

**Department of Electrical & Computer Engineering**

An Analysis of Character Representation from

Print (Book) vs Visual (Movie) Media

**CSE499B (Senior Design Project)**

Name: M Saydur Rahman ID: 1520490042  
 Name: Syed Md. M. A. Ashraf Nawaz ID: 1620762042  
 Name: Md. Tasnimul Hasan ID: 1520790042

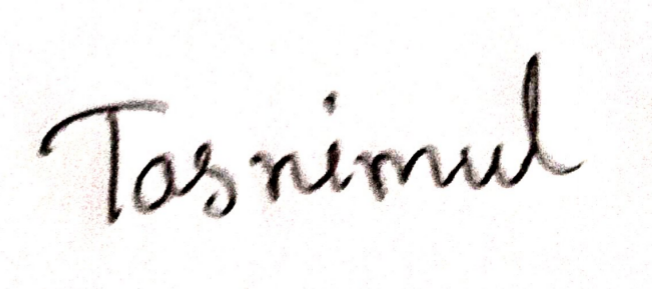
Name: Md. Fazly Rabby Chowdhury ID: 1621519042

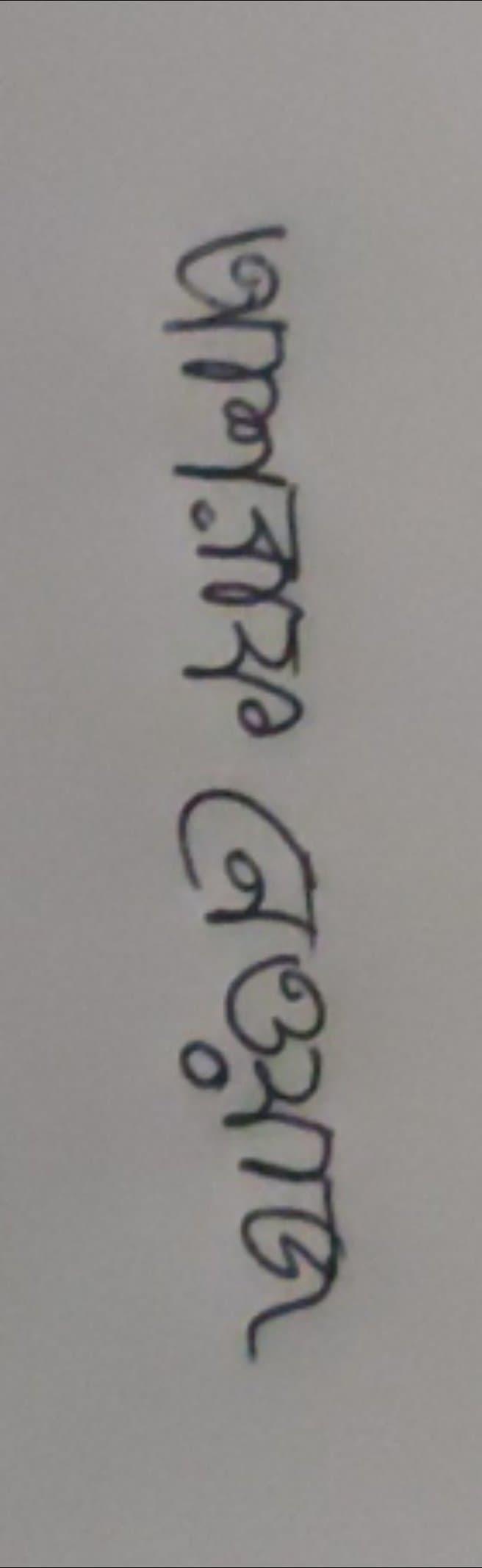
**Research Supervisor:   
  
Mohammad Ashrafuzzaman Khan**

Assistant Professor, ECE Department   
 North South University

**DECLARATION**

# This is to declare that this research report nor the project has been submitted anywhere for any degree or program. Throughout writing this report, help from different sources has been taken, and they are provided with proper acknowledgment in the reference section of the report. *Declaration given by:* ……………………….. M Saydur Rahman ID: 1520490042

****………………………..   
Md. Tasnimul Hasan  
ID: 1520790042

****………………………..   
Syed Md. M. A. Ashraf Nawaz  
ID: 1620762042

****………………………..   
Md. Fazly Rabby Chowdhury  
ID: 1621519042

**APPROVAL**  
  
This Research Report is prepared to fulfill the requirements of our CSE499B (Senior Design Project) and we ( M Saydur Rahman, Syed Md. M. A. Ashraf Nawaz, Md. Tasnimul Hasan, Md. Fazly Rabby Chowdhury) are glad and blessed to complete this project within the given time and submitting the report. We conducted this research with sheer encouragement and support from our faculty supervisor. We believe that with him beside us, the accomplishment of this project seemed much easier.   
  
The sole purpose of this report is for the course CSE499B (Senior Design Project), which requires multidisciplinary students working together with the Electrical and Computer Engineering Department of North South University to complete research.   
  
Hence, this is our earnest request to the respected Chairman, Department of Electrical and Computer Engineering to accept this report as the fulfillment of the degree of Bachelor of Science in Computer Science and Engineering under Electrical and Computer Engineering Department of North-South University.

***Approval given by:***

**………………………………. ………………………………**Research Supervisor Dr. Mohammad Rezaul Bari

Dr. Mohammad Ashrafuzzaman Khan Chairman, ECE Department

Assistant Professor, ECE Department North South University  
North South University Dhaka, Bangladesh   
Dhaka, Bangladesh  
  
  
  
  
**Abstract**

Our project is about An Analysis of Character Representation from Print(Book) vs Visual(Film) Media. Process those frames in any type of model or pre-trained model to count the characters. Another important part is NLP (Text Analysis). With many books getting adapted as movies.AI provides a useful tool that assists video analysis. In this project, we formulate a process to determine the importance of any character in movies with their book counterparts. We propose some model classifiers to solve the video part, a search model to analyze book representation, and statistical analysis for comparison. How much a movie adaptation of a book gives characters true representation! We are trying to find the answer to that question via automation via the use of neural networks. We are going to analyze the video via our neural network while trying to collect statistical data. The text analysis will provide us data on a character representation from the book. We plan on using statistical analysis to find out the representation true to the book author's intention.

Keywords: Character Count · Video Analysis · Text Analysis · Statistical analysis.

# Table of Contents

Table of Contents

[Abstract](#_30j0zll) 4

[Table of Contents](#_1fob9te) 5

1 Introduction 7

[2 Background](#_3dy6vkm) 8

2.1 Bert 8

2.2 Spacy 8

2.3 GPT2 9 2.4 Stanford NLP Library10

2.5 Python Imaging Library11

[3](#_3rdcrjn) Algorithms and Models 12

[4 Results and discussion](#_26in1rg) 14

4.1 Google Web Server14

4.1.1 Google Colab 15

4.2 Jupyter Notebook15

4.3 Pycharm 16

[5 Methodology](#_lnxbz9) 16

5.1 Video Analysis17

5.2 Text Analysis17

5.3 Statistical Analysis 17

6Results and discussion 18

6.1 Outputs19

6.1.1 Print Media Text Analysis Output 19

6.1.2 Visual Media Output 20

7 Conclusions and recommendations 21

8 [Acknowledgements](#_35nkun2) 21

9 [References](#_1ksv4uv) 22

10 Appendix 25

# 

# 

# 

# 

# 

# 

# 1 Introduction

Here are some Important Ways Storytelling Is Different in Books vs. Movies. The same story might be different in books and movies or not different! Book Will “Tell” More than a Movie! Book Will Have Less Subtext Than a Movie or less character representation in movie or book! So here we found some differences. We had to make some model solutions. We searched for some approaches. Our first approach was character count in a movie. The screen time of an actor in a movie or an episode is very important. We want to know how much time our favorite character acted on screen. So, have you ever wondered how we can calculate the total screen time of an actor? The answer is with deep learning. The idea is nothing but videos are nothing but a collection of a set of images. These images are called frames and can be combined to get the original video. So, a problem related to video data is not that different from an image classification or an object detection problem. How much a movie adaptation of a book gives characters true representation! We are trying to find the answer to that question via automation via the use of neural networks. We are going to analyze the video via our neural network while trying to collect statistical data. The text analysis will provide us data on a character representation from the book. We plan on using statistical analysis to find out the representation true to the book author's intention.

**2 Background**

In our thesis, we worked on two media analysis. One is print media text analysis and another one is visual media analysis. For our print media text analysis, we tried 4Natural language processing (NLP) models. Those are - Bert, Spacy, GPT2, and Stanford NLP Library. We found our desired output on the Spacy model. Also, the Bert model gave us a better output than the GPT2 and Stanford NLP Library. For visual media analysis, we can train a model ourselves and also can use a pre-trained model. First, we approach training a model for visual analysis. We found that training a model is very time consuming and needs much more resources than a pre-trained model. So, then we moved our focus to use a pre-trained model. And for our visual media analysis, we use the PIL Library and Face Recognition Module. For this library and module, we need not train the model ourselves.

# 2.1 Bert

One of the biggest challenges in [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) (NLP) is the shortage of training data. Because NLP is a diversified field with many distinct tasks, most task-specific datasets contain only a few thousand or a few hundred thousand human-labeled training examples. However, modern deep learning-based NLP models see benefits from much larger amounts of data, improving when trained on millions, or billions, of annotated training examples. To help close this gap in data, researchers have developed a variety of techniques for training general-purpose language representation models using the enormous amount of unannotated text on the web (known as pre-training). The pre-trained model can then be fine-tuned on small-data NLP tasks like [question answering](https://en.wikipedia.org/wiki/Question_answering) and [sentiment analysis](https://en.wikipedia.org/wiki/Sentiment_analysis), resulting in substantial accuracy improvements compared to training on these datasets from scratch. [23]

The BERT model was proposed in BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. It’s a bidirectional transformer pre-trained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia.

They introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement). [24]

* BERT is a model with absolute position embeddings so it’s usually advised to pad the inputs on the right rather than the left.
* BERT was trained with the masked language modeling (MLM) and next sentence prediction (NSP) objectives. It is efficient at predicting masked tokens and at NLU in general but is not optimal for text generation.

# 2.2 Spacy

Spacy has its own deep learning library called think used under the hood for different NLP models. for most (if not all) tasks, spaCy uses a deep neural network based on CNN with a few tweaks. Specifically for Named Entity Recognition, spaCy uses. A transition-based approach borrowed from shift-reduce parsers, which is described in the paper Neural Architectures for Named Entity Recognition by.

A framework that's called "Embed. Encode. Attend. Predict"

Embed: Words are embedded using a Bloom filter, which means that word hashes are kept as keys in the embedding dictionary, instead of the word itself. This maintains a more compact embeddings dictionary, with words potentially colliding and ending up with the same vector representations.

Encode: A list of words is encoded into a sentencing matrix, to take context into account. spaCy uses CNN for encoding.

Attend: Decide which parts are more informative given a query, and get problem-specific representations.

Predict: spaCy uses a multi-layer perceptron for inference. Advantages of this framework, per Honnibal are:

i)Mostly equivalent to sequence tagging

ii)Shares code with the parser

iii)Easily excludes invalid sequences

iv)Arbitrary features are easily defined

**2.3 GPT2**

For OpenAI GPT-2 model was proposed in [Language Models are Unsupervised Multitask Learners](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf) by Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever. It’s a causal (unidirectional) transformer pre-trained using language modeling on a very large corpus of 40 GB of text data.

GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages. GPT-2 is trained with a simple objective: predict the next word, given all of the previous words within some text. The diversity of the dataset causes this simple goal to contain naturally occurring demonstrations of many tasks across diverse domains. GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data. [26]

* GPT-2 is a model with absolute position embeddings so it’s usually advised to pad the inputs on the right rather than the left.
* GPT-2 was trained with a causal language modeling (CLM) objective and is therefore powerful at predicting the next token in a sequence. Leveraging this feature allows GPT-2 to generate syntactically coherent text as it can be observed in the run\_generation.py example script.
* The PyTorch models can take the past as input, which is the previously computed key/value attention pairs. Using this past value prevents the model from re-computing pre-computed values in the context of text generation. See [reusing the past in generative models](https://huggingface.co/transformers/quickstart.html#using-the-past) for more information on the usage of this argument.

**2.4 Stanford NLP Library**

Stanford NLP is a Python natural language analysis package. It contains tools, which can be used in a pipeline, to convert a string containing human language text into lists of sentences and words, to generate base forms of those words, their parts of speech, and morphological features, and to give a syntactic structure dependency parse, which is designed to be parallel among more than 70 languages, using the [Universal Dependencies formalism](https://universaldependencies.org/). In addition, it is able to call the CoreNLP Java package and inherits additional functionality from there, such as constituency parsing, coreference resolution, and linguistic pattern matching.

This package is built with highly accurate neural network components that enable efficient training and evaluation with your own annotated data. The modules are built on top of [PyTorch](https://pytorch.org/). You will get much faster performance if you run this system on a GPU-enabled machine. This package is a combination of software based on the Stanford entry in the [CoNLL 2018 Shared Task on Universal Dependency Parsing](http://universaldependencies.org/conll18/), and the group’s official Python interface to the Java [Stanford CoreNLP software](https://stanfordnlp.github.io/CoreNLP). The CoNLL UD system is partly a cleaned-up version of code used in the shared task and partly an approximate rewrite in PyTorch of the [original Tensorflow version](https://github.com/tdozat/Parser-v3) of the tagger and parser. [25]

Stanford NLP features:

* Native Python implementation requiring minimal efforts to set up;
* Full neural network pipeline for robust text analytics, including tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) and morphological features tagging and dependency parsing;
* Pretrained neural models supporting [53 (human) languages featured in 73 treebanks](https://stanfordnlp.github.io/stanfordnlp/models.html#human-languages-supported-by-stanfordnlp);
* A stable officially maintained Python interface to CoreNLP.

# 2.5 Python Imaging Library

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool[22].

Pillow offers several standard procedures for image manipulation. These include[21]:

I)Per-pixel manipulations,

ii)Masking and transparency handling,

iii)Image filtering, such as blurring, contouring, smoothing, or edge finding,

iv)Image enhancing, such as sharpening, adjusting brightness, contrast or color,

v)Adding text to images and much more].

# 3 Algorithms and Models An algorithm is a sequence of computational steps that transforms the input into the output. In our opinion, the algorithm makes solvable problems run faster and make hard problems solvable. The purpose of an algorithm is to give a set of rules by which one can solve a problem. Think of it as laying out a step-by-step guide that will accomplish a specific task or solve a specific calculation if the steps are followed in order. An important aspect to think about when putting together an algorithm is whether or not it’s generic. That is, will these steps produce the correct result even if we apply them to tasks that differ slightly in their preconditions? To illustrate the importance, imagine that we would like to solve some task involving a vector. We start off laying out an algorithm accomplishing this task in a two-dimensional vector space because that’s what we need right at this very moment. The important difference between generic and non-generic algorithms comes into play when we move on and want to accomplish the same task in n-dimensional vector space. A generic algorithm will still provide the correct result, while a non-generic algorithm will get a wrong result or won’t even make sense when faced with these changed conditions. In most simplistic terms. BERT stands for Bidirectional Encoder Representation from Transformers. It is a language representation model and basically a trained transformers encoder stack. BERT applies bidirectional training of transformer encoders using Auto-encoder language modeling. It performs two unsupervised learning tasks during pertaining, one is masked language modeling and the other one is next sentence prediction tasks. It finally achieves state-of-the-art results on GLUE, MNLI, SQuAD data-sets. If we look at the architecture of BERT. It consists of 12 stacked encoder units in the base model and 24 stacked encoder units in the large model. Each encoder unit has a similar architecture to the Transformer model which is an attention-based model having encoder-decoder architecture. Coming to the architecture of encoders it consists of two layers one is a multi-headed attention layer and another one is a position-wise fully connected feed-forward neural network. The attention layer calculates attention on the input embedding using this formula which is then added, normalized, and passed to the neural network. The neural network applies two Linear transformations with a ReLU activation in between. The output is then normalized and passed to the next encoder unit. The output of the final encoder layer is then passed to a soft-max classifier to output the probabilities of different classes and to predict the final outcome. Spacy has its own deep learning library called think used under the hood for different NLP models. For most tasks, spaCy uses a deep neural network based on CNN with a few tweaks. Specifically for Named Entity Recognition, spaCy uses. The Python Imaging Library or PIL allowed you to do image processing in Python. The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool[22]. OpenCV is capable of image analysis and processing. This means it is great at taking frames out of the video or taking in two frames from a stereoscopic camera and running algorithms to extract information. For example, using OpenCV would give us the mathematical tools required to capture images and track a particular object as it moves around. This is not provided directly but the mathematical tools required to process the images to extract such information are available. We can see from the above example that although it can do other things such as stretch an image or change color, it's purpose is not to serve as an image processing engine similar to Photoshop or such. It is intended to be very fast almost real-time if the hardware supports it and performs all sorts of functions such as Fourier transforms very fast and then allows us to either glean information or transform the stream of images as we like. So the answer really is that its application is infinite in the domain of video/image analysis and processing including things such as facial recognition, tracking objects, determining the distance of objects from a stereoscopic input, etc. None of these are given ready-made, but the mathematical tools are provided which we need to know how to apply.

# 4 Available Platforms and Tools The process of our work requires multiple analytical tools and powerful computing platforms such as Google Colab, Jupyter Notebook, etc. Tools that are required have been explained in a previous section. Though both platforms have some positive and negative attributes. But we have found Google Colab a better platform for our work.

## 

## 4.1 Google Web Server

It does matter that google has a really large number of servers to serve requests from multiple locations. Plus google applications are coded from the get-go, to make good use of tools like Content Delivery Networks (CDNs) which reduce the amount of time it takes for google services to send static content to your devices from a server that is geographically closer to you. So essentially, they divide the website load onto more than one machine of which one only deals with processing and generating the page that is to be sent to you, whereas the other machine. The only job is to serve the static content to you as fast as possible. Smart geographic deployment of servers and smart caching of data on those servers. Using very well optimized hardware and software to perform just the tasks those machines are expected to do and we have to admit having the infrastructure to deliver that kind of load does matter. Only when you work these factors as a whole do you get the kind of performance we have come to expect from google services. Just buying the biggest server and hosting it at the biggest data center is no longer enough.

### 4.1.1 Google Colab

Google Collab has a prominent role in the field of machine learning and deep learning. An analyst who is not a part of college or universities, to buy GPU power is very costly. In maximum universities and colleges have GPU power but the college which has not good financial. They can not afford the GPU power. For that college, Google Collab is an excellent platform to run the ML or DL model without any cost. Google Collab provides 12 hours of free access to K-80 GPU power. Google Colab runs on Google Drive. The best part of that is, there is no need to install any ML or DL library. Without installing, we can use it very easily. In the future, Google must increase the computing power of Google Colab, which is a very positive message for Deep learning and Machine learning enthusiasts.  
  
Pros: Faster computation than your typical 8–16 GB RAM.  
Cons: It doesn’t save our code automatically, so if the internet is gone, power is off, we closed the lid of your notebook, the code is gone. So hit save after every line you write. Too bad if you’ve slow internet.

## 4.2 Jupyter Notebook

First, we have to understand the purpose of notebooks or notebook documents. These are documents in which we bring together code and rich text elements (figures, links, equations, …). In the case of Jupyter Notebook, these notebook documents can be produced by the Jupyter Notebook App. The use cases in which we need to join our code and rich text elements in one document are quite varied: we might be working on a data science project, or building our data science portfolio, or we’re making a dashboard or an ETL flow, … we might also use Jupyter Notebook for research purposes. On a bigger scale, data science teams can use Jupyter Notebooks to collaborate on projects. These use cases explain why we saw a link to the Jupyter project on a Pandas page. As a data manipulation and analysis package for Python, Pandas can really be used in most (if not all) use cases that are described above. Nevertheless, we can also see why Pandas isn’t the only package that can come in handy in a data science project or a dashboard. According to our use case, we might resort to, for example, the other typical Python data science libraries that are out there. I would consider all that I have said before the basic use of the Jupyter Notebooks. we can do much more, as Jupyter Notebook also allows us to run many other languages, such as R, Julia, Scala, SAS, … we can easily make a notebook document that contains code that is another programming language than Python and we can interactively switch between languages in our notebook documents. We think this briefly covers what Jupyter Notebook is and why it can come in handy.

## 4.3 Pycharm

PyCharm has been the best-utilized IDE (Integrated Development Environment) for the most commonly used general-purpose programming languages of today which is python. Python is a very versatile language and many programmers nowadays opt for python to build software applications so they need the code to be concise, clean, and readable as well, They can even accelerate custom software application development by actually taking advantage of the number of IDE's support for python. Apart from that, Python language is also widely used.

# 5 Methodology

There are three distinct process combinations for the analyzing system to work. Three distinct problems will be solved differently:

1. Video Analysis
2. Text Analysis
3. Statistical Analysis

**5.1 Video Analysis:**  To extract representation information from the video we divide videos into frames, the number of frames needs to be low to reduce workload while greater accuracy requires higher frame rates. So standard needs to be set deepening on the capacity of processing power available at hand. Frames will go through a model trained with characters names and images which will count those characters and provide data on character representation.

**5.2 Text Analysis:**

We used spacy for text analysis, which provides the number of times characters are mentioned (noun or pronoun). Noun mention and count individually is quite easy to gather. Pronouns are rather challenging. While pronouns that are adjacent can be found out using spacy pronouns that refer to nouns not directly like either A or B, A nor B leading to a pronoun is not readily detected. Text analysis provides us numerical data to make a statistical comparison.

**5.3 Statistical analysis:**

Data gathered from both visual and print media is compared which provides us the representation differences. This shows the result of the question is the representation in cross-platform content true to each other.

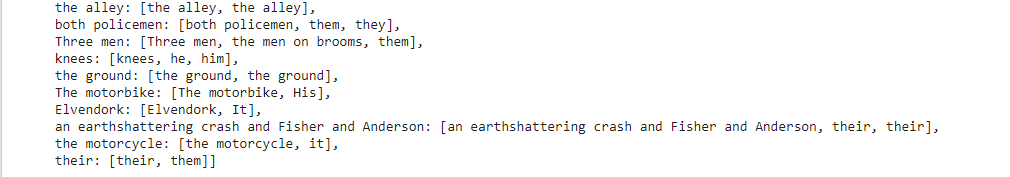
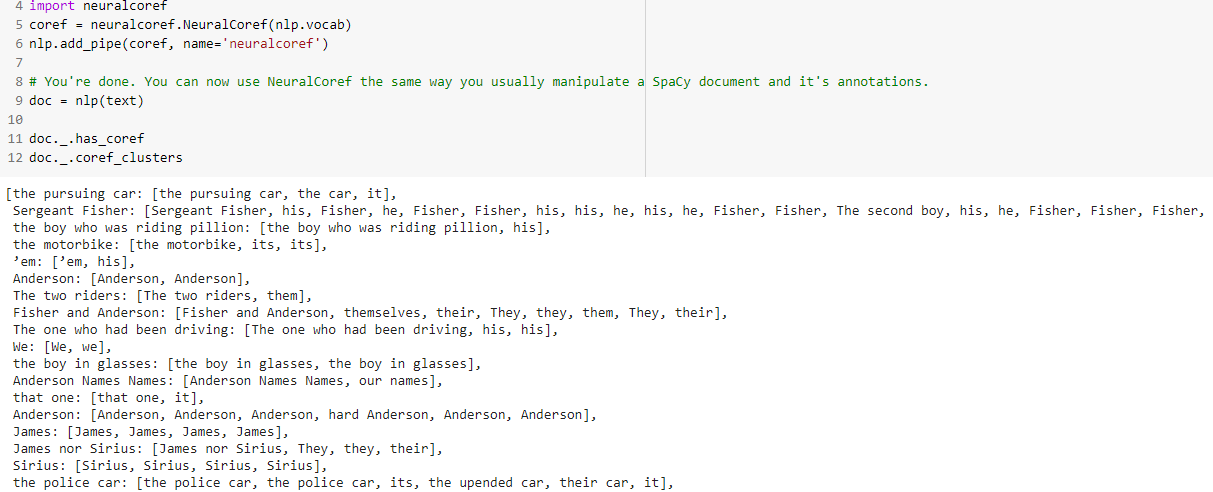
# 6 Results and discussion

If the question is if cross-platform representation is the same the answer is no. There are differences. But with the use of AI things like that can be easily detected. This system is not reclusive to print and visual media. Audiobooks comics every type of media content can be analyzed to some extent by the use of AI. There are some difficulties in this process, going through a large number of frames requires very high processing power. For text analysis, we need a relevant part of the story with the video. Text analysis lacks the capacity to recognize references in some cases. Colab has its limitations. For visual media analysis, we faced problems choosing appropriate models between supervised and unsupervised for our work. We faced problems on visual media to recognize characters correctly and count the respected character frame. For print media analysis, we got confused about which model is appropriate for our work. Also, we faced problems identifying the pronoun coreference of the related proper noun.

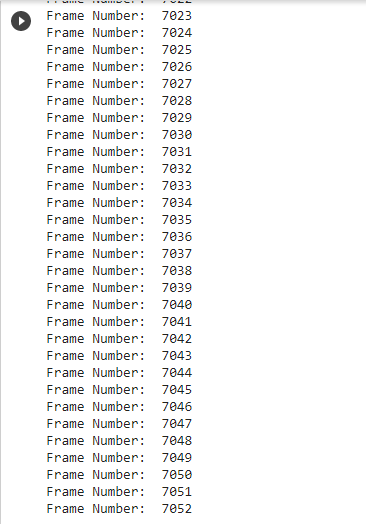
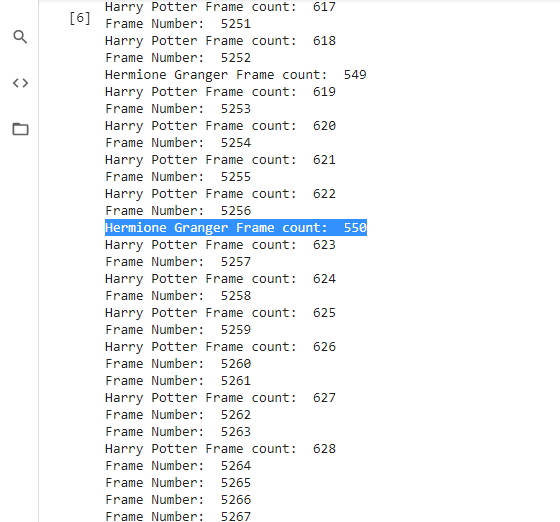
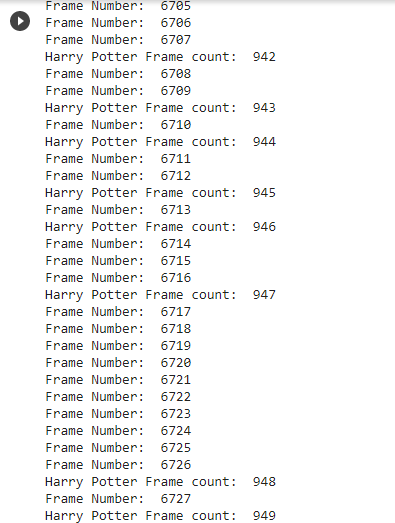
**6.1 Outputs**

There is some output of our print media text analysis and visual media analysis below:

**6.1.1 Print Media Text Analysis Output**

****

**6.1.2 Visual Media Output**

****

# 7 Conclusions and recommendations

The movie vs book representation analyzer can be a bais free way of rating media contents. In the future, the system can be improved to analyze multi-part books to film adaptations with improvement even tv series can also be analyzed.

The movie analysis can be improved using character time on screen (neural network based facial recognition) with improved accuracy but at a higher workload.

The book representation can be improved with reference recognition (how many times a character comes up in conversations).

Adding audio-based character recognition can also be accurate in finding out character representation in movies and books in the future to improve.

# 8 Acknowledgements

This project is under our combined academic course CSE 499 A and B, titled “Senior Design Project I and II” and we conducted this work under the supervision of Mohammad Ashrafuzzaman Khan. We are very grateful to him for giving us a great opportunity to work on such a challenging and new topic. We would like to express our very great appreciation to Dr. Mohammad Ashrafuzzaman Khan for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated.

# 9 References: [1] Video Summarization Using Fully ConvolutionalSequence Networks. <https://openaccess.thecvf.com/content_ECCV_2018/papers/Mrigank_Rochan_Video_Summarization_Using_ECCV_2018_paper.pdf>

[2] Learning video summarization from unpaired data.  
<https://openaccess.thecvf.com/content_CVPR_2019/papers/Rochan_Video_Summarization_by_Learning_From_Unpaired_Data_CVPR_2019_paper.pdf>

[3] Cisco visual networking index: Forecast and methodology, 2016-2021.   
<https://www.cisco.com/>.

[4] Open video project.<https://open-video.org/>

[5] DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.<https://ieeexplore.ieee.org/document/7913730>.

[6] An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.<https://arxiv.org/abs/1803.01271>.

[7] Diverse Sequential Subset Selection for Supervised Video Summarization.  
<https://papers.nips.cc/paper/5413-diverse-sequential-subset-selection-for-supervised-video-summarization>.

[8]Statistical Methods for Comparison of Data Sets of Construction Methods and Building Evaluation <https://www.sciencedirect.com/science/article/pii/S1877042813029686/pdf?md5=2ee7f259710d2a331e4a54e73e979d69&pid=1-s2.0-S1877042813029686-main.pdf>

[9]A character-based convolutional neural network for language-agnostic Twitter sentiment analysis  
<https://www.researchgate.net/publication/318329681_A_character-based_convolutional_neural_network_for_language-agnostic_Twitter_sentiment_analysis/>  
  
[10] Convolutional Neural Networks for Sentence Classification  
<https://www.aclweb.org/anthology/D14-1181.pdf>  
  
[11 Text Understanding from Scratch  
<https://arxiv.org/pdf/1502.01710.pdf>  
  
[12] Character-level Convolutional Networks for Text Classification  
<https://arxiv.org/pdf/1509.01626.pdf>  
  
[13] Recurrent Convolutional Neural Networks for Text Classification  
<https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745/9552>  
[14] Text Analysis <https://monkeylearn.com/text-analysis/>  
  
[15] An Improved Algorithm for Video Summarization – A Rank Based Approach.<https://www.researchgate.net/publication/306362760_An_Improved_Algorithm_for_Video_Summarization_-_A_Rank_Based_Approach>.  
  
[16] Face detection techniques a review

<https://www.researchgate.net/publication/326667118_Face_Detection_Techniques_A_Review>  
  
[17] Video summarization using textual descriptions for authoring video blogs

<https://link.springer.com/article/10.1007/s11042-016-4061-3>.  
  
[18] Jitao Sang, Changsheng Xu. Character-based movie summarization  
<https://dl.acm.org/doi/abs/10.1145/1873951.1874096?download=true>.

# [19] Yingbo Li, Bernard Merialdo, Mickael Rouvier, Georges Linares. Static and Dynamic Video Summaries <http://www.eurecom.fr/en/publication/3438/download/mm-publi-3438_1.pdf>. [20] Important Ways Storytelling Is Different in Books vs. Movies <https://www.helpingwritersbecomeauthors.com/5-important-ways-storytelling-different-books-vs-movies/>

<https://en.wikipedia.org/wiki/Python_Imaging_Library>

[22]The Python Imaging Library adds image processing capabilities to your Python interpreter.This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool

<https://pypi.org/project/Pillow/>

[23] [Google AI Blog: Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing (googleblog.com)](https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html)

[24] [BERT — transformers 4.2.0 documentation (huggingface.co)](https://huggingface.co/transformers/model_doc/bert.html)

[25] [StanfordNLP 0.2.0 - Python NLP Library for Many Human Languages | StanfordNLP](https://stanfordnlp.github.io/stanfordnlp/index.html)

[26] [OpenAI GPT2 — transformers 4.2.0 documentation (huggingface.co)](https://huggingface.co/transformers/model_doc/gpt2.html#overview)

# 

**10 Appendix:**

**Github Link:** [MSaydurRahman/An-Analysis-of-Character-Representation-from-Print-Book-vs-Visual-Movie-Media (github.com)](https://github.com/MSaydurRahman/An-Analysis-of-Character-Representation-from-Print-Book-vs-Visual-Movie-Media)

**Code for Print Media Text Analysis:**

import spacy

from spacy import displacy

nlp = spacy.load('en\_core\_web\_sm')

from google.colab import drive

drive.mount('/content/drive')

t = open('/content/drive/MyDrive/499/dialoug\_text - Copy.txt', "r")

text = t.readline()

print(text)

doc = nlp(text)

for word in doc:

print(f'{word.text:{12}} {word.pos\_:{10}} {word.tag\_:{8}} {spacy.explain(word.tag\_)}')

from collections import Counter

#nlp = spacy.load('en\_core\_web\_sm')

# all tokens that arent stop words or punctuations

Words = [token.text for token in doc if token.is\_stop != True and token.is\_punct != True]

# Proper noun tokens that arent stop words or punctuations

proper\_nouns = [token.text for token in doc if token.pos\_ == "PROPN"]

# noun tokens that arent stop words or punctuations

nouns = [token.text for token in doc if token.pos\_ == "NOUN"]

# adjective tokens that arent stop words or punctuations

adj = [token.text for token in doc if token.pos\_ == "ADJ"]

# verb tokens that arent stop words or punctuations

verb = [token.text for token in doc if token.pos\_ == "VERB"]

# pronoun tokens that arent stop words or punctuations

pronoun = [token.text for token in doc if token.pos\_ == "PRON"]

# common tokens

word\_freq = Counter(Words)

common\_words = word\_freq.most\_common()

print("Common Words:", common\_words)

# common proper noun tokens

prop\_noun\_freq = Counter(proper\_nouns)

common\_prop\_nouns = prop\_noun\_freq.most\_common()

print("Common Proper Nouns:", common\_prop\_nouns)

# common noun tokens

noun\_freq = Counter(nouns)

common\_nouns = noun\_freq.most\_common()

print("Common Nouns:", common\_nouns)

# common adj tokens

adj\_freq = Counter(adj)

common\_adj = adj\_freq.most\_common()

print("Common Adjectives:", common\_adj)

# common verb tokens

verb\_freq = Counter(verb)

common\_verb = verb\_freq.most\_common()

print("Common Verbs:", common\_verb)

# common pronoun tokens

pronoun\_freq = Counter(pronoun)

common\_pronoun = pronoun\_freq.most\_common()

print("Common Pronouns:", common\_pronoun)

pip install neuralcoref

!git clone https://github.com/huggingface/neuralcoref.git

!pip install -U spacy

!python -m spacy download en

%cd neuralcoref

!pip install -r requirements.txt

!pip install -e .

import neuralcoref

#neuralcoref.add\_to\_pipe(nlp)

print(text)

# Load your usual SpaCy model (one of SpaCy English models)

# load NeuralCoref and add it to the pipe of SpaCy's model

import neuralcoref

coref = neuralcoref.NeuralCoref(nlp.vocab)

nlp.add\_pipe(coref, name='neuralcoref')

# You're done. You can now use NeuralCoref the same way you usually manipulate a SpaCy document and it's annotations.

doc = nlp(text)

doc.\_.has\_coref

doc.\_.coref\_clusters

**Code for Visual Media Analysis:**

from google.colab import drive

drive.mount('/content/drive')

!pip install face\_recognition

!pip install pillow

import cv2 # for capturing videos

import math # for mathematical operations

import matplotlib.pyplot as plt # for plotting the images

%matplotlib inline

import pandas as pd

from keras.preprocessing import image # for preprocessing the images

import numpy as np # for mathematical operations

from keras.utils import np\_utils

from skimage.transform import resize # for resizing images

from PIL import Image, ImageDraw

from IPython.display import display

# The program we will be finding faces on the example below

pil\_im = Image.open("/content/drive/MyDrive/499/HR.jpg")

display(pil\_im)

import face\_recognition

import numpy as np

from PIL import Image, ImageDraw

from IPython.display import display

# This is an example of running face recognition on a single image

# and drawing a box around each person that was identified.

# Load a sample picture and learn how to recognize it.

hermonie\_image = face\_recognition.load\_image\_file("/content/drive/MyDrive/499/HR.jpg")

hermonie\_face\_encoding = face\_recognition.face\_encodings(hermonie\_image)[0]

# Load a second sample picture and learn how to recognize it.

harry\_image = face\_recognition.load\_image\_file("/content/drive/MyDrive/499/DR.jpeg")

harry\_face\_encoding = face\_recognition.face\_encodings(harry\_image)[0]

# Create arrays of known face encodings and their names

known\_face\_encodings = [

hermonie\_face\_encoding,

harry\_face\_encoding

]

known\_face\_names = [

"Hermione Granger",

"Harry Potter"

]

print('Learned encoding for', len(known\_face\_encodings), 'images.')

import glob

import face\_recognition

import numpy as np

from PIL import Image, ImageDraw

from IPython.display import display

count = 0

count1 = 0

count2 = 0

folder="/content/drive/MyDrive/ExtractFrame"

for filename in glob.iglob(folder + '\*\*/\*.jpg', recursive=True):

count += 1

print ("Frame Number: ",count)

# Load an image with an unknown face

unknown\_image = face\_recognition.load\_image\_file(filename)

# Find all the faces and face encodings in the unknown image

face\_locations = face\_recognition.face\_locations(unknown\_image)

face\_encodings = face\_recognition.face\_encodings(unknown\_image, face\_locations)

# Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library

# See http://pillow.readthedocs.io/ for more about PIL/Pillow

pil\_image = Image.fromarray(unknown\_image)

# Create a Pillow ImageDraw Draw instance to draw with

draw = ImageDraw.Draw(pil\_image)

# Loop through each face found in the unknown image

for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):

# See if the face is a match for the known face(s)

matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = "Unknown"

#count=1

# Or instead, use the known face with the smallest distance to the new face

face\_distances = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)

best\_match\_index = np.argmin(face\_distances)

if matches[best\_match\_index]:

name = known\_face\_names[best\_match\_index]

#while name == "Unknown" :

#count += 1

#name.count("Unknown")

if name == "Harry Potter":

count1 +=1

print("Harry Potter Frame count: ", count1)

elif name == "Hermione Granger":

count2 +=1

print("Hermione Granger Frame count: ", count2)

#else:

#print("Unknown: ")

# Draw a box around the face using the Pillow module

draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))

# Draw a label with a name below the face

text\_width, text\_height = draw.textsize(name)

draw.rectangle(((left, bottom - text\_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))

draw.text((left + 6, bottom - text\_height - 5), name, fill=(255, 255, 255, 255))

# Remove the drawing library from memory as per the Pillow docs

del draw

# Display the resulting image

#display(pil\_image)

**Extract frames:**

# Program To Read video   
   
#Extract Frames   
  
import cv2   
  
#Function to extract frames   
  
def FrameCapture(path):   
  
#Path to video file  
  
vidObj = cv2.VideoCapture(path)  
   
#Used as counter variable   
  
count = 0  
   
#checks whether frames were extracted   
  
success = 1   
  
while success:  
  
#vidObj object calls read

#function extract frames   
  
success, image = vidObj.read()  
   
#Saves the frames with frame-count   
  
cv2.imwrite("frame%d.jpg" % count, image)   
  
count += 1   
  
#Driver Code   
  
if \_\_name\_\_ == '\_\_main\_\_':   
  
#Calling the function  
  
FrameCapture("C:\\Users\\Asus\\Downloads\\Video\\Time.mp4")