UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Modeling book to movie adaptation recommendations using machine learning

Author:
Solomiia DUBNEVYCH

Supervisor:
Andriy GAZIN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Solomiia DUBNEVYCH, declare that this thesis titled, “Modeling book to movie adaptation recommendations using machine learning” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Books and movies are like apples and oranges. They both are fruit, but taste completely different"

Stephen King

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Modeling book to movie adaptation recommendations using machine learning

by Solomiia DUBNEVYCH

Abstract

The art of adapting existing text for the film is a challenge for the scriptwriter. This study was conducted to develop the tool to generate tips on the book-to-movie adaptation analyzing existing scenarios. To do it the Natural Language Processing techniques and Machine Learning were tried. The hypothesis of using Neural Networks as the main approach were denied. However, it is very efficient as a helping tool.

The result of this work is a web tool for book and script analysis visualization and recommendations generation. It is the instrument for story exploratory analysis created for scriptwriters and fandom members.

Acknowledgements

I want to thank my family for the endless support they expressed during nowadays hard times.

I am also grateful to my supervisor Andriy Gazin for guiding me in the right direction, his ideas, time, and effort.

Special thanks to Kartel, my small university band, which became a family during these four years.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
LSTM	**L**ong **S**hort **T**erm **M**emory
RNN	**R**eccurent **N**eural **N**etwork
BOW	**B**ag **O**f **W**ords
NER	**N**amed **E**ntity **R**ecognition
MVP	**M**inimum **V**iable **P**roduct

Dedicated to my family and all the rest bookworms on our planet

Chapter 1

Introduction

1.1 Prehistory

There is a claim that the best movies are those which are based on books. It can be easily proven if we look at the statistics. Each fifth movie on the IMDb "Top 100" rating is based on a paper-written story.

Of the twenty films with the highest sales ever, only six have original stories, the remaining fourteen are based on printed materials such as books, comics, diaries, and more. There are several reasons for choosing an already published story over a new scenario, but the main is to mitigate the risk associated with new movie production.[5]

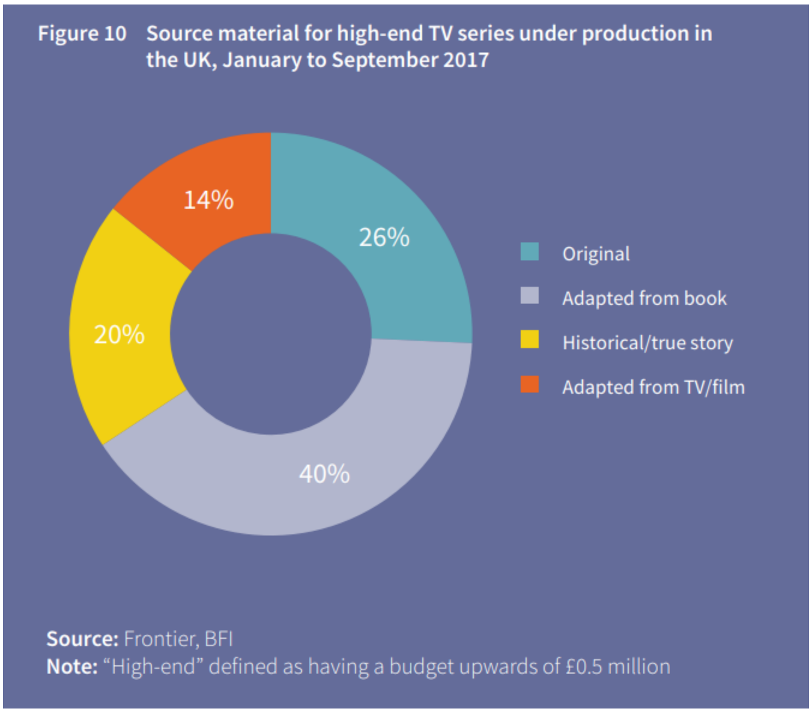

FIGURE 1.1: Source material for high-end TV series under production in the UK, January to September 2017

The best source for another hit in the industry is, undoubtfully, already known among the audience. Firstly, fans' blogs and forums are a good source of ideas for the adaptation. Secondly, they are expecting the product and advertising it unintentionally. And, finally, they are those, who can properly evaluate the film, as "to experience it as an adaptation, however, as we have seen, we need to recognize it as such and to know its adapted text, thus allowing the latter to oscillate in our memories with what we are experiencing"[2] .

1.2 Problem Statement

Knowing audience usually is not satisfied with the film adaptation of a book. Movie critics focus on budget, actor play, background music, and many other factors. On the contrary, fans' center of attention is the scenario, more specifically - plot correspondence to the original story. The hardest work for the screenwriter is to meet the expectations of the knowing audience. To give an example, my friends frequently catch me on the following phrases: "It was different in a book", "Why is this character so unlike..?" and even "How dare they cut off the scene of...?".

It should be reminded that the adaptation is just another form of film. It is not a rule that a movie of such type should concentrate on pleasing the knowing audience, its main aim is still making a story alive on the screen.

The focus of this work is to help the screenwriter, which struggles to meet the demands of viewers, critics, and the industry as a whole, to create the best version of the script for the well-known or not so story.

The core challenge in the adaptation process is the actual format difference. While the original text can exceed a thousand pages, the duration of the film is usually limited to two hours. As a consequence, dialogues, scenes, characters, and even the whole storylines may be cut from the scenario. Hence, our problem solving approach is fidelity to the story.[1]

1.3 Goals & Objectives

The purpose of this research is to develop a tool that makes it easy for screenwriters to overview book texts and compare them with a new-written script.

We cross-analyze books versus movies using Natural Language Processing (NLP) techniques to find similar patterns in stories, trace character development through the timeline, and find censor changes. As a result of the project, it is expected, that the screenwriter uploads the book of his interest into the system, and gets visualized analysis and book text with highlighted elements.

Chapter 2

Related Works

As soon as people teach the computer to play chess, they start dreaming of the machine outmatching the human mind. To succeed, it should learn how to imagine and create things we call art: paintings, sculptures, music, and stories. People rely on their senses: sight, hearing, smell, taste, and touch. We express our feelings and previous experience in the piece of art. Now, we need to teach the machine to feel.

There were a few types of works we have encountered investigating the book-to-movie adaptation topic. The main difference between them is in the definition of the problem they are trying to solve, thus in the solution as well.

The first type is projects developed for the analysis of either book or script text of a researcher's choice. They most frequently center on character dynamics throughout the time using exploratory analysis. The output of such a study is summarised insights for a specific book or movie. For instance, a project analyzing the most popular characters and their words in the first 3 parts of the Harry Potter series. (Harry Potter and the text mining)

In so-called example-based works, AI train on prepared existing scripts to generate the original script from the input description, including the start scene, main characters, first action, etc. The basic principle is to teach the machine to predict the word, phrase, or sentence, which goes after another one. A similar logic is used for the auto-complete function in messengers and searchers.

In 2016, filmmaker Oscar Sharp and researcher Ross Goodwin produced AI experimental science fiction scenario. It was generated by a Long short-term memory (LSTM) recurrent neural network (RNN) AI, called Benjamin. It was trained on science-fiction movie scenarios to create scenes, characters, and dialogues and even wrote the accompanying song text. [4]

Although the idea is exciting, the outcome is extremely amusing. A 9-minutes video, based on the generated script, is full of illogical phrases combination and bizarre scenes.

There are more advanced variations of script generators. For example, Masterpiece Generator (plot-generator.org) creates an integrated, unique story, but it requires a full list of entry parameters to be specified.

As we mentioned in the Introduction chapter, the problem we are trying to solve is book-to-movie adaptation complexity. We call this type of work story-based. There is no point in teaching the computer to imitate the scriptwriter, trying to adapt the book to screen, because our task is not to create the original story but to adapt the existing one.

We discovered a few projects attempting to train the neural network to build the script from book text on open source platforms. By way of illustration, the RNN project for Harry Potter script generation (Harry Potter script generation). As in the example-based case, the results are messy and cannot be used in film-making for now.

Given the above, we decided to choose the adaptation recommendation visualization tool as the approach to this study.

Chapter 3

Data Overview

3.1 Data specification

There are two types of texts analyzed in this study: books and scripts. The Goodreads list "Books Made into Movies"[3] was taken as the reference for database creation. We collected 120 pairs for our research, which vary in genre year of production and rating.

The main source for the scripts set is IMSDb (imsdb.com). This website provides free to read and download movies scripts. Additional sources of data are listed below:

- Actorpoint (actorpoint.com),
- AwesomeFilm (awesomfilm.com),
- The Script Savant (thescriptsavant.com),
- thescriptlab (thescriptlab.com),
- SimplyScripts (simplyscripts.com)

Files containing movie scripts are named with the original story title and have a .pdf or .txt extension. The example of a script page is provided in Attachment Figure 2.

Books texts usually are not free to access, so we collected them from open sources with the restriction to use only for educational purposes. Some of them are listed below:

- pdfdrive (pdfdrive.com),
- pdfcoffee (pdfcoffee.com),
- bookspdf4free (bookspdg5free.com),
- twirpx (twirpx.com)

Files containing book texts are named correspondingly to the scripts plus the prefix "Book-", and have a .pdf extension.

3.2 Preprocessing

Before starting actual work, we needed to prepare the data. There were two stages of cleaning: manual and automated.

Firstly, we transformed each file from .pdf format into .txt, and manually cut off sections such as:

- Half-title page
- Title page
- Copyright page
- Dedication
- Epigraph
- Table of Contents
- Foreword
- Preface
- Acknowledgements
- Introduction

Secondly, we used BOW (bag-of-words) model to represent our data in fixed-length vectors. To prepare the text data for the model building we perform the following NLP pre-processing steps:

1. Lower casing
2. Tokenization
3. Lemmatization
4. Removing Stop words

Our first step is converting words into lower case. Next, we split the text into tokens and exclude punctuation. For our project, we select words as tokens. The third step is reducing words to their root form. For this use WordNetLemmatizer from nltk library. Finally, we remove words that are not useful for the analysis, using the stopwords list again from nltk library.

Chapter 4

Methodology

The main idea is the instrument created to help the scriptwriter. It should interpret the book from different perspectives using text mining and machine learning. We perform data analysis with 2 purposes: exploratory review and tool feature definition.

4.1 Data Analysis

There are many important details in a book text, which cannot be noticed with an unaided eye.

For example, it is hard to see the role of characters in different parts of the story. Readers, viewers, and industry professionals are biased when it comes to character importance, as they have their favorite and disliked ones. It is useful for the scriptwriter to know in which chapter the character is in the spotlight to make enough time for him/her in the movie or which characters can be cut off from the story at all.

There is also a need to trace the story locations: where it starts and ends, how many different places are described in the text, which is more popular and less, etc. Having it visualized, the scriptwriter can list the top exterior and interior action locations and start shaping the story in the scenario.

One of the trickiest tasks in the scenario creation process is story censoring. It is widely known, that movies always struggle to be allowed within different age categories. The lower the PG rating is, the bigger audience your film has. Automatic profanity detection in each sentence and its visual representation is one of the jobs the machine can handle faster than a man.

Last but not least, the story's emotional charge is what the scriptwriter should always pay attention to. The sentiment text analysis may be practical for this task.

To start the fact-finding analysis we need to split the text into smaller pieces. As there are books in which chapters are not defined and scenarios de facto have no parts specified, we divide texts into equal chunks.

Each story, either script or book, contains a different amount of sentences. To be able to compare them through time, we normalize 'coordinate system' by defining 'chapters'.

These 'chapters' are sets of N sentences, where

$$N = total\_sentence\_amount // 45$$

4.1.1 Characters

The **flair** library is used for named entity recognition (NER). It classifies sentence parts into 4 different types: person (PER), location (LOC), organization (ORG), other name (MISC). (github.com/flairNLP/flair)

For example:

```
Sentence: "George Washington went to Washington ." →

["George Washington"/PER, "Washington"/LOC]
```

We recognize persons in text, clear mistakenly tagged ones like 'Okay', 'Hello', 'Aye', etc., convert them into lower case and count the repetitions. As a result, we get the occurrence of each character by chapter. To get the top 10 characters referenced in the story, we count their mentions in the text and sort the result in descending order. To explain character development history, we build a heat-map of their occurrences on the timeline.

As an illustration, here is Character distribution in 'The Lord of the Rings: Fellowship of the Ring' by J. R. R. Tolkien followed by the character representation in the movie.

In Figure 4.1, we observe a rating of accents put on the character in the book versus the movie. It contains the top 10 characters, sorted alphabetically. This is a heat map, where light green squares mark a little percentage of occurrences and dark blue marks a big amount. Using this visual approach, we can trace character importance in different parts of the story.

It can be stated that in the book Frodo is at the center of attention as the main character. However, we notice that he is not so favorable in the movie. Gandalf, Merry, Pippin, Sam, and Aragorn (also known as Strider) are referred to more often in adaptation than in the original story. It is also interesting to know that Legolas and Gimly, who are considered to be in the main personas set by a viewer, are not in the top 10. It is eye-catching that in the movie, part 14, there are only two characters highlighted - Gandalf and Saruman. We assume, it is the scene of Saruman's exposure. We can see the culmination fight, where Boromir and Frodo outweigh the other characters. And this explanation of the Character Occurrence through time may be continued.

In Figure 4.2, we can see the general overview of the character occurrences in the book versus the script.

We discover the total amount of references to the top 10 characters and sort them by occurrences in the script.

The top 3 personas are the same both in the book and script. But, the next places in the rating are mixed up, meaning that these characters are shown differently in two pieces of text.

FIGURE 4.1: Character Occurrence through the time in The Lord of the Rings: Fellowship of the Ring

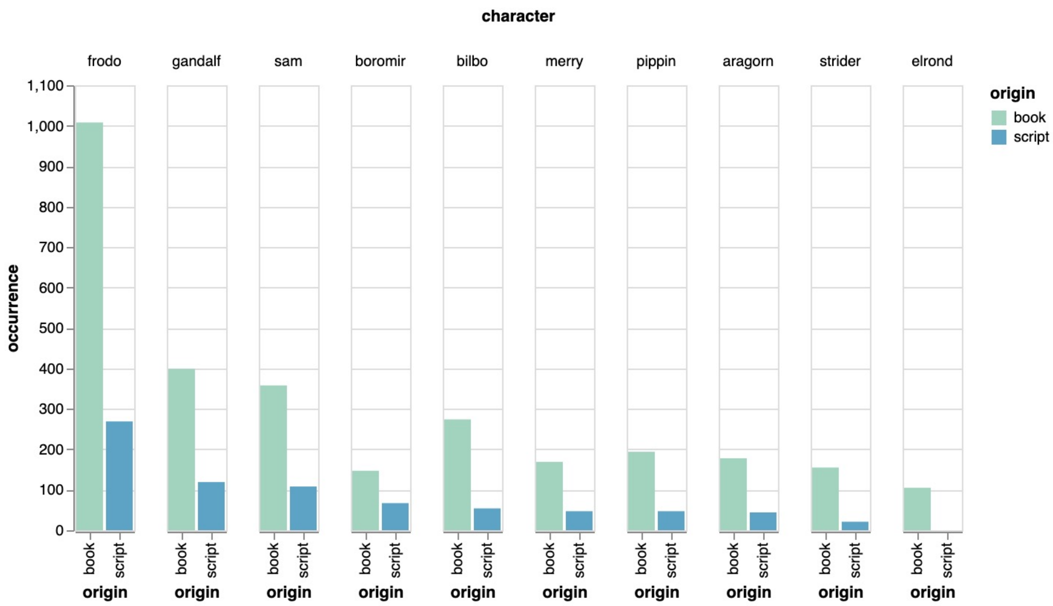

FIGURE 4.2: Total Character Occurrence in The Lord of the Rings: Fellowship of the Ring

4.1.2 Profanity

The **better_profanity** library is used for profanity detection by analyzing strings. In comparison with other libraries, this one is not based on ML, which makes it faster. It recognizes a swear word even if the part of it is censored by symbols: '4', '*', '@', etc. If the sentence contains swear words, the method returns True, in the other case - False. (pypi.org/project/better-profanity)

As an example, here is the profanity percentage compared to 'The Martian' by Andy Weir.

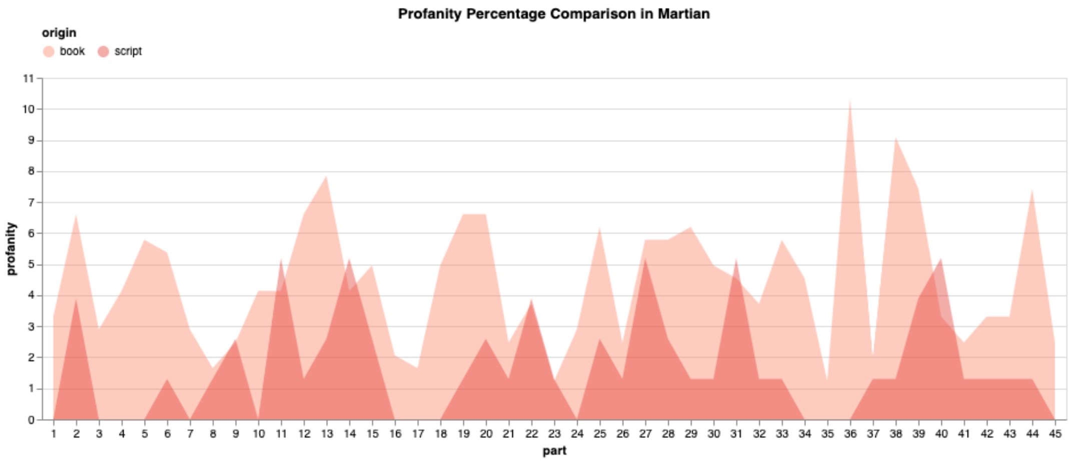

FIGURE 4.3: Profanity Percentage Comparison in The Martian

In the Figure 4.4, we can see that in this specific text, there are no chapters in the book without swear words. Nevertheless, the movie adaptation is more censored. If we have a good look, we can notice the pattern of profanity percentage throughout the story. There are peaks highlighting the chapter mood, and the lifts and falls follow a similar shape too.

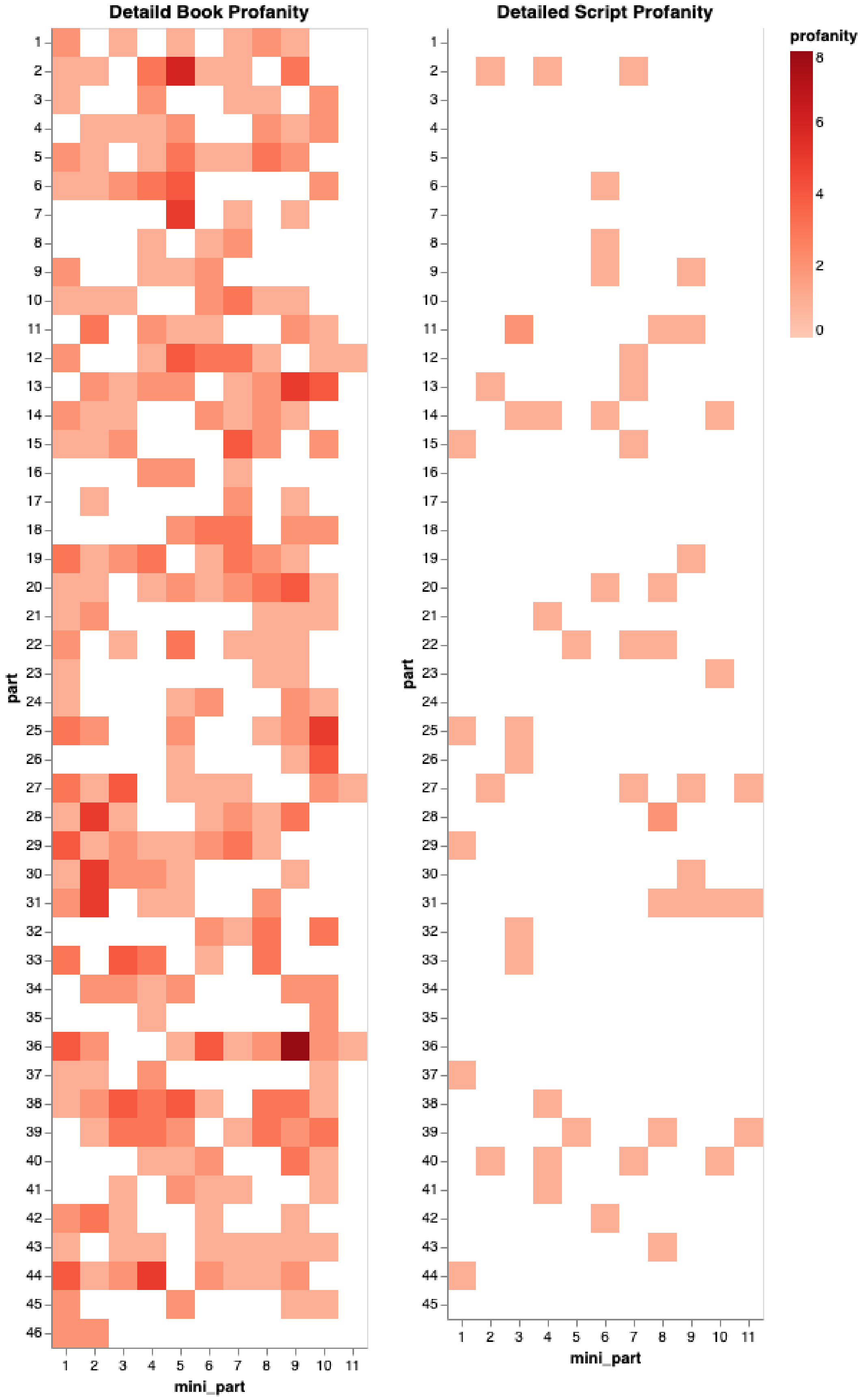

FIGURE 4.4: Detailed Profanity Comparison in The Martian

4.1.3 Sentiment

The **flair** library is used for sentiment analysis of text pieces. It classifies the sentence into positive, negative, or neutral in percentages.

For instance, here is a sentiment analysis for 'Coraline' by Neil Gaiman.

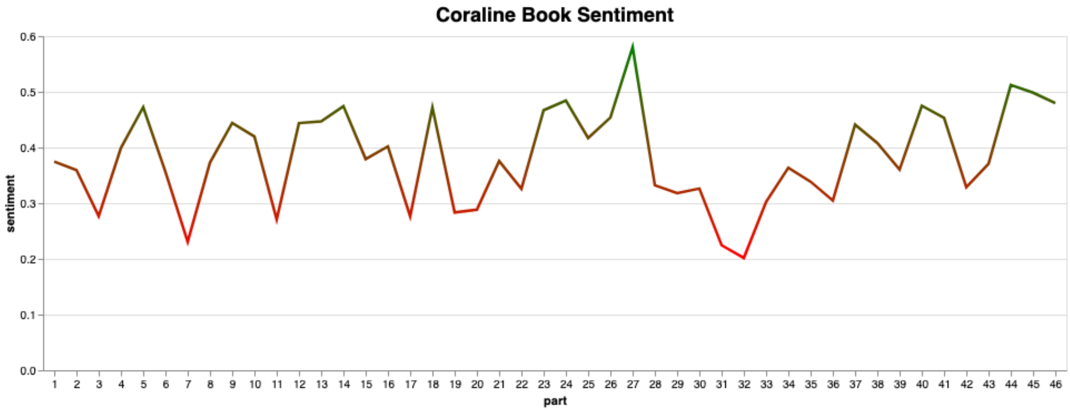

FIGURE 4.5: Sentiment analysis in Coraline book

In the Figure 4.5, we see the percentage of positivity in book chapters. It is green when the chapter is rather positive and red, if negative. We can say that 'Coraline' is a gloomy story, as its chapter is more often negatively charged.

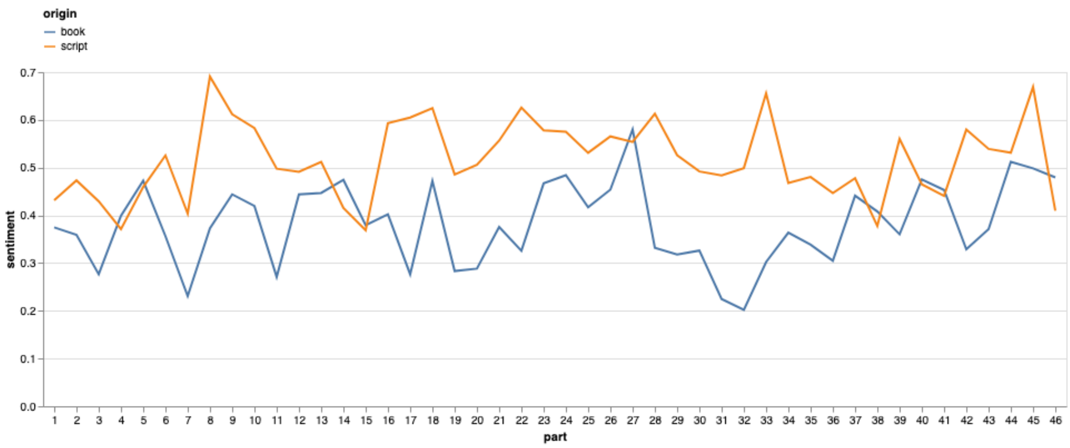

FIGURE 4.6: Book VS. movie sentiment analysis in Coraline

In the Figure 4.6, we see that the story sentiment in the script is much more positive than in the book. Although there are a few similar patterns on the chart, we can only assume that they refer to the same episodes in texts. This visualization is the story mood overview - not a detailed emotions representation.

4.1.4 Location

This analysis is implemented with the same tool as Characters (NER) and with the same methods. It can give us an understanding of the main places in the story and the characters' journey through them. It is important to say, that this analysis may be valid for linear stories, such as Lord of The Rings or Life of Pi. But stories with a lot of places in parallel may give us a confusing result.

FIGURE 4.7: Locations distribution in Book VS. Movie

In Figure 4.7 we can discover the main locations mentioned in the texts. In this particular example, places in the movie scenario are shortened to a smaller amount. Also, we can see that Neural Network is not accurate as it recognizes 'pacific' and 'pacific ocean' as two unique locations.

4.2 Modeling

Our task is to validate if ML can be applied to the book-to-movie adaptation recommendations. To test this hypothesis, we build the neural network, which should propose character distribution in the movie script based on the book text.

To build this model, we use the dataset of 120 book-script pairs. We prepare input data by processing it using NER to find the top 10 characters in the specific text and write it into the corresponding .csv file. As an example, there are names and their occurrences in 2001: A Space Odyssey book is shown in Figure 4.8. We expect the output to be the same characters with their occurrences in the script proposed.

	Name	Occurrence
0	Bowman	183
1	Floyd	118
2	Hal	97
3	Poole	61
4	MoonWatcher	39
5	Dave	22
6	Betty	17
7	Michaels	16
8	Japetus	16
9	Halvorsen	15

FIGURE 4.8: Processed data example

We build our model using **tensorflow** framework. It is a sequence of three dense layers:

1. Dense 10
2. Dense 16
3. Dense 10

This model was trained on 10 epochs and showed no improvement in accuracy (40%). Neither the increase in layer amount nor units in layers, does not change the model efficiency. It can be explained with no mathematical dependencies in character distribution difference in script compared to the book.

Each book-to-movie adaptation is a unique work. There is no success recipe - it can be done in many different ways. And the computer cannot recommend how to distribute characters, what to include or exclude from the original story, etc.

4.3 Problems

The data collection complexity was the main problem in this study. As scripts are not usual to be posted on the internet and books are not free of charge, it took 20 minutes to find 1 pair. We got 120 adaptation sets from 600 processed pairs. It took 200 man-hours to form the database. Some of these files needed additional manual refinement: clearing unknown characters and 'watermarks', extracting a story from a collection of books, cropping the beginning and the end, etc.

The NER models used for Data Analysis work 10 minutes on average. Data preparation for ML took 80 hours of non-stop text mining. Thereafter, some errors, fixes, and changes cost a lot of time. This model has 2 main flaws: case sensitivity ('Voldemort' != 'VOLDEMORT') and false-positive error ('Hello' is marked as name).

Chapter 5

Proposed Solution: Web tool for book to movie analysis

5.1 Solution overview

The result of this work is a web tool for a story analysis with further adaptation recommendations.

The scriptwriter is able to recognize characters and locations on his/her own, he/she notices swear words, understands story mood, and can define story lines. However, he cannot precisely remember the top characters sorted or he doesn't see all story mood shifts, etc. So, the computer can make it easy for him with a visual tool approach.

There is one more persona interested in this solution. It is a geek (fandom member in other words). The value they receive from our 'product' is the ability to explore the world of stories they like so much.

Role	I want to	So that
As a scriptwriter	see available features	I can choose the one I need
As a scriptwriter	select the story to be analysed	I can get personalized analysis
As a scriptwriter	see highlighted entities in the story	I have them on spot of attention
As a scriptwriter	select the amount of characters for analysis	I concentrate only on important ones
As a scriptwriter	see the cross-analysis	I can compare the book with my script
As a scriptwriter	save the generated report	I can access it later offline
As a geek	have a catalog of stories	I don't have to search for the one by myself
As a geek	select book-movie pair from catalog	I can explore my favorite story

TABLE 5.1: General requirements list

The table above illustrates the basic set of requirements of a scriptwriter and a geek for the MVP(Minimum Viable Product) of our tool. It covers alone book or script analysis and cross-analysis of the two.

To meet specified requirements we designed the following flow:

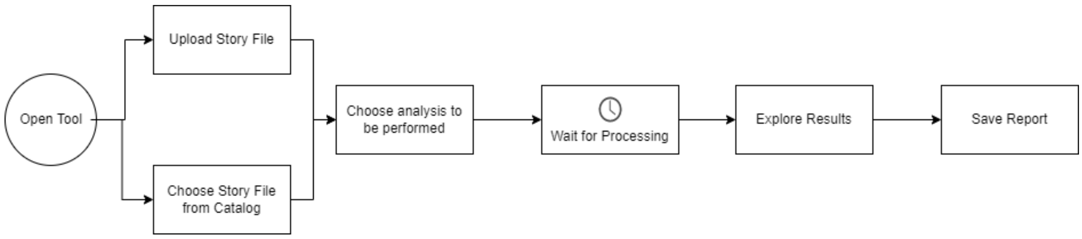

FIGURE 5.1: User flow

1. Open the Tool User lands on the home page, where he/she can see the fields for files uploading and the catalog of book-movie pairs stored in the database.

2. Upload/Select file(s)

   Users can upload the book and/or film script. There are two fields for uploading: one for books and one for the movie. At least one of them must be filled to enable uploading.

   Or users can select the story from the prepared catalog and then choose between book, film, or both of them to upload for analysis.

3. Choose what report will include

   There are 5 options for analysis:

   - Word Summary
     Base analysis of word counted, story volume, and most used words in the text.
   - Characters
     Interactive charts of Characters' Distribution and their presence in the story. Users can perform cross-analysis for a specific character to compare the difference of presence in the storyline between the movie and book. It is achieved by selecting one character from the top personas rating. Users can define how many characters are included in the top list (up to 10).
   - Locations
     Charts, which show the main locations and how they change through the story, so you can better understand the characters journey.
   - Profanity
     Users can highlight swear words in text or censor them. He/she can estimate how much profanity is used in the text.
   - Sentiment
     On these charts, users can find peak moments such as culmination, happy or sad moments, and understand the average mood of the story.

   Most of these charts are illustrated in the Methodology Chapter. So users can choose what options are interesting to him/her and start an analysis.

4. Wait for Processing

   It takes some time to analyze text, so the user must wait a few minutes before the results are shown.

5. Explore Results

   Now users can see all charts and data that he was interested in and explore them. More specifically, analyze the movie and/or script or compare them (if both were uploaded).

6. Save Report

   To avoid repeated analysis users can save generated reports and have access to it offline.

5.2 Technology stack

There is one serious limitation of this solution - analysis speed. The average time to run a full analysis from the previous chapter is more than 1 hour. But this problem can be avoided by splitting the tool into Front-end and Back-end.

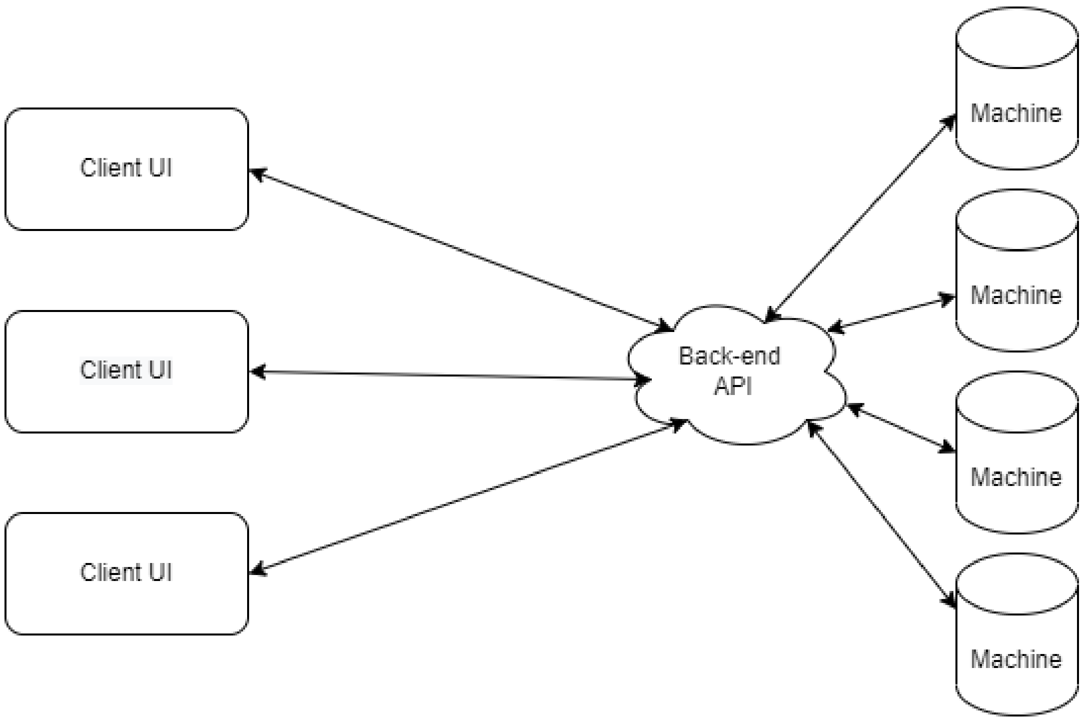

FIGURE 5.2: Technical architecture

- Front-end

  Under front-end, we mean the application, which can form requests for story analysis. To complete this task user have to select a file with a story and choose what he wants to see in the report. So we need a simple app with minimum logic in it.

  A good choice of technology for implementing such a task is **Flutter**. It is a framework for Dart language, created by Google. Flutter provides tools for creating cross-platform applications (Android, Linux, Web, etc.). After implementing Front-end we need to deploy it, so anyone can access it. If we talk about web apps, we need a website to place them. In native apps case

(Android, iOS) application file is deployed to the corresponding market (App Store, Play Market).

We implemented a web application and used **GCP**, namely **App Engine**, to handle its deployment, hosting, and support. This is a service, which creates the **VM** (Virtual Machine), configures the application there, and helps to handle a domain name.

- Back-end

  Python is used for Back-end implementation. It has a large variety of libraries for developing APIs, working with GCP, and handling async operations. It can be divided into 3 parts:

  1. API
     It is a facade for Front-end, which accepts requests from a website. API saves the file for the analysis in Storage and delegates tasks to Computing instances.
     Redis tracks the progress and notifies API when the task is done. When the processing is finished, API collects them, forms the final report, and sends it back to the user. It is also powered by App Engine.
  2. Storage
     Cloud Storage from GCP contains reports and files for analysis.
  3. Computing instances
     This is the most important part of the Back-end. It runs all analysis and forms result for tasks, which were given by API. As tasks do not have any dependencies, they can be processed in parallel on different machines for speed improvement. If it is not enough, text can be decoupled into many chapters and distributed between machines, so each instance is responsible for completing exactly one task.
     When a task is received, the machine gets data from Storage and informs API through Redis about its progress.
     For requirements specified above, we use **Google Compute Engine**, which allows users to launch virtual machines on demand. As we are using Neural Network models for text processing, Virtual Machines with GPU are the best choice, because video cards are known for better performance.

As a result, we have a highly scalable solution, which is easy to deploy, update and monitor its 'health'. By monitoring statistics we can develop further functionality and improve performance.

5.3 Screens

In this section the following web tool screens are presented:

1. Landing page (5.3)
2. Upload page (5.4)
3. Catalog page (5.5)
4. Preview page (5.6)

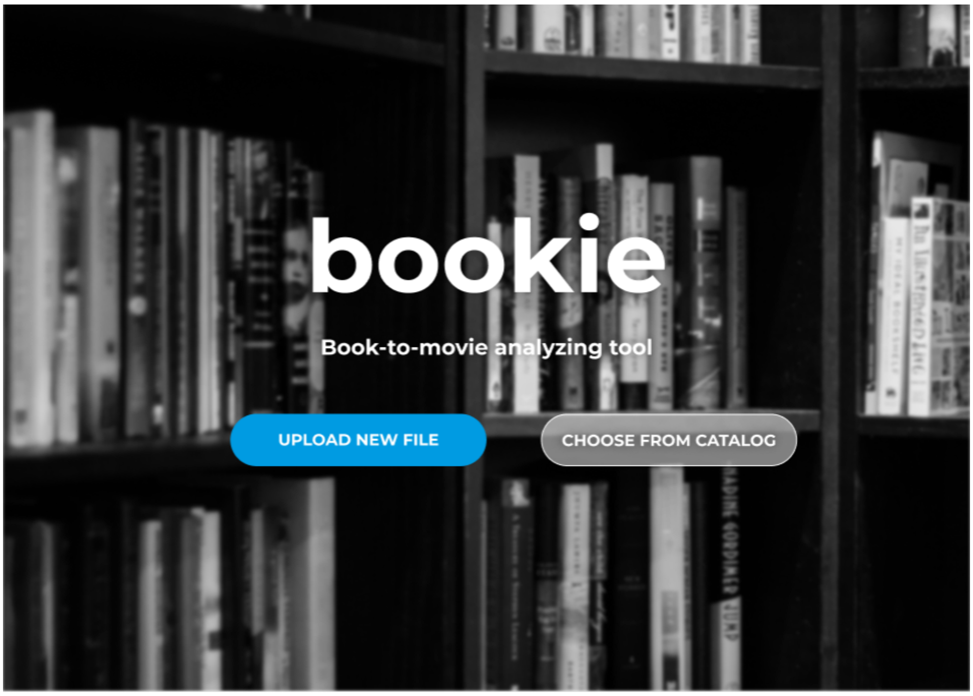

FIGURE 5.3: Landing page

FIGURE 5.4: Upload page

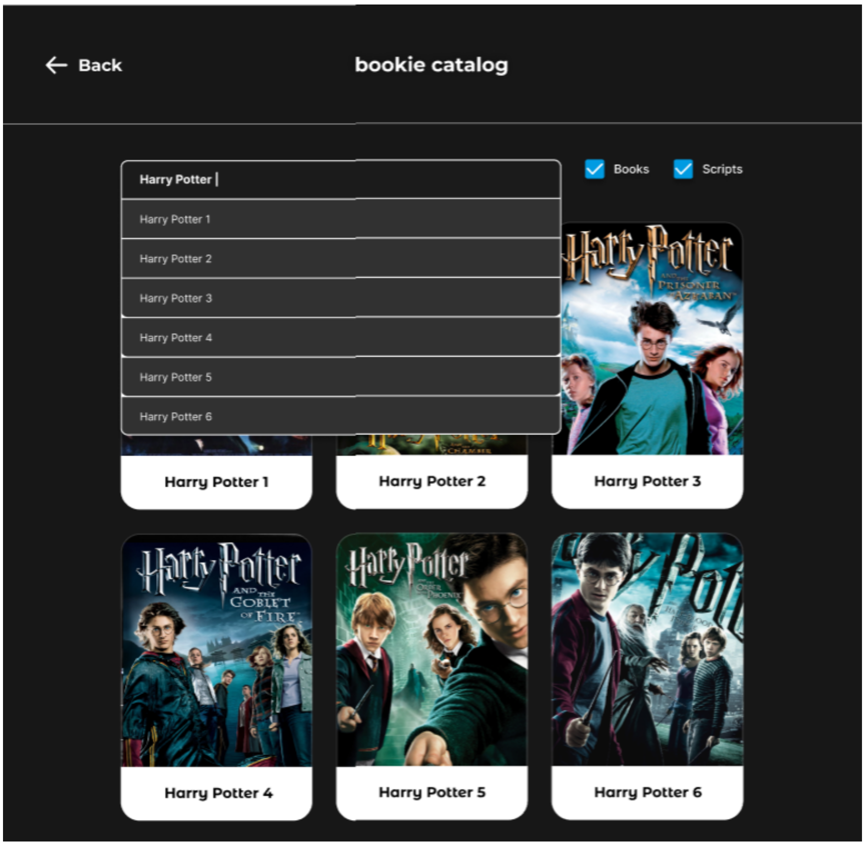

FIGURE 5.5: Catalog page

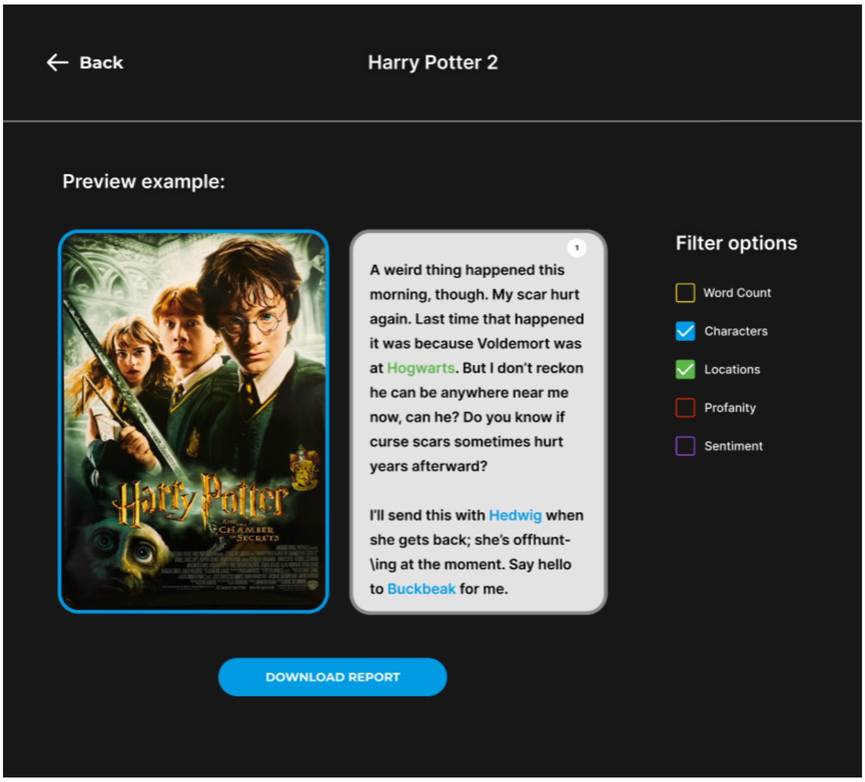

FIGURE 5.6: Preview page

5.4 Further improvements

Now we have MVP, which can be greatly improved:

- Replace NER models with our own ones, that are more specific for our needs. In detail, one model is oriented only on book text analysis, and one only on the script.
- Implement Android, iOS, macOS, Linux, Windows native applications
- Add database to Back-end for caching analyzed texts and reusing the results for future requests
- Add book/movie genre classification
- Add character relationships diagram
- Combine existing features: Character & Location, Character & Sentiment, Location & Sentiment, Character & Profanity, etc.
- Increase Catalog data volume. Give users the possibility to add pairs to the Catalog.

Chapter 6

Conclusions

6.1 Summary

The main idea of this work was to investigate the application of ML in book-to-movie adaption and visual approach development.

The hypothesis that Neural Networks can be used for scriptwriting recommendations is incorrect. For now, they cannot generate meaningful dialogues, identify storylines, tell what to cut off the story and what to put the accent on. It remains to be the art for humans.

However, it is a powerful instrument for story analysis. We used Neural Networks in Entity Recognition and Sentiment Analysis tasks. They are not one hundred percent accurate, but it is not major in terms of big text analysis.

For the best results representation, we built a web tool, which can visualize analysis on interactive charts, and produce a document with insights highlighted.

The idea for future direction is to include a movie video file as a third element of the book-to-movie adaptation analysis and reviews. There are several things we can work on in the future:

- Actors to play the role of the top character based on IMDb historical data
- Screen time recommendation based on viewers' feedback
- Scene composition according to the description

6.2 Lessons Learned

After conducting this study, I understood that ML is not as powerful as I thought it to be. For the present, it cannot replace people in the story creation process. But, it can be useful in analyzing existing texts. I learned the basis of Neural Networks modeling, improved result presentation skills, built a system architecture, and practiced web development.

I have to mention that it was a real pleasure for me to look into my favorite stories.

Bibliography

[1] Philippe Aurier and Guergana Guintcheva. *From book to movie: an investigation of adaptation and its impact on spectators' evaluation judgment*. 2014. URL: https://www.afm-marketing.org/en/system/files/publications/20160516143721_AURIER_GUINTCHEVA.pdf.

[2] Linda Hutcheon. *A Theory of Adaptation*. Routledge, 2006, pp. 120–122.

[3] Listopia. *Books Made into Movies*. URL: https://www.goodreads.com/list/show/252.Books_Made_into_Movies.

[4] Annalee Newitz. *Movie written by algorithm turns out to be hilarious and intense*. 2021. URL: https://arstechnica.com/gaming/2021/05/an-ai-wrote-this-movie-and-its-strangely-moving/.

[5] Frontier Economics publishers. "Publishing's contribution to the wider creative industries". In: *Frontier Economics* (2018), pp. 3–11. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(94)90419-7. URL: https://www.sciencedirect.com/science/article/pii/0377221794904197.