To see the world, things dangerous to come to, to see behind walls, to find each other and to feel. That is the purpose of LIFE

Un grande Grazie alla mia famiglia per credere sempre in me, ai miei Amici per esserci sempre, e al Professor Hovy per la guida ed il sostegno.

A tutta la prima classe del BAI, alle persone incontrate nel tragitto e alle nuove avventure.

"So Long, and Thanks for All the Fish"

Contents

Introduction

Cinema, Movies, and TV-Shows have long been powerful mediums capable of transporting the audience to distant worlds and unfamiliar realities. This art form goes beyond mere entertainment; it serves as a mirror reflecting the complexities of human nature and society. While some follow Aristotle's belief that the plot is the most crucial aspect of narrative[1], modern theoretical dramatists and screenwriters believe that intriguing and multifaceted characters are what truly captivate an audience.[2]

Within the vast array of narratives, characters emerge as the beating heart of cinematic creations, forging a deep emotional connection with the audience. These diverse and complex characters embody the thoughts, dreams, and struggles that resonate with the collective consciousness of humanity. As a result, it becomes more appropriate to view them as **personas** *(The Protagonist, The Villain, The Love Interest, The Best Friend etc.)* rather than the conventional understanding of mere characters.

A **persona** can be defined as the combination of a character's essence *(is)*, actions *(does)*, and experiences *(undergoes)*[1]. These personas represent specific types or classes of people that may appear across various movies, yet possess similarities among themselves (e.g. both "He-Who-Must-Not-Be-Named"[3], from the Harry Potter series and Gorr from "Thor: Love and Thunder" *kill*, even if they have have different goals and perform different actions, they are both considered Villains). As the viewers dive

[1] "Dramatic action ... is not with a view to the representation of character: character comes in as subsidiary to the actions ... The Plot, then, is the first principle, and, as it were, the soul of a tragedy: Character holds the second place." Poetics I.VI (Aristotle, 335 BCE). From Bamman, 2013[1].

[2] 2"Aristotle was mistaken in his time, and our scholars are mistaken today when they accept his rulings concerning character. Character was a great factor in Aristotle's time, and no fine play ever was or ever will be written without it" (Egri, 1946, p. 94); "What the reader wants is fascinating, complex characters" (McKee, 1997, 100). From Bamman, 2013[1].

[3] Voldemort

deeper into the vast cinematic universes, some of these personas evolve into what can be defined as *stereotypes*, serving as recognizable social norms and clichéd portrayals, which help the audience in better understanding and characterizing a given character. Acquiring a deeper comprehension of these **personas** and **stereotypes** enhances the audience's overall understanding of the plot, drawing them closer to individual characters rather than to the storyline (studios tend to extrapolate characters and produce movies around the ones which the viewers love most, see "Kronk's New Groove" (2005), from "The Emperor's New Groove" (2000), or "Minions" (2015), from "Despicable Me" (2010)). Consequently, this emotional connection with characters fosters empathy, sympathy, and a shared experience between the audience and the cinematic world.

Cinema holds the power to transport audiences to diverse and captivating stories, but it is the complex and relatable **personas** they create that truly captivate viewers. Understanding these personas, whether they evolve into stereotypes or not, significantly impacts how audiences perceive and engage with the narrative, and can help filmmakers create richer and more resonant cinematic experiences for their audiences.

Objective of the thesis

The primary aim of this thesis is to exploit the potential of Natural Language Processing techniques for comprehending and analyzing character **personas** in film narratives, focusing the analysis on English texts and proposing a general framework to expand the research.

While much of the previous studies have centered on movie plots, modern screenwriters recognize the audience's preference for stories revolving around intricate characters. Consequently, it becomes crucial to extend the research scope to these characters.

To achieve this, diverse techniques, including *coreference*, *named entity recognition* and *dependency parsing*, will be used to extract characters and their information from Wikipedia movie summaries. These information will then be used to model

the character **personas** and create well-defined classes.

Next, Latent Dirichlet Allocation will be implemented to categorize similar **personas** into groups as done in *Topic Modeling.* This step will build classes of character personas, showing the defying features of certain classes or stereotypes. This will facilitate the identification of similarities between characters from diverse cinematic universes, allowing the comprehension of particular personas. Ultimately, these findings could be employed to assess the likeness of a movie based on the presence of specific persona classes or could be used, in the future, to create such characters.

This study draws inspiration from the work of David Bemman's "Learning Latent Personas of Film Characters"[1], while also exploring novel NLP techniques, a new Wikipedia dataset and proposing possible expansion to other languages and how to use movie scripts rather than just summaries.

Background

Natural Language Processing

Natural Language Processing (NLP) is the field of study that delves into the interaction between computers and human language. It represents a powerful branch of artificial intelligence that seeks to bridge the gap between human communication and machine understanding.

At its core, NLP aims to equip computers with the ability to comprehend, interpret, and generate human language similar to human beings. Through various algorithms and techniques, NLP enables machines to analyze and extract valuable insights from vast volumes of unstructured textual data.

The fundamental challenge within NLP lies in the inherent complexity and ambiguity of natural language. Unlike structured data prevalent in databases, human language is context-dependent, and is full of intricate linguistic patterns. This intricacy necessitates advanced methods to process and decipher text effectively.

In recent years, NLP has witnessed significant advancements, largely attributed to

the rise of deep learning techniques and the availability of vast datasets and faster and cheaper computing power. Models like Transformer-based architectures and pre-trained language models, such as BERT or GPT, have revolutionized language understanding by capturing intricate linguistic patterns and context.

The practical applications of NLP are diverse and far-reaching. From virtual assistants like chatbots and speech recognition systems to sentiment analysis for understanding customer feedback, NLP plays a pivotal role in shaping human-computer interaction.

In the context of this study, NLP techniques serve as a key pillar for comprehending character **personas** in film narratives. By employing NLP algorithms, we can extract valuable information from movie summaries, enabling the analysis and clustering of diverse **personas** across different cinematic universes.

In this study, NLP is used to perform coreference resolution, named entity recognition and dependency parsing. Using these techniques it is possible to fully extract the information that are useful to identify the characters' classes using methods like Latent Dirichlet Allocation (used for topic modeling) and the K-Means clustering algorithm.

Personas

When we refer to *personas*, we dive into the essence of a character, exploring the traits that define them in a narrative context.

To comprehend the latent nature of a character, we examine the stereotypical actions they **perform** (e.g., a PROTAGONIST might *win*), the actions **done to them** (e.g., VILLAINS might be *captured*), and the **attributes** that describe them (e.g., BEST FRIENDS might be *caring*)[1].

To capture this intuitive understanding, we can define a **persona** as a collection of three distributions: one for the words associated with actions performed by the character as an agent, one for the words related to actions directed towards the character as a patient, and one for the words describing the character's attributes. These distributions represent a fixed set of latent words or *topics* that characterize the persona, making it possible to model them.

Methods

Data Collection

To conduct this study, the initial step involves data collection to obtain movie plots. Utilizing Python and the Wikipedia API, it is possible to retrieve movie titles and their corresponding plots from the English Wikipedia. This process results in the creation of a dataset, containing movie plots for analysis.

It is important to keep in mind that Wikipedia is curated by volunteers around the world, so while the plots are mostly full of information and correct, it happens that some are shorter and, maybe, incorrect. It also happens that not all Wikipedia pages follow the same structure. For the purpose of this study they were parsed to extract the text between `\nPlot(.*?)\n` and `\n\n` *(RegEx expression used to get the text between "newline Plot(anything) newline" and "newline newline").* While mostly of the movie pages follow this format, some use other words rather than "Plot," which makes it too difficult to consider all the cases to extract the movies.

The English movie plot dataset[4] consists of 38,837 movie plots (out of 55,979 titles) updated as of July 20^{th}, 2023, extracted from the "Plot" section of the Wikipedia pages, wherever available. On average, these plots have 380 words, offering a concise overview of the movie events and character portrayals (e.g., Walter Mitty chronically daydreams, legendary photojournalist Sean O'Connell)[5].

With the obtained dataset at our disposal, we will employ NLP techniques to extract the three distributions that define a character **persona** and use topic modeling

[4]https://github.com/leonardosaveri/latent_personas/blob/main/data/movie_plots_complete.csv

[5]From the Wikipedia page: "The Secret Life of Walter Mitty (2013 film)" - updated July 20^{th}, 2023

techniques to extract classes to then cluster using KMeans.

NLP techniques

For this study, NLP was used for three main tasks: Coreference Resolution, Named Entity Recognition and Dependency Parsing.

Coreference Resolution

Coreference resolution is a fundamental natural language processing task that plays a significant role in understanding and interpreting text. It involves identifying when two or more words or phrases in a text refer to the same entity or concept. This process is essential in this study to be able to extract the largest number of information from the plots.

Why Coreference Resolution

Imagine reading the following sentence:

"Walter Mitty is a negative assets manager at Life magazine living alone in New York City. He chronically daydreams and has a secret crush on Cheryl Melhoff, a coworker."[6]

If we were to extract information about the characters without performing coreference resolution, for Walter Mitty, we would loose the information that he "chronically daydreams."

By performing Coreference Resolution with the crosslingual coreference[7] we obtain the following result:

"Walter Mitty is a negative assets manager at Life magazine living alone in New York City. Walter Mitty chronically daydreams and has a secret crush on Cheryl Melhoff, a coworker."

[6]From the Wikipedia page: "The Secret Life of Walter Mitty (2013 film)" - updated July 20^{th}, 2023

[7]https://pypi.org/project/crosslingual-coreference/

now, we can be sure that our code will extract most of the information that we want to then perform LDA.

Named Entity Recognition

Named Entity Recognition (NER) is a fundamental Natural Language Processing technique that plays a crucial role in our study of character *personas* within movie plots. NER is the process of identifying and classifying named entities, such as names of people, locations, organizations, dates, and more, within text data.

In the context of our study, NER is a highly valuable **Character Identificator.** Named Entity Recognition helps us identify and extract the names of characters within movie plot summaries. This is essential for distinguishing between different characters and their associated actions and attributes. The NER process typically involves the use of pre-trained models and libraries. These models are trained on vast amounts of text data and are capable of recognizing named entities in various languages and domains.

NER models analyze text by assigning a label to each word or token in the input text. These labels correspond to the type of named entity that the word represents. Common entity types, for Spacy, include "PERSON" for names of individuals, "GPE" for place names, "ORG" for organizational names, and more.

Named Entity Recognition models use machine learning algorithms and linguistic features to make these predictions. They consider the context of each word and its neighboring words to determine whether it is part of a named entity.

Using, for example, Spacy's en_core_web_lg[8], an English pipeline optimized for CPU and having, as components, tok2vec, tagger, parser, senter, ner, attribute_ruler and lemmatizer, we are able to pass a text like "Walter Mitty is a negative assets manager at Life magazine living alone in New York City." through it and, using a visualizer, understand how it performs NER (Figure 1).

NER models are highly valuable in information extraction tasks like ours, as they

[8]https://spacy.io/models/en

enable us to identify, categorize, and extract character names and related information from movie plot summaries. This information forms the basis for constructing character *personas* and understanding their roles and attributes within the cinematic narratives.

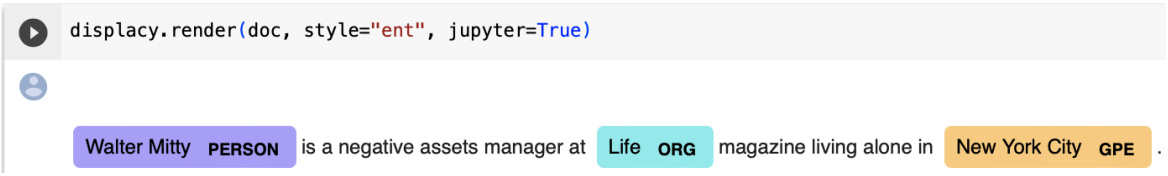

Figure 1: NER Example

Dependency Parsing

Dependency Parsing is the most important technique in our study of character *personas*. It is the process of analyzing the grammatical structure of a sentence to determine the relationships between words. Dependency parsing is particularly valuable for understanding how words within a sentence are connected and what roles they play.

In this study, Dependency Parsing is used to extract the actions performed by the characters *(agent verbs)* and the actions done to them *(patient verbs)*, as well as mapping their *adjectives* and *attributes*.

The process of Dependency Parsing involves the use of specialized NLP libraries and tools that employ syntactic and grammatical rules to parse sentences. These tools assign labels to words to indicate their grammatical relationships, such as subject, object, modifier, and more.

Dependency parsing typically starts with tokenizing the input text into words or tokens and then assigning each token a part-of-speech (POS) tag. POS tags represent the grammatical category of each word (e.g., noun, verb, adjective).

Next, the parser analyzes the dependencies between words by assigning each word a "head" word to which it is related. The relationships are often represented as directed arcs or edges connecting words in a tree-like structure. For example, in the sentence

"Walter Mitty is a negative assets manager at Life magazine living alone in New York City." we can see from the visualization (Figure 2 and 3), using spacy, "is" is a verb to the subject Mitty *(compound Walter)*.

Dependency parsing models leverage linguistic features and machine learning algorithms to make these predictions. They consider word order, grammatical rules, and contextual information to determine the most likely grammatical relationships.

In this study, Dependency Parsing is a the main step in extracting character actions, attributes, and roles from movie plot summaries. It enables us to dissect the narrative structure, uncover character interactions, and build detailed character *personas* that reflect their roles and behaviors within the cinematic context.

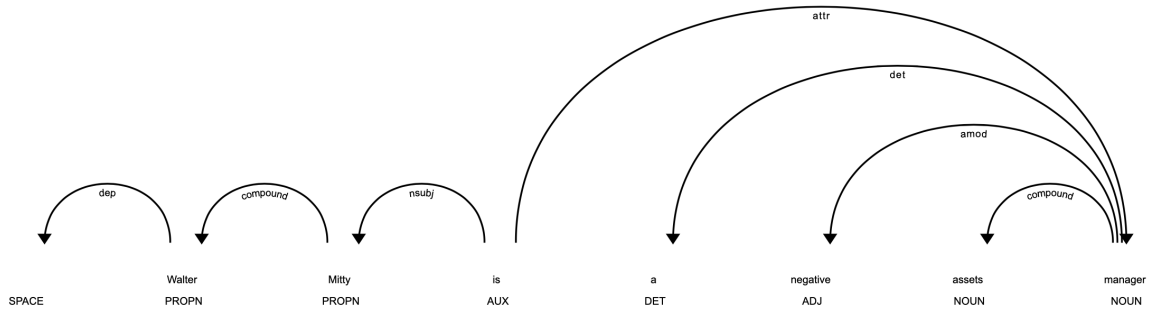

Figure 2: Dependency Parsing pt.1

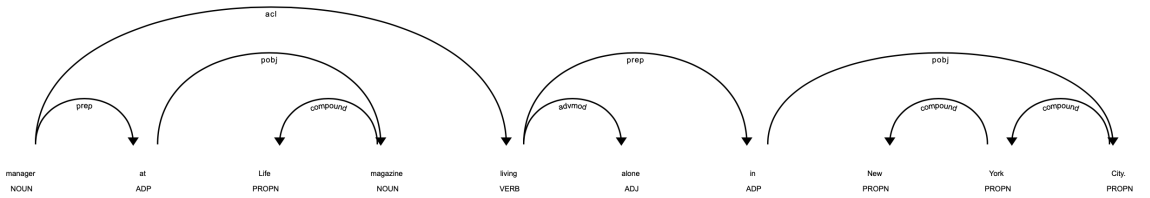

Figure 3: Dependency Parsing pt.2

Unifying Characters & Lemmatization

To make sure to get just a set of extracted information for each character, a very simple approach of name consistency was adopted. After performing coreference resolution and having extracted the attributes and verbs, shorter names information

where merged with the full name's ones. This is as changing:

"Walter Mitty is [...] a coworker. Walter attempts to contact [...]"

to:

"Walter Mitty is [...] a coworker. Walter Mitty attempts to contact [...]"

In this way, instead of considering Walter Mitty and Walter as two different characters, we combine all the information to get single characters for each plot with all the possible attributes and verbs.

In the mean time, Lemmatization was used to unify the different words describing the characters and the actions. Lemmatization involves the process of reducing words to their base or root form, known as the lemma. This transformation ensures that different inflected forms of a word are simplified to their common root, making text analysis more effective. For example, lemmatization would convert words like "running" and "ran" to their lemma, "run." By applying lemmatization to character attributes and text data, we achieve consistency in word forms and improve the accuracy of our analysis by also reducing the bias, ultimately enhancing our understanding of character *personas*.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) (Blei et al., 2003 [2]) is a generative statistical model that allows us to discover latent topics in a collection of documents. In our context, each document corresponds to a movie plot summary, and the goal is to identify underlying character *personas* by soft-clustering words that appear in similar contexts. LDA assumes that words are drawn from a mixture of K latent word topics, where each topic represents a distribution over the vocabulary words.

The generative story of the LDA model, as adapted for character *personas*, unfolds as follows:

For each movie script $\mathbf{w}$ in a corpus $\mathbf{D}$ (collection of documents):

1. It chooses $N \sim \text{Poisson}(\lambda)$

2. It chooses $\omega \sim Dir(\alpha)$

3. For each of the N descriptors w_n:

   (a) It chooses a *persona* $z_n \sim$ Multinomial(ω)

   (b) It chooses a descriptor w_n from $(p(w_n)|z_n, \beta)$, a multinomial probability conditioned on the persona z_n.

Applying LDA to the dataset of characters, where the characters are represented in a Pandas[9] dataframe with all the attributes, active and patient verbs, allows us to obtain character classes. This clustering helps us identify common patterns and similarities among characters, enabling a more nuanced analysis of character *personas* and their interactions within cinematic narratives.

KMeans

KMeans clustering is an unsupervised machine learning technique that will be used in this study to group characters with similar *personas* into distinct clusters, shedding light on common character archetypes and behaviors within cinematic narratives.
At its core, KMeans clustering is a partitioning algorithm that seeks to divide a dataset into "K" distinct, non-overlapping subgroups or clusters. Each data point belongs to the cluster with the nearest mean value, making the clusters as internally similar as possible while ensuring maximum dissimilarity between them.
The key components of K-Means clustering include:

1. **K:** The user specifies the number of clusters (K) they want the data to be divided into. This is a critical parameter as it defines the granularity of the clustering.

2. **Centroids:** Each cluster is represented by a centroid, which is essentially the mean of all data points in that cluster. The centroids act as representatives of their respective clusters.

[9] https://pandas.pydata.org

3. **Distance Metric:** K-Means relies on a distance metric (typically Euclidean distance) to measure the similarity between data points and centroids. Data points are assigned to the cluster with the nearest centroid.

4. **Iteration:** The algorithm iteratively assigns data points to clusters and updates the centroids. This process continues until convergence, where the assignment of data points remains unchanged.

The KMeans algorithm then operates in the following steps:

1. **Initialization:** K initial centroids are randomly selected from the data points or by other methods. These centroids serve as the starting points for the clusters.

2. **Assignment:** Each character is assigned to the cluster whose centroid is closest to it, based on the chosen distance metric. This forms the initial clusters.

3. **Update:** The centroids of the clusters are recalculated as the mean of all data points in each cluster. This step re-positions the centroids to better represent the data points within their respective clusters.

4. **Reassignment:** Data points are reassigned to the cluster with the nearest centroid based on the updated centroids. This process repeats until convergence, where data point assignments remain unchanged.

The Study

The *(non)* Preprocessing

For the purpose of this study, extensive preprocessing of the text data was not required. In fact, Capital Letters and Stop Words, that in classical preprocessing are often removed, played crucial roles in extracting the necessary information.

Capital Letters proved to be particularly significant in Named Entity Recognition, a process employed, in our case, by Spacy to identify characters within sentences and determine where to initiate the search for attributes and verbs. On the other hand, Stop Words were fundamental in understanding the relationship between a character and its attributes, agent, and patient verbs.

In this scenario, employing the classical approach to preprocessing text for NLP proved to be of limited utility. Hence, it was more effective to work directly with the raw text, extracting the relevant information in the form of dictionaries.

The only preprocessing step conducted before extracting the three sets of words for each character of each plot was Coreference Resolution.

Coreference Resolution is the process that involves determining the noun or name to which a specific pronoun refers and substituting it with the corresponding noun or the possessive form of it. The resolution of these references helps in gaining a wider selection of attributes and actions for each character of the plot.

To perform Coreference Resolution, the crosslingual coreference[10] module was utilized. This module, that works with Spacy, is trained to handle this task in multiple languages using a model based on English data. It leverages cross-lingual embeddings, which

[10]https://pypi.org/project/crosslingual-coreference/

allows it to work in languages with similar linguistic structures rather than having to re-train.

The available models within the crosslingual coreference package and their respective results are summarized in the table below:

Model	Score	Description
spanbert	83	Best quality for english texts.
xlm_roberta	74	Best quality for multi-lingual texts.
minilm	74	Best quality speed trade-off for mult-lingual and english texts.

Out of these models, the "AllenNLP spanbert" model, achieving a score of 83 on OntoNotes Release 5.0 English data, was chosen since the text utilized in this study is in English. While the other models were best if used in a multi-lingual context, "spanbert" was yielding the best results on English text, making it the best choice.

By performing Coreference Resolution, the texts are effectively resolved to represent the characters and their actions with clarity and consistency. This preprocessing step prepares the texts for the subsequent extraction of the most comprehensive information possible, laying the groundwork for character persona analysis and clustering.

Extracting the information

The information extraction process is the fundamental stage in this study as it involves extracting relevant data from the movie plots that will be used to define and characterize character *personas*. The primary objective is to identify and collect information about characters, their *attributes*, and the actions they perform as *agents* or *patients.*

To achieve this, we employed NLP techniques. Specifically, we used the Spacy library to process the movie plots and extract relevant information about characters and their attributes. The function `extract_characters_attributes`[11] was implemented to perform this task.

[11] https://github.com/leonardosaveri/latent_personas/blob/main/notebooks/extract_info_movies.ipynb

The function starts by creating a Spacy Doc object for each movie plot. After performing Coreference Resolution it then iterates through the entities detected in the plot, focusing on those labeled as "PERSON". For each character entity, the function extracts various pieces of information:

- **Character Name**: The name of the character is extracted and used as the key in the dictionary to store their information.
- **Attributes**: The function looks for adjectives and nouns that modify the character entity, considering them as attributes that describe the character. Tokens with dependency labels such as "amod" (adjectival modifier) or parts of speech like "ADJ" (adjective) and "NOUN" (noun) are considered to capture the attributes associated with the character.
- **Agent Verbs**: Agent verbs are verbs where the character is the one performing the action as the agent. The function identifies agent verbs by looking for specific grammatical dependencies. Tokens with dependency labels like "nsubj" (nominal subject) that are connected to a "VERB" indicate that the character is the one performing the action as the subject. These verbs reflect the character's proactive role in the plot.
- **Patient Verbs**: Patient verbs are verbs where the character is the one undergoing the action as the patient. Similar to agent verbs, the function identifies patient verbs by examining specific grammatical dependencies. Tokens with dependency labels such as "dobj" (direct object), "nsubjpass" (nominal subject of passive), "iobj" (indirect object), or "prep" (prepositional argument) connected to a verb indicate that the character is affected or involved as the object of the action, making them the patient of the verb.

Handling Complex Cases: To ensure a comprehensive extraction of attributes and verbs, the function traverses the subtree of the character entity's root. This step accounts for complex cases where attributes or verbs may be modified by other parts

of the sentence. By considering additional adjectives and verbs within the subtree, the function captures a broader range of descriptive attributes and verbs that provide context to the character's portrayal.

The resulting dictionary contains the extracted information for each character, including their attributes, agent verbs, and patient verbs.

By leveraging these specific grammatical dependencies and linguistic patterns, the function effectively identifies, extracts and lemmatizes relevant information that helps characterize each character's persona and their roles within the movie plot.

After extracting all the descriptors of the characters, we end up with 194,215 characters containing at least one term for at least one of the categories (*patient*, *agent* or *attribute*).[12]

Dirichlet Persona Model

The Latent Persona Model is the model we decided to use for out study, serving tool for topic modeling. This model allows us to uncover hidden patterns and structures in character behaviors, actions, and attributes, ultimately leading to the creation of rich character *personas*.

Generating Words

Before delving into the Latent Persona Model itself, we begin by preprocessing the textual data associated with characters. The goal is to transform unstructured text into a format suitable for analysis. Specifically, we aim to extract relevant words that capture the essence of each character's attributes, agent verbs, and patient verbs.

The process begins with the application of a simple word extraction function. For each character, we preprocess their attributes, agent verbs, and patient verbs, converting them into a list of words. This step ensures that the textual information is in a structured and manageable format for subsequent analysis.

[12]https://github.com/leonardosaveri/latent_personas/blob/main/data/characters_info_final.csv

Creating a Dictionary and Corpus

With the extracted words in hand, we move on to building a dictionary and a corpus. The dictionary contains a unique identifier for each distinct word present in the character descriptions. Meanwhile, the corpus is a collection of documents, where each document represents the word frequency distribution for a character's attributes. The dictionary and corpus provide the foundation for topic modeling, allowing us to quantify the presence of words and their relationships within the character descriptions.

Latent Dirichlet Allocation

The heart of the Latent Persona Model lies in the application of Latent Dirichlet Allocation. In our study, we leverage LDA to uncover latent topics within character attributes, agent verbs, and patient verbs.
In our study:

1. **Topic Discovery:** LDA assumes that each character's attributes, agent verbs, and patient verbs are generated from a mixture of latent topics. These topics represent clusters of words that frequently co-occur in the character descriptions. The number of topics is a parameter specified in advance (for this study, 100 topics, as suggested in David Bamman's work [1]).

2. **Word Distributions:** For each topic, LDA estimates a probability distribution over words. These distributions capture the words most associated with each topic.

3. **Topic Assignments:** LDA assigns each word in a character's description to one of the latent topics, based on the likelihood of the word belonging to each topic.

4. **Character Persona Creation:** After running the LDA model, we obtain a distribution of topics for each character. These topic distributions represent the character's persona, reflecting their attributes, behaviors, and roles.

LDA allows us to discover character archetypes, identify commonalities among characters, and uncover hidden traits within cinematic narratives.
Through this approach, we gain a deeper understanding of character *personas*, enabling us to analyze character behavior, interactions, and roles across a wide range of movies.
To perform LDA, we decided to use the LDA Multicore from gensim[13] to be able to utilize all the cores of the CPU and make the analysis faster. The 'LdaMulticore' model is responsible for extracting latent topics from character attributes, patient and agents verbs. For the three analysis, we always used the following parameters:

```
lda_model = gensim.models.ldamulticore.LdaMulticore(
        corpus=corpus, id2word=id2words, num_topics=100,
        random_state=42, chunksize=1000, passes=10)
```

Let's break them down:

- 'corpus': This parameter represents the preprocessed corpus of character attributes, agent verbs or patient verbs. It's the collection of characters used for topic modeling.

- 'id2word': The 'id2word' parameter is a dictionary mapping unique identifiers to words in the vocabulary. It allows the model to understand the relationship between words and topics.

- 'num_topics': Using this parameter we specify the number of topics we want the model to discover within the character attributes. This parameter determines the granularity of our topic analysis.

- 'random_state': The 'random_state' parameter ensures reproducibility. It sets the initial state of the random number generator, ensuring consistent results across runs.

- 'chunksize': This parameter controls the number of documents processed at each iteration. It influences the efficiency of the model.

[13]https://radimrehurek.com/gensim/models/ldamulticore.html

- ‘passes‘: The ‘passes‘ parameter determines how many times the model iterates over the entire corpus. Each pass refines the topic distributions.

Why 100 topics

In this study, we decided to use 100 topics for each of the three distributions mainly because of Bamman's work and because training a model for various number of topics (for example, between 10 and 150) would require much more time than we could afford. For future research, it would be optimal to use either coherence scores *(v_c, or u_mass)*, or silhouette scores to understand if 100 is the optimal number of topics or if it would be better to decrease or increase it.

Kmeans

Having now, for each character, three lists of distributions, in the form [(topic_n, percentage of topic_n in the document), ...] we can apply KMeans (see Section "KMeans" on page 11) to cluster the characters together. Also here, following Bamman's work, we want 100 clusters to represent 100 different personas. We will define a matrix where, for each characters, we get the "percentage of topic_n in the character" as one of the features and we will perform KMeans over it, hoping to find different and defined archetypes. To obtain more and less specific classes one can change the (K) parameter.

We used the KMeans packet form sklearn.cluster[14] (with random_state = 42 *(the meaning of life, the universe, and everything (D. Adams, 1989 [3])* over the 300 features for each character we are able to cluster similar personas together.

Why 100 personas

Again, we decided to use 100 classes mainly because of Bamman's work and because training a model for various number of topics (for example, between 10 and 150) would require more time than we could afford. In the future, it would be interesting to analyze using, for example, the elbow method (Thorndike, 1953. [4]), whether a different amount of personas would be more optimal.

[14]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Results

To visualize the output obtained by the LDA model, we used the pyLDAvis[15] library. PyLDAvis is a python library for interactive topic model visualization, which will make it easier to understand the different clusters and their main words.
In a PyLDAvis visualization we can see, on the left we can see a representation of the topics in a distance map. In this representation:

- Each topic is visually represented by a bubble, with its size proportional to the percentage of characters in the corpus related to that topic (*persona* class).
- The distance between bubbles reflects their dissimilarity; the farther apart they are, the more distinct they become.

On the right we can see:

- The frequency of words in the entire corpus is denoted by blue bars. When no topic is selected, these bars display the most commonly used words.
- Red bars provide an estimate of how many times a specific term is associated with a particular topic.

Analyzing character classes from attributes

Using pyLDAvis we obtain the topic representation graph shown in figure 5.
Ideally, topics will not overlap and, given the big number of topics we wanted to model we are happy to see that most of them are not overlapping, with just a few exception.

[15] https://pypi.org/project/pyLDAvis/

Let's now analyze a couple of topics that are close to one another (Figure 6 and 7)
We can see in Figure 6 that selecting **topic 20** we are able to see the most relevant terms for the topic. This include *"teenage"* and *"angry"* which already define a pretty big group of movie characters. Near it, we can see **class 55**, described by terms such as *"parent"*, *"feeling"*, *"worried"* (Figure 7). This is another big class of movie characters, that we can find in many different movies and that can often be found in the same movies.
PyLDAvis also makes it easy to understand in which topics a specific word is most likely to appear, for example, for the word *"scientists"* we see that it is very likely to appear in the left side of the distance map (Figure 8) while a word like *"indian"* is more likely to appear in a topic that is in the right corner (Figure 9). As we know, there aren't many Indian scientists represented in movies and this result are feasible with the findings.
Being able to fully understand these topic classes would require much more time than we can afford, so we will just show some of the results that we were able to obtain without diving too deep into the specific bubbles.

Analyzing character classes from agent verbs

By plotting the topic that were found by the LDA model with the agent verbs (Figure 10), we can see, even if the bubble do overlap a little more than in the case of the attributes, they are still pretty disjoint.
By exploiting one of the topics, we can see that in **topic 8** (Figure 11) is clearly a cluster of *"murderers."* The main terms is in fact *"kill"*, *"decapitated"*. We can see that most of the terms are combines together and that in the case of the mentioned, they are almost only in the selected **topic 8** (the blue bar is equal to the red bar).

Analyzing character classes from patient verbs

By plotting the topic that were found by the LDA model with the patient verbs (Figure 12), we can see they are still fairly disjoint between them.

By exploiting one of the topics, we can see that **topic 55** (Figure 13) is clearly a cluster of *"victims."* The main terms are in fact *"murderer,"* and *"drown,"* but if in the case of the agent verbs a term like *"kill"* would refer to a murderer, in the case of passive verbs this would describe an action that was done to the characters, making them most probably, a victim.

Analyzing character classes

Much more interesting is the case in which we are able to determine classes of characters considering all the different descriptors we have obtained from the text. To do so we have used KMeans (see Section "KMeans" on page 11) while still considrering 100 different classes.

By applying KMeans we are able to get general classes that represent, more or less, the characters that we have. By looking at Figure 4 we are able to see that with the exception of **class 27,** we don't have any big outlier. **Class 27,** with a total of 50,757 characters, is the most present, while **class 34,** with a total of 307 is the least present. On average, each class has around 1942 characters (black - - line in 4).

Let's now dive deeper into **class 27,** the class with most characters, and **class 34,** the class with least characters.

Class 27 has many different protagonists, Walter Mitty (from "The Secret Life of Walter Mitty, 2013"), Harry Potter (from the analogous series), James Bond (from the analogous series) and more. This shows us that the class was able to combine different characters together, mainly protagonist.

For **class 34,** on the other hand, characters like Kaitlin (from the movie "Burnt, 2015") or King Harold (from the movie "Shrek Forever After, 2010") where clustered together. This are both side characters, but they are not, to our knowledge, related

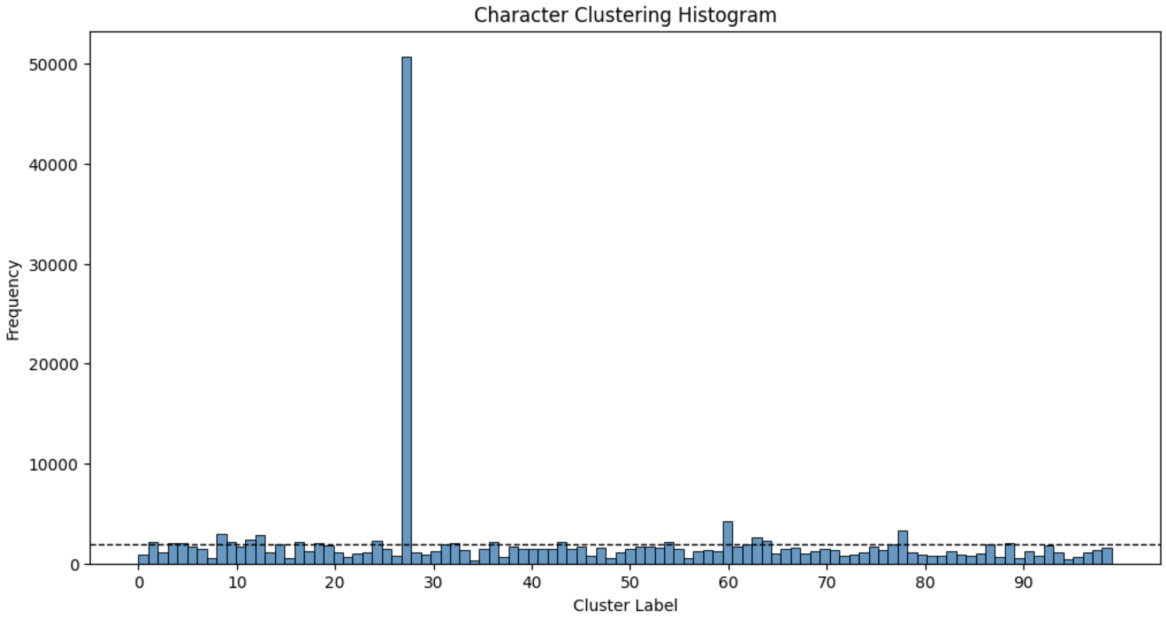

Figure 4: KMeans Histogram

in any way. These could be because the KMeans algorithm is not the best way to cluster, but also because **class 34** might be made of characters who were not able to be inserted into other classes.

It is nice to notice, by looking at specific characters, which are the relevant terms that made them cluster together. For example, by looking at "You-Know-Who"[16] from "Harry Potter and the Prisoner of Azkaban (2004)" we can see that he is clustered into **class 78.** By looking at many characters inside **class 78** we can see that many shares the term *"kill"* as a patient verb. "You-Know-Who"[16] is known to be the *villain,* but in the third movie of the Harry Potter franchise while he doesn't appear, and is only spoken about, it is mentioned, *wrongly*, that somebody wants to kill him (see section "Limitations of the study" on page 26). It is also interesting to see how some characters change over the movies, and that for example "He-Who-Must-Not-Be-Named"[16] is clustered in different clusters in different movies. This is because in other movies, he could be described as a killer, rather the one who was killed.

This is something that, in some cases, can help us understand the overall character development but in others it might be caused by the way the characters are described by people writing on Wikipedia (no coherent way of describing the characters) (see

[16]Voldemort

section “Limitations of the study” on page 26)

Using KMeans we were able to construct simple clusters out of the outputs of the LDA algorithm to group characters over their features, obtaining 100 different classes and finding similarities between them.

The codes, data, jupyter notebooks and outputs of this study can be found at https://github.com/leonardosaveri/latent_personas

Conclusions

Ethical and Bias Considerations

In this study about *personas* through Natural Language Processing techniques, it is important to address ethical concerns and biases that may arise during the course of our analysis. Here we want to consider the ethical dimensions of stereotypes and bias, considering both algorithmic influences and the collaborative nature of Wikipedia plot summaries.

Stereotypes: Character *personas* often align with recognizable archetypes or stereotypes present in cinematic narratives. While these stereotypes can provide valuable insights into character construction, they also carry harmful biases or reinforce social norms. It is essential to approach these character representations with sensitivity, recognizing the potential for misrepresentation or reinforcement of stereotypes.

Algorithmic Bias: NLP techniques, including topic modeling, are susceptible to algorithmic bias. Pre-existing biases within training data can result in wrong interpretations or characterizations.

Collaborative Plot Summaries: Wikipedia plot summaries are often authored collaboratively by individuals with diverse backgrounds and perspectives. While this collaborative nature enhances inclusivity, it can introduce subjectivity and inconsistency in writing style and content. This variations may influence our analysis, requiring us to consider the potential for diverse interpretations within plot summaries.

To try to address these ethical and bias considerations:

- We employed preprocessing techniques to mitigate algorithmic bias such as lemmatization, which is useful in removing bias against gender and has the additional benefit of reducing data sparseness. (Kinshuk Sengupta et. al., 2021 [5])

- Keeping in mind the collaborative nature of Wikipedia and factoring in the potential influence of subjective writing styles, striving for a balanced interpretation of character attributes.

Limitations of the study

While performing the study we were able to get different results, but it is crucial to acknowledge the study's limitations. Understanding these constraints enhances the interpretation and contextualization of our findings.

- **Inconsistencies in Textual Sources**: One notable limitation arises from the diverse origins of our textual sources. These summaries were authored by various individuals, resulting in inconsistencies in writing style and content. This variability may introduce subjectivity and ambiguity into our analysis, impacting the reliability of character *persona* extractions.

- **Incomplete Character Descriptions**: Many characters within cinematic narratives may not receive comprehensive descriptions in plot summaries. The absence of detailed character attributes, actions, or relationships can limit our ability to construct *personas* for every character. This limitation highlights the importance of considering both the presence and absence of information.

- **Dependency Parsing Challenges**: Dependency parsing, while being a powerful tool for information extraction, is not immune to errors. Ambiguities in language and sentence structures can lead to misinterpretations, potentially affecting the accuracy of our character *persona* extractions. For example, on

page 23, we say that our code assigned, *wrongly*, the patient verb "kill" to "You-Know-Who"[17]. This is the sentence it was taken from: *"Reuniting with his best friends Ron Weasley and Hermione Granger, Harry learns that Sirius Black, a convicted supporter of Lord Voldemort, has escaped Azkaban Prison and intends to kill him."*[18] We can see that the above is a fairly complex sentence and that the feature extraction algorithm is assigning to the wrong character (instead of assigning it to Harry Potter) the patient verb *"kill."*

- **Optimal Parameter Selection**: The selection of parameters in our Latent Dirichlet Allocation and KMeans clustering models plays a pivotal role in the quality of our results. While we adopted specific parameter values (e.g., 100 topics in LDA) based on previous research (Bannan et. al., 2013[1]), these values may not be universally optimal. Further experimentation with different parameter settings is essential to assess the robustness and sensitivity of our models.

- **Computational Complexity**: The process of extracting character descriptors demands substantial computational resources and time. This limitation affects the scalability and efficiency of our study, necessitating thoughtful considerations of computational constraints. This also proved to be impactful in our analysis because it prevented us to perform many different trials with many different texts.

- **Generalization to Other Languages**: Our study primarily focuses on English-language cinematic narratives. Extending the analysis to other languages may introduce additional complexities, mainly due to the fact that many of the models available in English are not available, or are not as good, in other languages, requiring adaptations and considerations specific to each language's linguistic characteristics.

[17] Voldemort

[18] From the Wikipedia page: "Harry Potter and the Prisoner of Azkaban (film)" - updated July 20^{th}, 2023

By acknowledging these limitations, we aim to provide a comprehensive perspective on the scope and boundaries of our research. These constraints provide inspiration for future researches to address these challenges and further enhance our understanding of character *personas*.

Future Work

As we conclude our exploration into character *personas* through Natural Language Processing, several promising directions emerge for future work and refinement of our approach. These possibilities extend from feature extraction improvements to broader linguistic and cultural dimensions.

- **Perfection of Feature Extraction**: One promising direction involves the fine-tuning and perfection of feature extraction methods. Dependency parsing, while powerful, can benefit from ongoing development and code improvements to enhance accuracy and reliability. Focusing on improving the identification of character attributes, actions, and relationships will lead to more precise *personas*.

- **Parameter Exploration**: To further refine our models, it is imperative to explore a range of parameters, such as the number of clusters in KMeans (one of the possible methods to find the optimal number of clusters could be the *elbow method*, Thorndike, 1953[4]) and different settings for Latent Dirichlet Allocation over the three distributions (this could both consider coherence scores or silhouette scores). This exploration will uncover the optimal configurations for character *persona* analysis.

- **Creation of Knowledge Graphs**: Building comprehensive knowledge graphs of characters based on their features is a promising application. This task could help analyse similarities between characters and could help to visualize and understand the output of the study.

- **Multilingual Analysis**: Expanding our research to include cinematic

narratives described in languages beyond English is an exciting possibility. Wikipedia provides data in various languages, but challenges like linguistic styles and differences in narrative styles must be addressed for meaningful analysis.

- **Metadata Integration**: Incorporating metadata from movies and characters, such as genre and character's descriptors (age, gender etc.), can further enrich our character *persona* analysis. This approach could provide valuable context and insights into character development.

In addition to these advancements, future work should consider the utilization of movie scripts as a more structured data source. To do so effectively, we propose the development of a framework for dependency parsing that extends beyond textual attributes. This framework could use sentiment analysis, linguistic patterns, and character interactions, including the dialogue sentiment and director's cues. By shaping this framework, we can unlock a deeper understanding of character *personas* and their intricate dynamics.

To illustrate, let's consider an excerpt from "Midnight in Paris (2011)."[19]

```
INT. CAFE #3 - NIGHT                                          24

A little late night cafe, very bohemian.  Scott, Zelda and
Gil enter, the group having thinned out.  The Fitzgeralds
drink a lot.

                    ZELDA
          Une bouteille de bourbon.

                    SCOTT
               (stops at another table)
          Greetings and salutations.  You'll
          forgive me - I've been mixing grain
          and grappa...  This is Gil - Gil?
          Yes, Gil.

                    GIL
          Gil Pender.

                    HEMINGWAY
          Hemingway.

                    GIL
          Hemingway?  Hey, is this some kind
          of a -

                    HEMINGWAY
          You liked my book?

                    GIL
          Liked - I loved - everything you
          wrote -
```

[19] from the script by Woody Allen

Extracting information from movie scripts like this passage, we could analyze characters based on sentiment, linguistic style, and interactions. We could obtain descriptors from the director's note ("The Fitzgeralds *drink*") and obtain the sentiment *(using sentiment analysis)* from the text (GIL is *enthusiast*). This would be a much much harder task to complete, and the use of LLMs will probably be needed to extract as much information as possible from the text. This holistic approach could lead to more complete character classes, enriching our cinematic understanding and storytelling.

The future of character analysis in movies holds tremendous promise. By delving into these research possibilities, we can deepen our understanding of characters, their complexities, and their significance in the world of cinema through the use of NLP. This avenue of exploration allows us to gain valuable insights into how stereotypes are portrayed and how characters evoke emotions that resonate with audiences, creating a profound connection with the story. As we navigate this exciting terrain that combines NLP and movies, we continue to reshape and illuminate the world of cinematic storytelling, bringing new dimensions and perspectives to the art of filmmaking.

THE END[20]

[20]Old movie closing credit - https://www.moviemaker.com/why-dont-movies-finish-with-the-end-anymore-reid-rosefelt-20100330/

Bibliography

[1] David Bamman, Brendan O'Connor, and Noah A. Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[3] D. Adams. *The Hitchhiker's Guide to the Galaxy.* Hitchhiker series. Harmony Books, 1989.

[4] Thorndike. Vol. 18, pp.266-267. *R.L. Psychometrika*, 1953.

[5] Kinshuk Sengupta, Rana Maher, Declan Groves, and Chantal Olieman. Genbit: measure and mitigate gender bias in language datasets. *Microsoft Journal of Applied Research*, 16:63–71, October 2021.

Appendix A: pyLDAvis Images

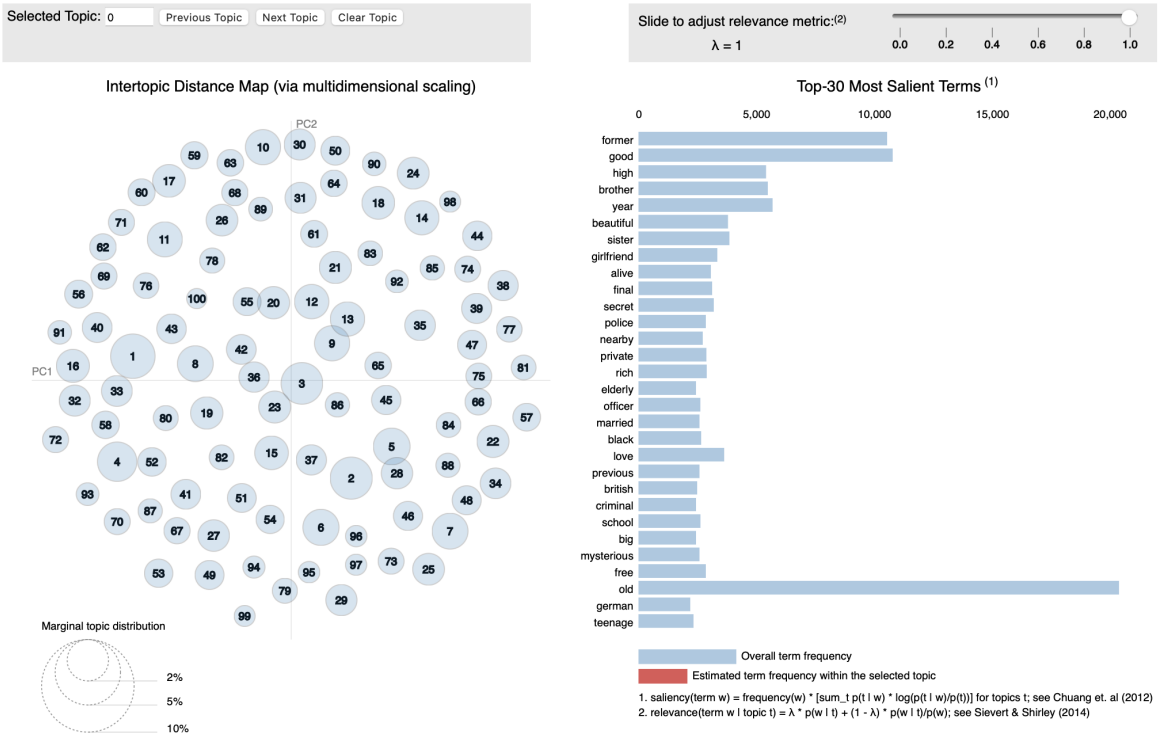

Figure 5: pyLDAvis Attributes

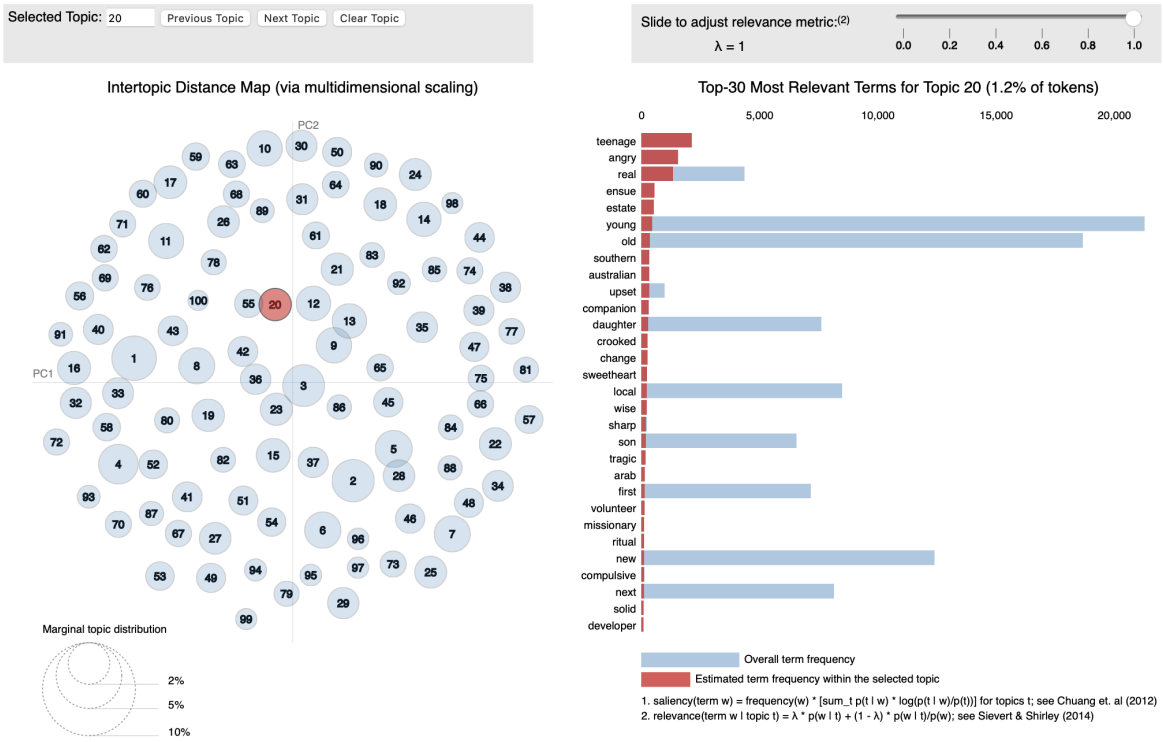

Figure 6: pyLDAvis Attributes - Class 20

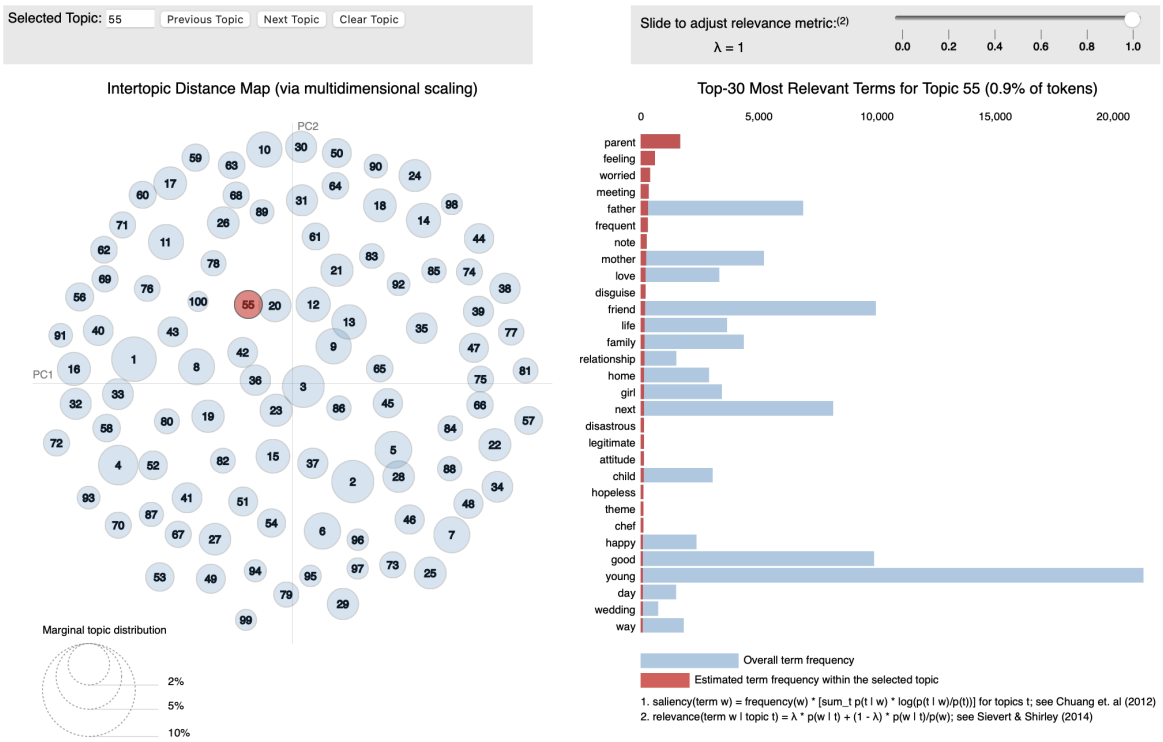

Figure 7: pyLDAvis Attributes - Class 55

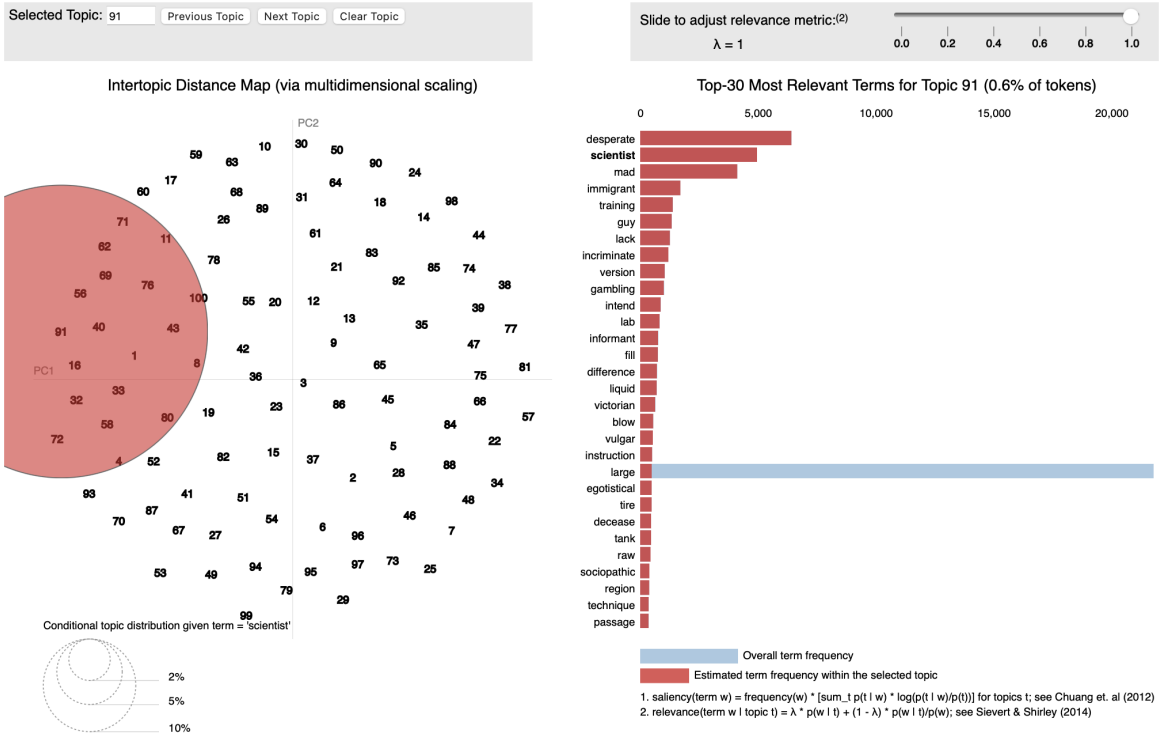

Figure 8: pyLDAvis Attributes - Word "Scientists"

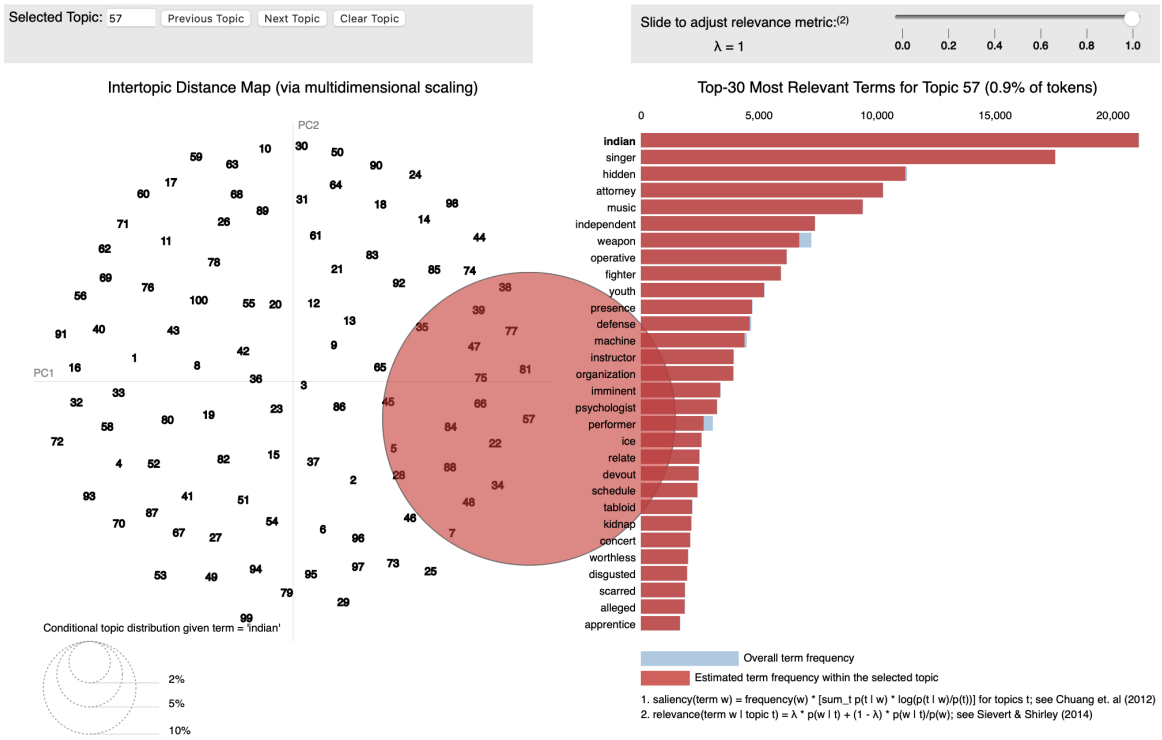

Figure 9: pyLDAvis Attributes - Word "Indian"

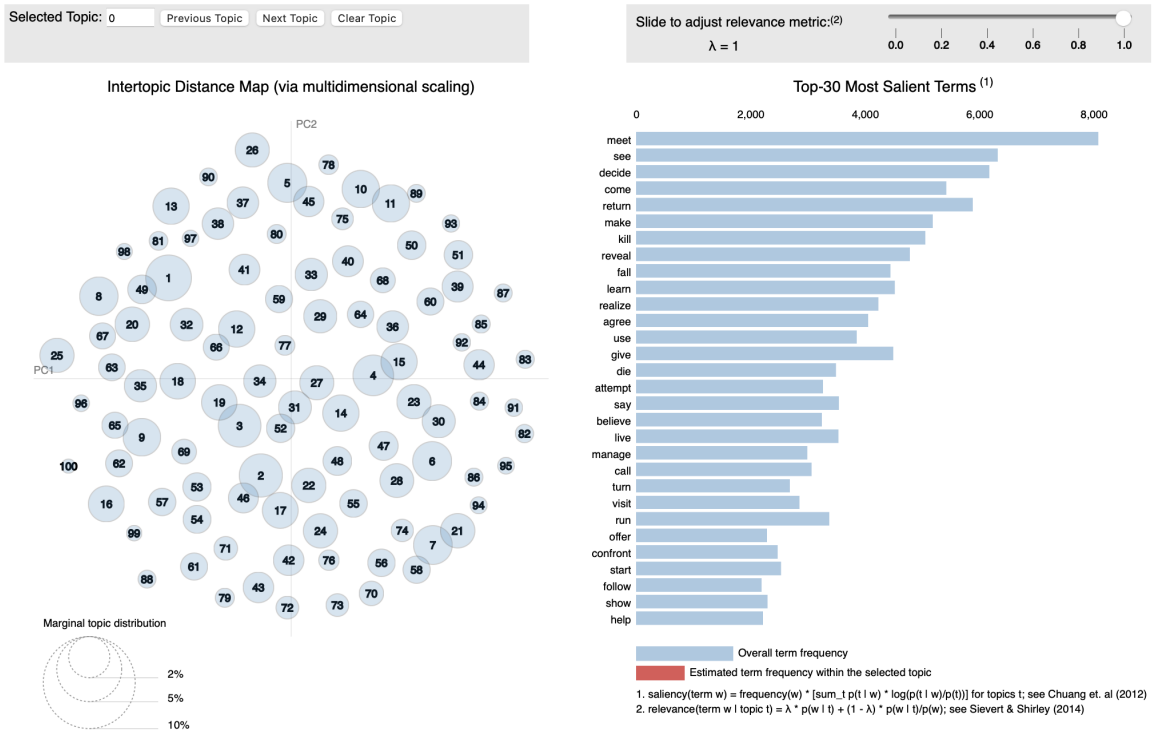

Figure 10: pyLDAvis Agents

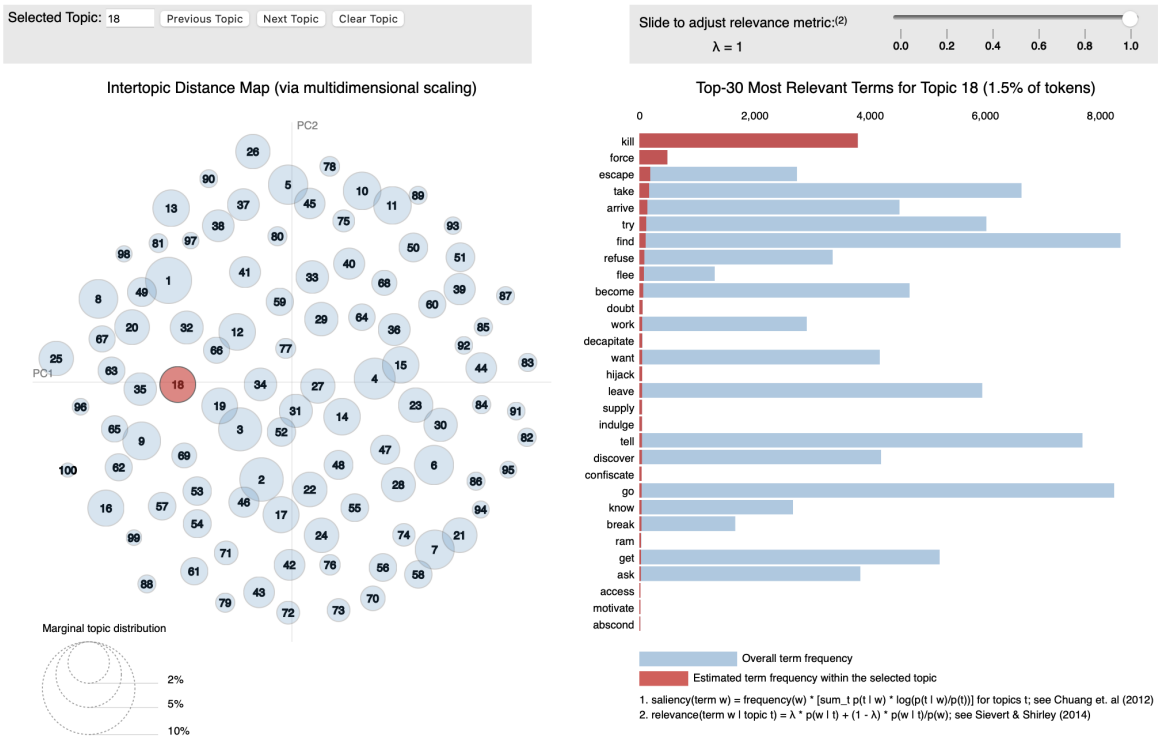

Figure 11: pyLDAvis Agents - Topic 18

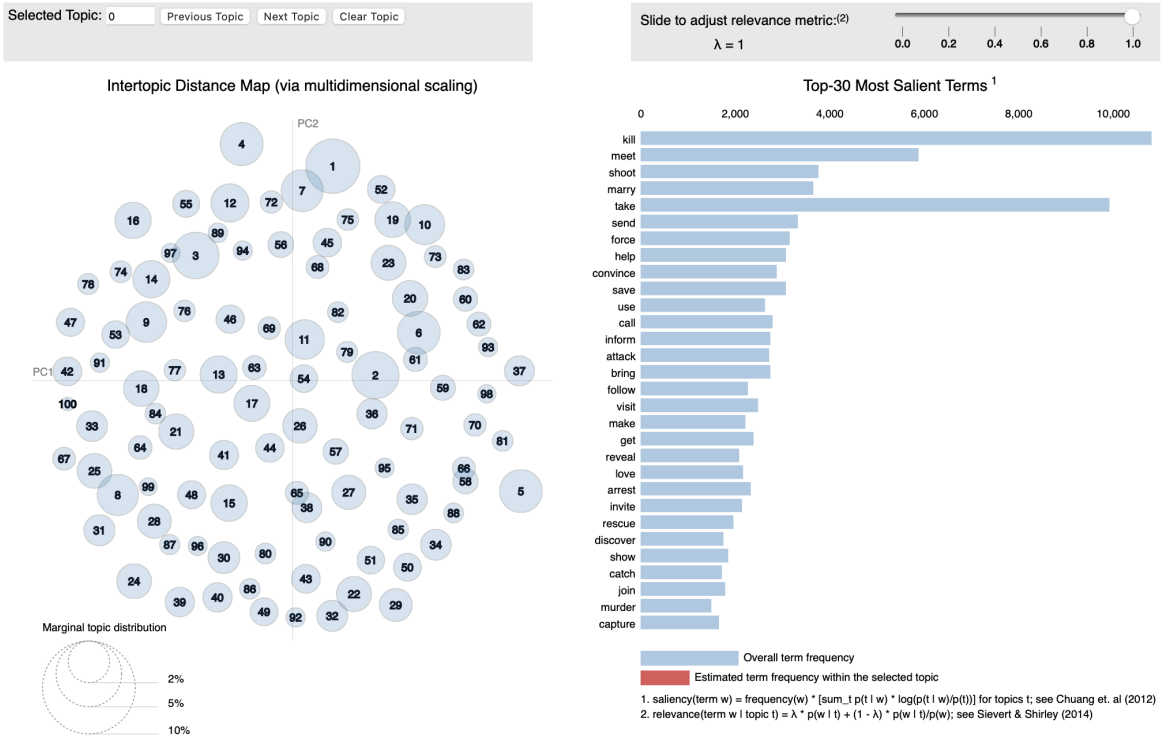

Figure 12: pyLDAvis Patient Verbs

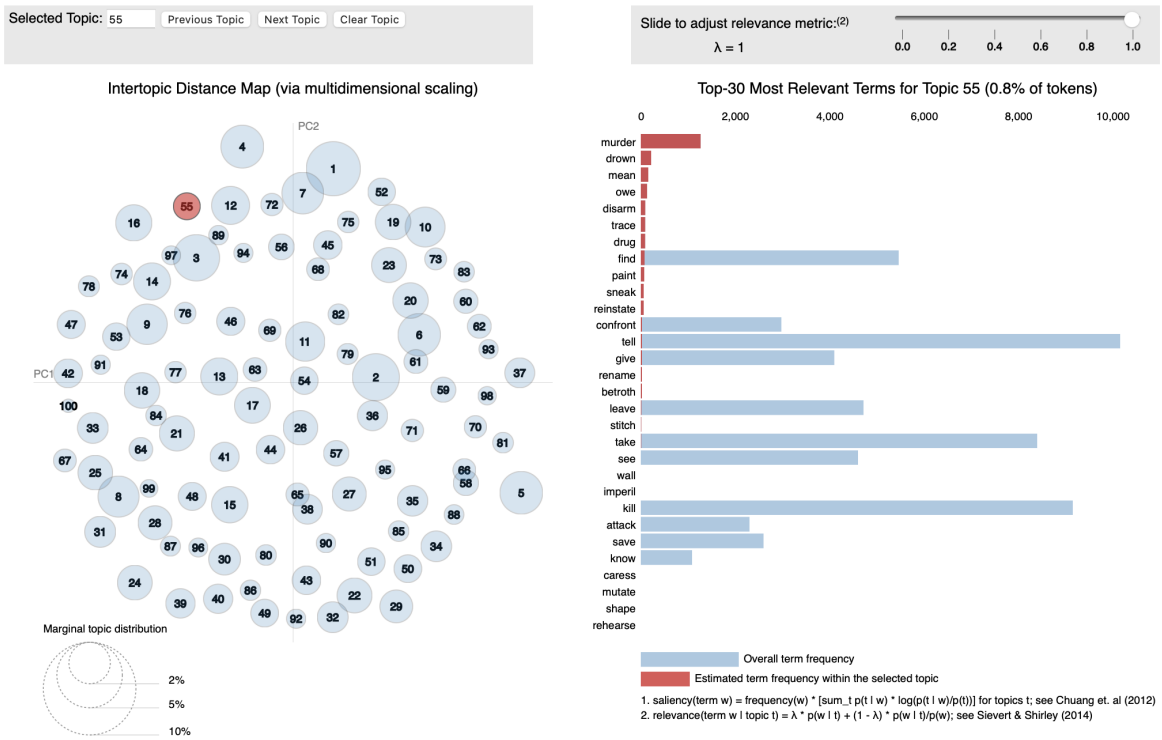

Figure 13: pyLDAvis Patient Verbs - Topic 55