actor or Oscar Winning Movies
5. (Movie Bot) Additional rating information apart from IMDB rating (e.g. Rotten Tomatoes)
6. Other

4.4.2 Questionnaire results

The document presents the questionnaire results in a organize format with the following structure.

- Line 1: Question
- Line 2: Kelly MovieBot (1st application)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- Line 3: And Chill (2nd application)
- Line 4: MovieBot (3rd application)
- Line 5: Comments

The first and the fifth lines describe the question asked to the application users and discuss the opinion results found. The remaining columns will color-highlight the most voted answer the three conversational agents received for the relevant question. The coloring applied is the following:

- Rank /Choice 1:
- Rank /Choice 2:
- Rank /Choice 3:
- Rank /Choice 4:
- Rank /Choice 5:

Question 1: Evaluate the recommendations of the action movie "Skyfall"

Rating

Kelly MovieBot

And Chill

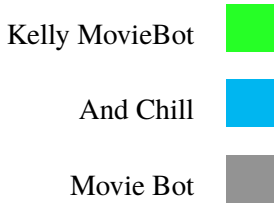
Movie Bot

Comments: The majority of the users (16/25) gave the five(5) number rate to the recommendations provided by Kelly MovieBot. On the contrary, most users rated the recommendations of the And Chill and the Movie Bot applications with the fourth rank, 13/25, 10/25, respectively. The outcome is positive for the recommendation algorithm running on Kelly MovieBot. Check the figure relative to Question 1 for reference to the question results.

Question 2: Evaluate the recommendations of the romance movie "Titanic"

Rating

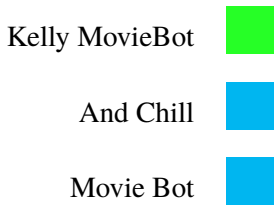
ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ



Comments: The majority of the users (17/25) gave the five(5) number rate to the recommendations provided by Kelly MovieBot. On the contrary, most users rated the recommendations of the And Chill and the Movie Bot applications with the fourth rank and the third rank, respectively. Specifically, eleven users awarded the fourth rank to And Chill, and another eleven users awarded the third rank to the recommendations of the Movie Bot agent. The outcome is very positive of the recommendation algorithm running on Kelly MovieBot. Check the figure relative to Question 2 for reference to the question results.

Question 3: Evaluate the recommendations of the thriller movie "Psycho"

Rating



Comments: Again the majority of the users awarded the highest of the ranks to the recommendations given by Kelly MovieBot. Individually, 13/25 (almost 52%) of the sample favor the movies proposed by Kelly MovieBot. On the contrary, And Chill received the fourth rank (10/25) in the majority of the votes. Movie Bot is coming last in the quality of recommendations, with respondents awarding the third and fourth ranks (9/25 in both ranks). Check the figure relative to Question 3 for reference to the question results.

Question 4: How do you evaluate the number of recommendations proposed by the chatbot?

Rating

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Kelly MovieBot

And Chill

Movie Bot

Comments: Kelly MovieBot received the fifth rank in the majority of the votes, which is equal to thirteen of the respondents. The upvote indicates that the four recommendations provided by Kelly Movie Bot are the sufficient number of options to choose a proposed movie. Moreover, AndChill and Movie Bot both awarded with the third rank, indicating the dissatisfaction of the users with the low number of recommendations proposed. Check the figure relative to Question 4 for reference to the question results.

Question 5: Do you find sufficient the Graphical User Interface (GUI) representation of the recommendations? Assess the recommendation(s) based on their representation (e.g., Images, URLs, interactive messages, etc.)

Rating

Kelly MovieBot

And Chill

Movie Bot

Comments: Kelly Movie Bot received ten votes on the fourth and the fifth rank equally. The result indicates that the use of images and the interactive buttons had a positive effect on the user experience. The results were also satisfactory for the And Chill and Movie Bot applications. However, even though Movie Bot received 10/25 responses in the fourth ranks, 12 out of 25 votes were distributed across the first, the second, and the third ranks. The downvote is a sign that the application uses too many or too few images and interactive buttons, making it difficult or confusing for users to navigate. Check the figure relative to Question 5 for reference to the question results.

Question 6: During your conversation with the chatbot, do you get the feeling that you interact with a robot or a human agent?

Rating

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Kelly MovieBot 

And Chill 

Movie Bot 

Comments: The question represents what most companies, developing similar applications, try to achieve. They want their users to get the feeling that they interact with a human agent and not a computer. They want the customers to perceive the interaction as a normal conversation flow. Kelly MovieBot accomplished this in the majority of the votes. Specifically, the chatbot received 20 votes on the fourth and the fifth rank. Furthermore, And Chill received fifteen votes in the first, second, and third ranks, indicating that the users had a feeling of interacting with a computer. The same pattern observed for Movie Bot application, with eighteen votes on the first three ranks. Check the figure relative to Questionnaire results for reference to the question results.

Question 7: Which feature(s) did you like the most in Kelly MovieBot?

Rating

Kelly MovieBot 

And Chill N/A

Movie Bot N/A

Comments: The public opinion highly voted (13/25) the feature *Rerun the whole conversation with only one response* as the top feature of the Kelly MovieBot. With that feature, the user can rerun the movie recommendation engine without first ending the conversation and then start again. The other features received an equal number of nine votes, indicating the neutral response of the public opinion over the elements of Kelly MovieBot. Check the figure relative to Questionnaire results for reference to the question results.

Question 8: Which feature(s) the other two chatbots had and you would like to see on Kelly MovieBot?

Rating

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Kelly MovieBot 

And Chill N/A

Movie Bot N/A

Comments: The majority of the users (12/25) voted the third feature, *YouTube trailer link per recommendation*, as an ideal good to have feature for the Kelly MovieBot. Next in preferences with nine and seven votes, respectively, are the features *Additional rating information apart from IMDB rating (e.g., rotten tomatoes)* and the *propose of movies based on favorite Actor or Oscar-winning Movies*. Those highlighted functionalities present opportunities to develop further and improve the human-computer interaction and the conversation flow between Kelly MovieBot and the user. Check the figure relative to Questionnaire results for reference to the question results.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

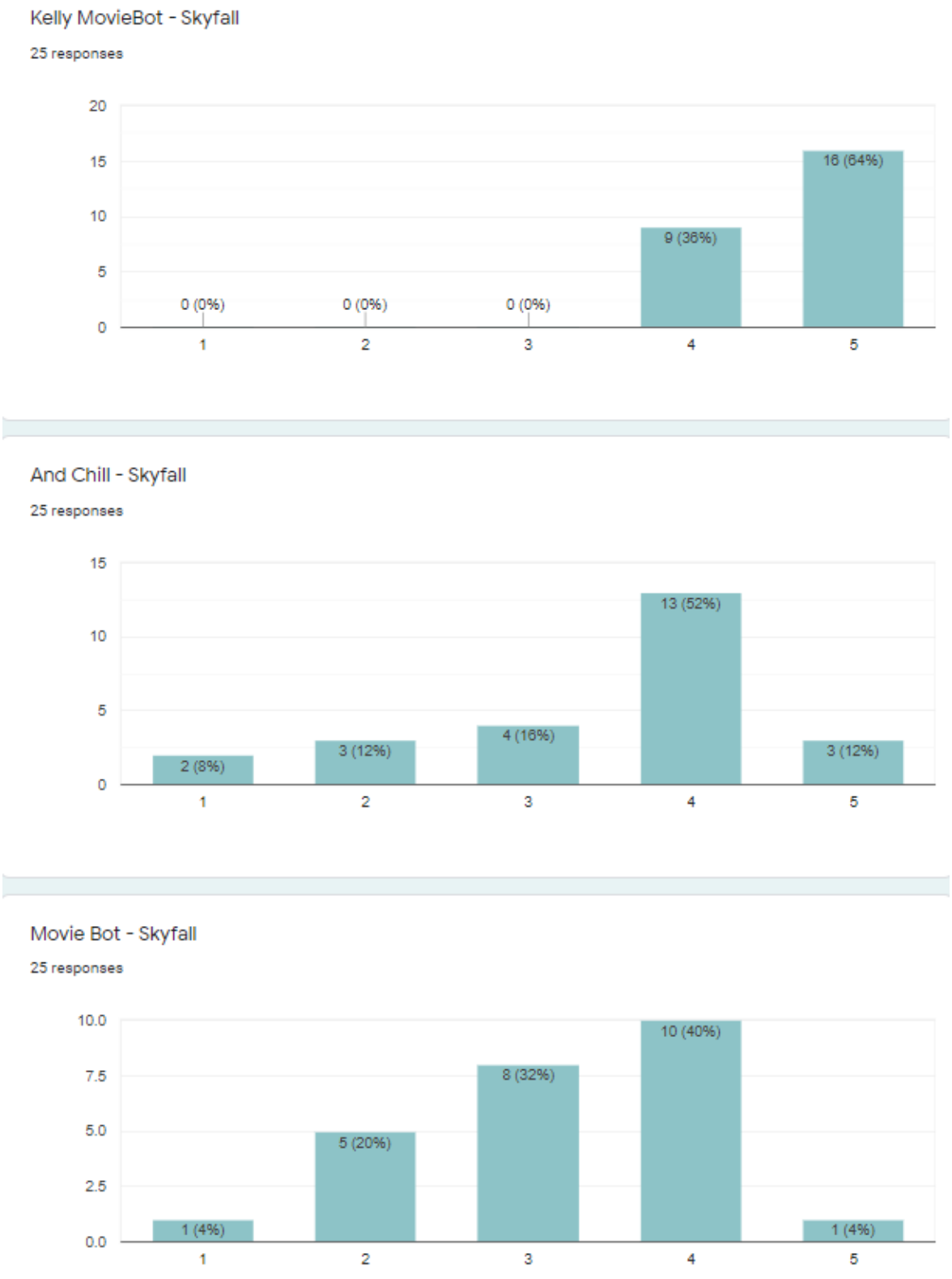


Figure 4.4 Question 1

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ



Figure 4.5 Question 2

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ



Figure 4.6 Question 3

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

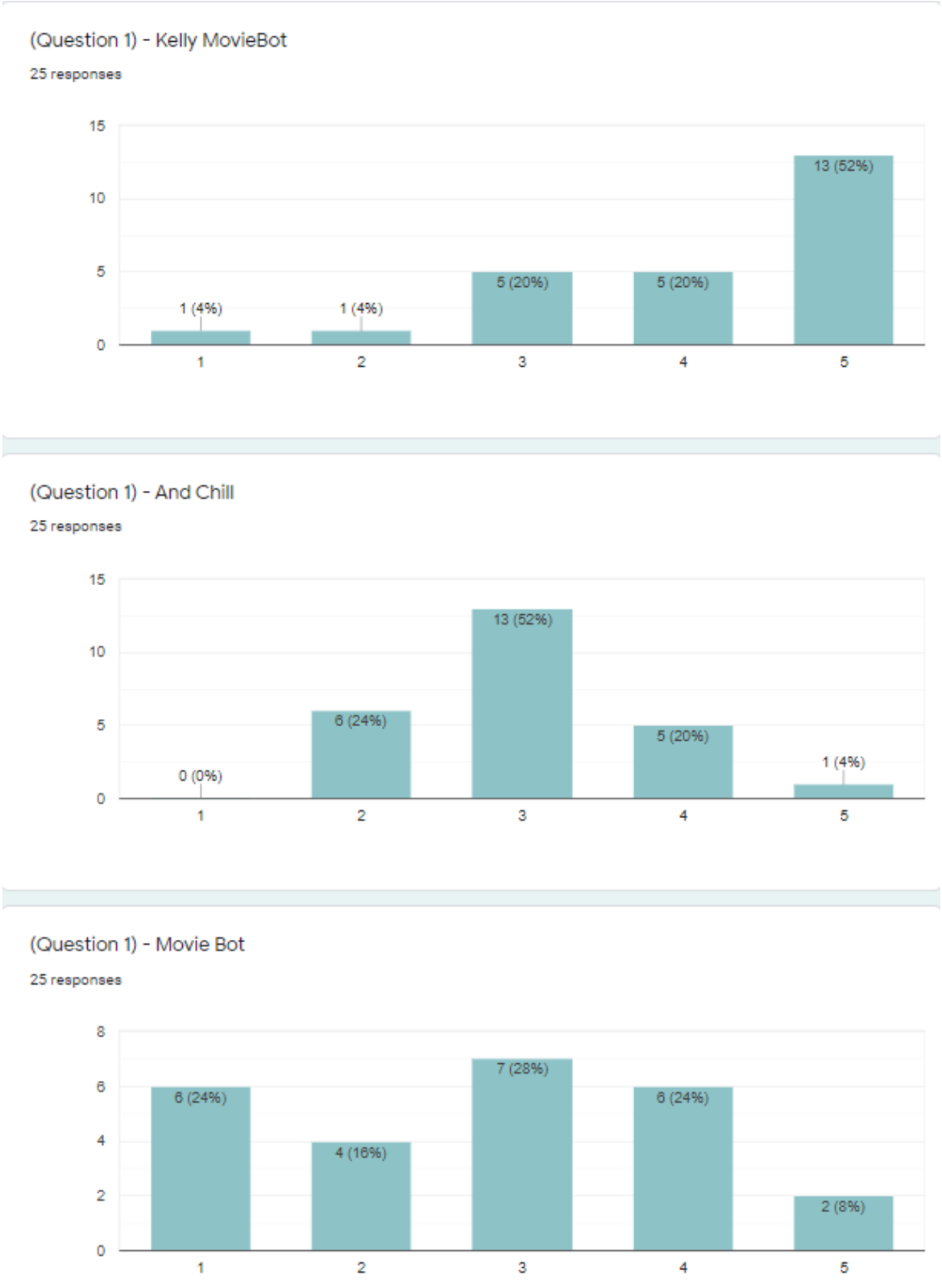


Figure 4.7 Question 4

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

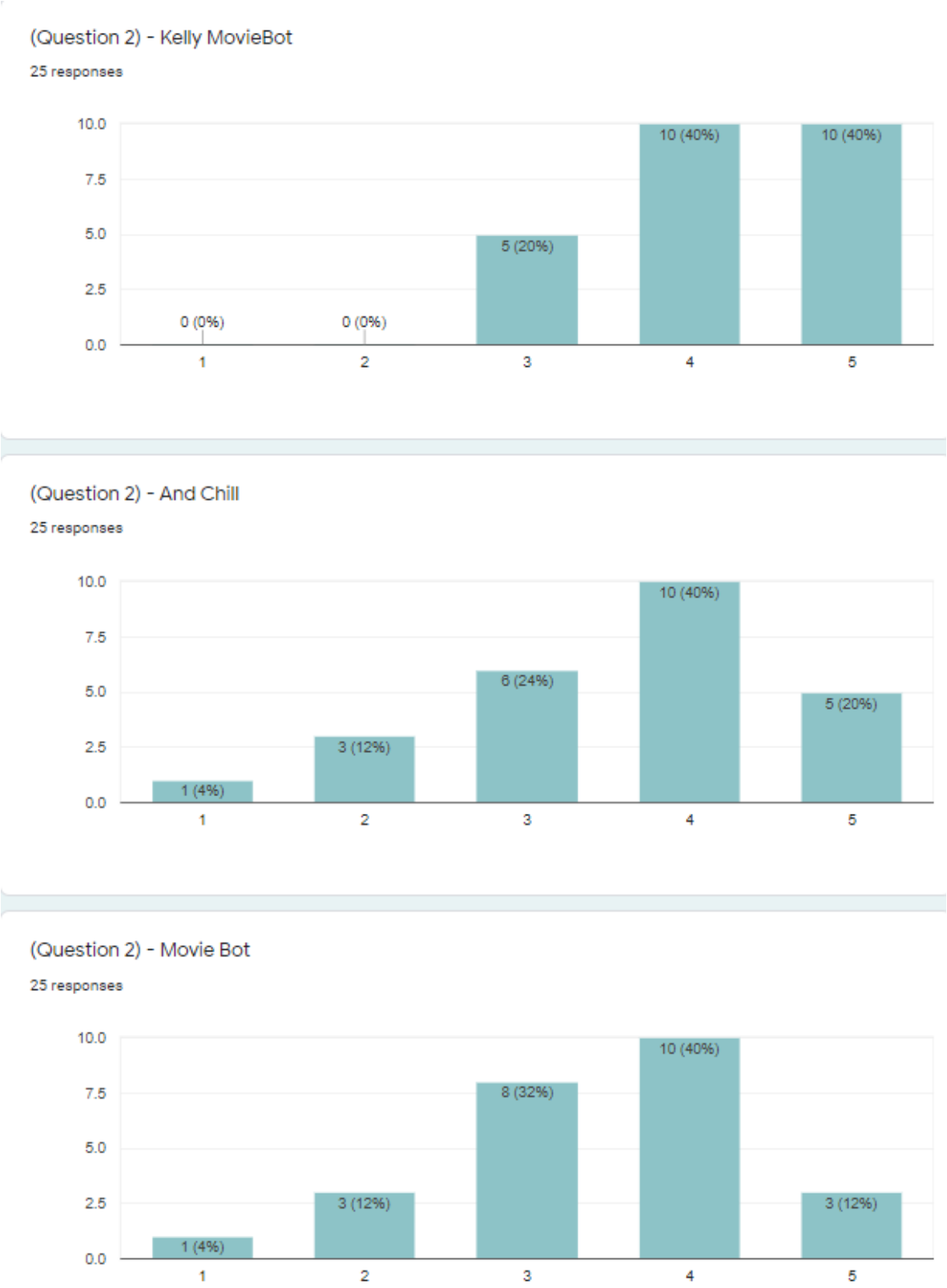


Figure 4.8 Question 5

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

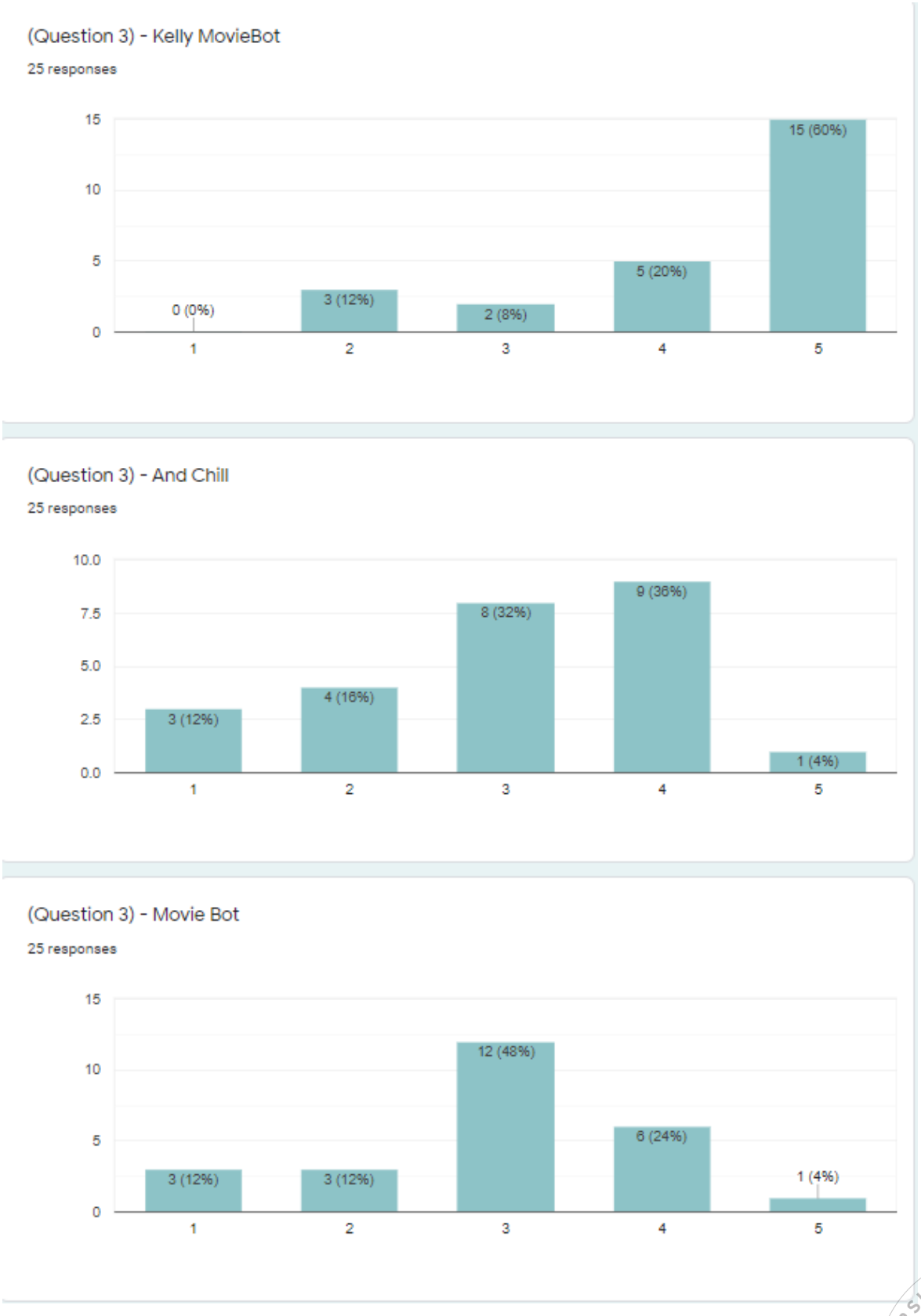


Figure 4.9 Question 6

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

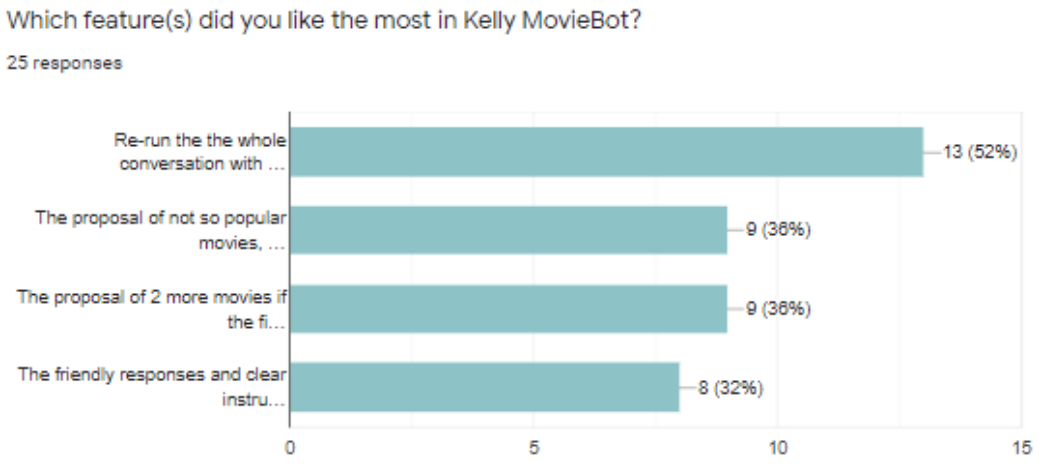


Figure 4.10 Question 7

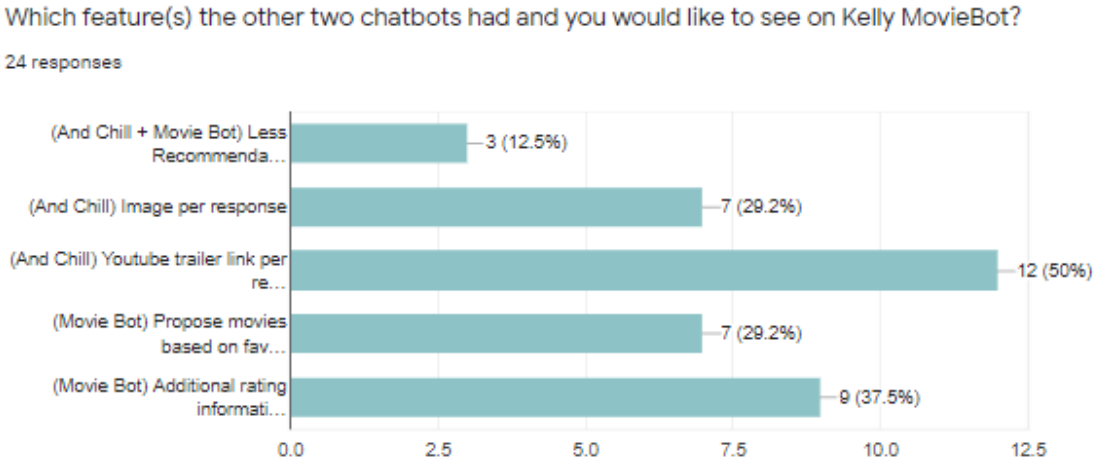


Figure 4.11 Question 8

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

4.4.3 Questionnaire conclusions

Take into consideration that the conclusions and results derived from a sample of twenty-five user responses.

Observing both the questionnaire results and the figures presented above, the authors derived the following conclusions. Initially, concerning the first topic monitored by the questionnaire, the recommendations proposed from *Kelly MovieBot* and *And Chill* are most relevant and highly voted by the users. On the first topic, *Kelly MovieBot* and *And Chill* received mostly the fifth and the fourth rank, indicating the overall user satisfaction over their recommendations. The applications proposed, in most cases, represent what the users would like to watch. Similarly, the users had the same opinion about the recommendations proposed by *MovieBot*. Although, the movie Titanic raised an exception. Most users highly voted the third rank from the available ranking scale, a neutral public opinion about the recommendations proposed by the third chatbot for the movie Titanic.

The second topic contains questions which monitor the general functionality and UI of the three applications. The first question evaluated if the number of recommendations proposed by each chatbot had a positive or negative effect on the user. The goal was to identify if too many recommendations confuse the user's judgment or too few recommendations force the user to select a movie already watched before. *Kelly MovieBot*, which proposed four movie recommendations to the user, received the fifth rank in half of the answers. The mechanism applied by *Kelly MovieBot* is to propose two movies, and then if the user has watched the movies already can select among the second pair of movies. In that manner, the recommendation algorithm executed by the first chatbot is two times more likely to propose a movie never seen before by the user than chatbots that propose only two movies or even a single movie. *And Chill* and *MovieBot* had fewer recommendation results based on the public opinion. The difference in the number of recommendations has not always bad effect on the reputation of the application. However, when an application provides options to the user, it is more likely to find movies the user has never watched before. The negative effect of recommending one or two movies is the recommendation of the same movie multiple times. Based on the second question of this topic, the public opinion expressed a positive reaction over the UI of the first two chatbots and a negative

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

reaction over the UI of the third chatbot. The negative opinion expressed indicates a non-so flexible and confusing user-journey over the application, from initiating the conversation up to recommendation proposal. The graphical user interface is a critical aspect of an application because it is what the users view and interact with. It is proved that public opinion prefers simple and interactive backgrounds. For the third and final question of this topic, the users questioned the human-computer interaction with the three applications. *Kelly MovieBot* stood out, since over half of the users awarded the chatbot with the highest rank on this question. The users get the feeling of interacting with a human agent rather than a strictly programmed computer application that follows specific steps. The same opinion also stands for the second application *And Chill*. On the contrary, the public vote for the third application was somewhere in the middle. About half of the users had a neutral opinion; however, six out of the total users voted that they completely experienced a straightforward application with no flexibility and strictly programmed steps.

With the third and final topic, the authors wanted to know which features of *Kelly MovieBot* made the difference in highlighting a useful application, and which features part of the functionality of the other two chatbots would be a good-to-have asset to equip *Kelly MovieBot*. On the essence of this topic, the authors wanted to extract insights about future versions of *Kelly MovieBot* to make the application more attractive to the users. Thus, the third topic is not comparison-driven but rather application improvement-driven. The feature of re-running the conversation using a single interactive button was the most highly voted functionality of *Kelly MovieBot*. The motive behind this functionality was to allow selecting another movie without ending the whole conversation and start over again by typing the same *starting phrase*. The *starting phrase* is a specific sequence of words that trigger the conversation with the chatbot. In most cases, this phrase is specific, and if the user does not write it correctly, the conversation will not start properly. Thus, allowing restarting the conversation with only one button is a necessary function for most applications. Finally, the display of the YouTube link per recommendation and additional rating information (e.g., the rotten tomatoes rating) were the two highest voted functionalities adopted by *And Chill* and *MovieBot* respectively. Adding the previous two functionalities in Kelly MovieBot will boost the reputation of the application and increase the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

positive user's opinion.

Overall *Kelly MovieBot* gathered positive opinions from the twenty-five users. The application's functionality as an interactive conversational agent managed to surpass the functionality, recommendations, and UI of two already functional and usable chatbots of the Facebook e-market. *And Chill* had a similar effect, with *Kelly MovieBot*, on the public user opinion since it is an already well-established application with many online users. Thus, the second chatbot was the main rival of the first chatbot. *MovieBot* came third in user votes, enhancing the positive public opinion for *Kelly MovieBot*. Those facts indicate that the predictions of the neural network, the recommendation algorithm, and the chatbot structure have succeeded in creating a movie recommendation agent capable of proposing similar and even better recommendations than already available chatbots.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter Five

Conclusions

In the final chapter, the documentation will refresh to the reader the conclusions derived from the third and the fourth chapter of the thesis. Those two chapters explicitly document the research and the experimentation part conducted by the authors of the thesis.

The third chapter of the documentation thoroughly presented the experimentation with Natural Language techniques and deep learning to solve the multi-genre classification problem. One of the concepts discussed across the thesis introduction. Apart from the final model classifiers presented and the comparisons made, the authors conducted much experimentation related to the structure of the neural networks; experimentation that it was never reported in the document. Based explicitly on that experimentation and testing on different neural networks to solve the problem of multi-genre classification and build a model estimator to associate movies to their genre sequence correctly, the authors derived the following conclusions:

- Reducing batch size can produce a better model. Even though the document mentions that a larger batch size means more information learned during the neural network training, setting the batch size too high can reverse the results. *It has been observed in practice that when using a larger batch, there is a degradation in the quality of the model, as measured by its ability to generalize*, as referenced by this excellent paper [11]. Recall that lack of generalization cause the model classifier to overfit. Learning very accurately, the training data due to a fast converge

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

into a local minimum, but it is difficult for the model estimator to predict the test target values correctly. Thus, the authors suggest testing both the performance of the model and the time taken to train it since the batch size value affects both.

- Removing or adding a dense layer(s) can improve or deteriorate the results. Initially, the structure of the custom-created neural network included two dense layers with the same configuration. Although, when the authors removed the second dense layer, the training and validation results improved. Keep in mind that dense layers add complexity to the model, which can sometimes have negative results on the classifier's overall performance.

- Removing regularization affected the results of the current experiment. This fact bounds to the specific experimentation and training of the model classifier. A researcher should test high, low, and none values for the regularization before approving the final state of the model's structure.

- Learning rate is one of the most important parameters, and it is the number one choice for tuning. If someone can only afford to tune one model parameter, due to a lack of resources, the learning rate is the default choice. In the initial model experimentation, the neural networks used the default learning rate values of the relative optimization functions (e.g., Adam, SGD, RMSprop). When the learning rate is static and does not dynamically change after a specific number of epochs, it can lead to conservative model classifiers that cannot learn at their optimum potential. Two are the best practices to follow when deciding the value of the learning rate. First, tune the parameter by choosing a wide range of three and more choices with the optimizer's default value always among the available options. In such a way, the model classifier will train on various learning rate values, and the researcher can monitor the performance of the model better. Secondly, set a high value of the learning rate and decrease it every N number of training epochs (e.g., every ten epochs decrease the learning rate by half).

- Drop out rate should also be used with caution, especially when the dataset trained is relatively small. High values of drop-out rate with a small dataset can create under-fitted model classifiers since a high percentage of the trainable parameters dropped. On the contrary, low values of

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

drop-out rate on relatively big datasets can increase the neural network's training time and quickly consume most of the memory available for training the neural network.

- Embeddings dimension is also playing an important role in the training of neural networks dedicated to solve text classification problems. It is critical to tune the specific parameter of the embeddings layer because it can substantially improve or deteriorate the classifier performance. The thesis documented one such phenomenon, observed between the custom-made neural network and the neural network trained on the GloVe embeddings. Using a smaller value of the embedding size vectors, specifically, 100-dimensions, produced a better model classifier than the one trained with the GloVe pre-trained embeddings and 300-dimensions on the embedding layer.

The points mentioned above were observed during the testing and the experimentation with the structure of the custom-made model classifier.

Moreover, the authors concluded based not only on the relative documentation but also on the experiments results that hamming loss is the most appropriate choice to measure the model estimator's performance and evaluate the model predictions on never-seen-before data. Hamming loss is the default option to evaluate multi-label classification models, compared to the commonly used metric of classification performance, the binary accuracy. Using three different optimization algorithms to optimize the loss metric of the model estimator, the authors made the following conclusions:

- Adam optimizer is a fast learner. Observing Adam's learning curves on the training and validation data, Adam optimizer converged very quickly at the lowest value of the training and validation loss. Adam optimizer also demonstrated the smoothest and most stable learning curves compared to the other two optimization algorithms.

- RMSprop optimizer had an almost identical performance with the Adam optimizer; however, it took to the optimizer a slightly higher number of epochs to learn the dataset's semantics and correctly predict the target values.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

- SGD optimizer demonstrated the worst results of the three optimizers when trained and tested on the specific movie dataset.

Finally, to solve the multi-genre text classification problem, five neural networks deployed. The first neural network was a custom-made sequential neural network. The second neural network used the pre-trained GloVe embeddings, while the other three neural networks were pre-trained and saved models hosted on the Tensorflow Hub community. The custom-made neural network outperformed the other four model classifiers by achieving the lowest test hamming loss value, the lowest bias and variance tradeoff, the smoothest and most stable learning curves, and finally, the most accurate predictions on each of the seventeen target movie genres. The best model selected has the following hyper-parameters:

- Embeddings dimension: 100
- Batch size: 32
- Learning rate: 0.001
- Decay steps multiplier: 10
- Optimizer: Adam

In the fourth chapter of the thesis documentation, the authors presented

1. The architecture of a custom recommendation algorithm
2. The creation of the environment to host the recommendation algorithm
3. An opinion-based comparison between the custom-created conversational agent and two already functional and usable chatbots of the Facebook e-market

For the recommendation algorithm's architecture, the authors extracted the weights of the best model classifier and created for each movie a 1248-dimensional vector. The multi-dimensional vector, the user feedback, and the movie content are used by the recommendation algorithm to propose four recommendations movies. The authors chose Dialogflow and the environment of a chatbot to host the

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

recommendation algorithm and to develop a more user-friendly movie recommendation application. Finally, the opinion-based research conducted by the authors compared *Kelly MovieBot* with *And Chill* and *MovieBot*. The questionnaire gathered twenty-five responses in total and featured *Kelly MovieBot*, the chatbot assembled by the authors, the top voted application for most of the questions. Closest in preference was the *And Chill* chatbot, a very well-established application, while *MovieBot* satisfied the least the public opinion. Embracing *Kelly MovieBot* with positive ratings, clearly puts the latter on a strong competitive position against interactive conversational agents currently in the market.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

APPENDIX

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Appendix A

First Appendix

A.1 List of Abbreviations

Table A.1 Abbreviations

Abbreviation	Explanation
NLP	Natural Language Processing
AI	Artificial intelligence
IMDB	Internet Movie Database
OOV	Out-of-Vocabulary
TF Hub	Tensorflow Hub
e-market	Electronic (online) market
FMCG	Fast Moving Consumer Goods
SVM	Support Vector Machines
IDE	Integrated Development Environment
URL	Uniform Resource Locator
API	Application Program Interface
HTTPS	Hypertext Transfer Protocol Secure
TV-series	Television series
TV-shows	Television shows

(To be continued)

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Abbreviation	Explanation
Sci-fi	Science-fiction
GloVe	Global Vectors for Word Representation
2D	2 dimensions
3D	3 dimensions
ROC-AUC	Receiver Operating Characteristic - Area Under the Curve
SGD	Support Gradient Descent
RMSprop	Root Mean Square propagation
TP	True Positive
TN	True Negatives
FP	False Positives
FN	False Negatives
GUI	Graphical User Interface
UI	User Interface

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

References

[1] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

[2] Michel Jose Anzanello and Flavio Sanson Fogliatto. *Learning curve models and applications: Literature review and research directions*. https://www.sciencedirect.com/science/article/abs/pii/S016981411100062X. ELSEVIER, 2011, pp. 573–583.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[4] Yoshua Bengio Xavier Glorot Antoine Bordes. *Deep Sparse Rectifier Neural Networks*. Research Gate, 2014, pp. 317–318.

[5] François Chollet. *Deep Learning with Python*. Shelter Island, NY 11964: Manning Publications Co., 2015.

[6] François Chollet. *Deep Learning with Python*. Shelter Island, NY 11964: Manning Publications Co., 2015, p. 187.

[7] François Chollet. *Deep Learning with Python*. Shelter Island, NY 11964: Manning Publications Co., 2015, p. 46.

[8] François Chollet. *Deep Learning with Python*. Shelter Island, NY 11964: Manning Publications Co., 2015, p. 49.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016, p. 194.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016, p. 111.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

[11] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. 2016. arXiv: 1609.04836 [cs.LG].

[12] Noam Shazeer et al. *Swivel: Improving Embeddings by Noticing What's Missing*. 2016. arXiv: 1602.02215 [cs.CL].

[13] Matthew Mayo. *A General Approach to Preprocessing Text Data*. 2017. URL: https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html.

[14] Sebastian Ruder et al. "Transfer Learning in Natural Language Processing". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 15–18. DOI: 10.18653/v1/N19-5004. URL: https://www.aclweb.org/anthology/N19-5004.

[15] Awan-Ur-Rahman. *What is Data Cleaning? How to Process Data for Analytics and Machine Learning Modeling?* URL: https://towardsdatascience.com/what-is-data-cleaning-how-to-process-data-for-analytics-and-machine-learning-modeling-c2afcf4fbf45.

[16] Jason Brownlee. *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. URL: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[17] Jason Brownlee. *Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning*. URL: https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/#:~:text=Bias.

[18] Jason Brownlee. *How to use Learning Curves to Diagnose Machine Learning Model Performance*. URL: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

[19] Chris. *Classifying IMDB sentiment with Keras and Embeddings, Dropout & Conv1D*. URL: https://www.machinecurve.com/index.php/2020/03/03/classifying-imdb-sentiment-with-keras-and-embeddings-dropout-conv1d/.

[20] Omar Elgabry. *The Ultimate Guide to Data Cleaning*. URL: https://towardsdatascience.com/the-ultimate-guide-to-data-cleaning-3969843991d4.

[21] Elvis. *A Light Introduction to Transfer Learning for NLP*. URL: https://medium.com/dair-ai/a-light-introduction-to-transfer-learning-for-nlp-3e2cb56b48c8.

[22] Scott Fortmann-Roe. *Understanding the Bias-Variance Tradeoff*. URL: http://scott.fortmann-roe.com/docs/BiasVariance.html.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

[23] Python Software Foundation. *pickle - Python object serialization*. URL: https://docs.python.org/3/library/pickle.html.

[24] Daniel Godoy. *Understanding binary cross-entropy / log loss: a visual explanation*. URL: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

[25] Google. *Intents*. URL: https://cloud.google.com/dialogflow/docs/intents-overview.

[26] Grouplens. *Movielens Dataset of IMDB movies*. URL: https://grouplens.org/datasets/movielens/.

[27] Hunter Heidenreich. *Stemming? Lemmatization? What?. Taking a high-level dive into what stemming and lemmatization do for natural language processing tasks and how they do it.* URL: https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8.

[28] Hafsa Jabeen. *Stemming and Lemmatization in Python*. URL: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python.

[29] ngrok. *ngrok*. URL: https://ngrok.com/product.

[30] Selva Prabhakaran. *Cosine Similarity – Understanding the math and how it works*. URL: https://www.machinelearningplus.com/nlp/cosine-similarity/.

[31] NLTK Project. *NLTK*. URL: https://www.nltk.org/.

[32] Kenneth Reitz & Real Python. *Data Serialization*. URL: https://docs.python-guide.org/scenarios/serialization/.

[33] RStudio. *Layer that concatenates a list of inputs*. URL: https://tensorflow.rstudio.com/reference/keras/layer_concatenate/.

[34] scikit-learn. *Model persistence*. URL: https://scikit-learn.org/stable/modules/model_persistence.html.

[35] scikit-learn. *sklearn.feature_extraction.text.CountVectorizer*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature%5C_extraction.text.CountVectorizer.html.

[36] scikit-learn. *sklearn.metrics.hamming_loss*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming%5C_loss.html.

[37] scikit-learn. *sklearn.metrics.precision_score*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision%5C_score.html.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

[38] scikit-learn. *sklearn.metrics.recall_score*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall%5C_score.html?highlight=recall#sklearn.metrics.recall%5C_score.

[39] scikit-learn. *sklearn.metrics.zero_one_loss*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.zero%5C_one%5C_loss.html#sklearn.metrics.zero%5C_one%5C_loss.

[40] Shivangi Singhal. *Processing Text data in Natural Language Processing*. URL: http://datascience.sharerecipe.net/2019/06/14/processing-text-data-in-natural-language-processing/.

[41] Sivakar Siva. *The Data Science Life-Cycle*. URL: https://towardsdatascience.com/stoend-to-end-data-science-life-cycle-6387523b5afc.

[42] Wikipedia. *Data Analysis*. URL: https://en.wikipedia.org/wiki/Data-analysis.

[43] Wikipedia. *Data cleansing*. URL: https://en.wikipedia.org/wiki/Data_cleansing#Data_quality.

[44] Wikipedia. *HTML*. URL: https://en.wikipedia.org/wiki/HTML.

[45] Wikipedia. *Parsing*. URL: https://en.wikipedia.org/wiki/Parsing.

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ