TECHNICAL UNIVERSITY OF CRETE

Providing Personalized Recommendations for Interactive Story Generation

Author:
Petros I. Portokalakis

Thesis Committee:
Associate Prof. Georgios Chalkiadakis (Supervisor)

Associate Prof. Michail Lagoudakis

Associate Prof. Georgios Giannakakis

A thesis submitted in partial fulfillment of the requirements for the degree of Diploma in Electrical and Computer Engineering

in the

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

November 8, 2020

TECHNICAL UNIVERSITY OF CRETE

Abstract

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Diploma in Electrical and Computer Engineering

Providing Personalized Recommendations for Interactive Story Generation

by Petros I. Portokalakis

Interactive narrative is a form of digital entertainment where players can create or influence a storyline through actions, usually by controlling the role of one (or more) characters in a virtual world. In story-based games or any other interactive story system in general, a drama manager is an omniscient agent that acts to guide the user through the story space. While drama managers tend to improve user enjoyment, they do not take into account the user's preferences. In this thesis, we present a drama manager that tries to tackle the sequential recommendation problem, while taking into account user preferences. In order to create and maintain user engagement, we present a recommendation approach using probabilistic topic modeling, intertwined with reinforcement learning. We use the Latent Dirichlet Allocation topic modeling algorithm, applied in a Choose Your Own Adventure (CYOA) book to capture its latent topics. The key aspect of our drama manager is that we model the user herself as an evolving document represented by its respective mixture of latent topics; and which is appropriately updated every time the user consumes an item. Also, inspired by reinforcement learning literature, we introduce the use of variable learning rate for drama managers, directly associated with the user model updating, and based on the well-known "Win of Learn Fast" reinforcement learning method. The algorithm is trained using all the Wikipedia articles referring to books. We also provide an efficient parser for Wikipedia articles. Experimental evaluation results are promising, showing that our drama manager is capable of providing efficient recommendations to the user.

Η διαδραστική αφήγηση (interactive narrative) είναι μία μορφή ψηφιακής ψυχαγωγίας όπου οι χρήστες δημιουργούν ή επηρεάζουν μία πλοκή. Αυτό συνήθως γίνεται μέσω του έλεγχου ενός (ή περισσότερων) χαρακτήρων σε έναν εικονικό κόσμο. Σε παιχνίδια βασισμένα στην πλοκή ή άλλα διαδραστικά συστήματα, ο διαχειριστής δράματος (drama manager) είναι ένας παντογνώστης πράκτορας που έχει σαν σκοπό να καθοδηγήσει τον χρήσει μέσα στον χώρο που δημιουργείται από όλες τις πιθανές πλοκές που μπορεί να προκύψουν (story space). Στα περισσότερα διαδραστικά συστήματα αφήγησης εώς τώρα, ένας διαχειριστής δράματος προσπαθεί να βελτιώσει την εμπειρία του χρήστη, χωρίς όμως να λαμβάνει υπόψη τις προτιμήσεις του. Σε αυτήν την διπλωματική εργασία, παρουσιάζουμε έναν διαχειριστή δράματος που προσπαθεί να λύσει το πρόβλημα της ακολουθιακής σύστασης (sequential recommendation problem) ενώ ταυτόχρονα λαμβάνει υπόψη τις προτιμήσεις του χρήστη. Το σύστημά μας πρέπει να δημιουργήσει και να διατηρήσει την ενασχόληση του χρήστη με αυτό. Οπότε, προτείνουμε μια προσέγγιση για προσωποποιημένες συστάσεις χρησιμοποιώντας πιθανοτικά μοντέλα (probabilistic topic models) μαζί με ενισχυτική μάθηση (reinforcement learning). Χρησιμοποιούμε τον αλγόριθμο πιθανοτικού συμπερασμού Latent Dirichlet Allocation (LDA), με εφαρμογή σε βιβλία Choose Your Own Adventure με απώτερο σκοπό να ανακαλυφθούν οι θεματολογίες που πραγματεύονται. Ο βασικός πυλώνας του διαχειριστή δράματος που προτείνουμε, είναι η μοντελοποίηση του χρήστη ώς ένα αναπτυσσόμενο (στον χρόνο) κείμενο, το οποίο αποτελείται από διάφορα θέματα σε διαφορετικά ποσοστά, και το οποίο ανανεώνεται κάθε φορά που ο χρήστης "καταναλώνει" ένα αντικείμενο που του έχει συσταθεί. Ακόμα, εμπνευσμένοι από τον τομέα της ενισχυτικής μάθησης προτείνουμε την χρήση μεταβλητού ρυθμού μάθησης (learning rate) σε διαχειριστές δράματος. Η τεχνική για μεταβλητό ρυθμό μάθησης που χρησιμοποιούμε βασίζεται στον γνωστό αλγόριθμο ενισχυτικής μάθησης "Κέρδισε ή Μάθε Γρήγορα" (Win or Learn Fast). Ο ρυθμός μάθησης είναι συνδεδεμένος με την ανανέωση του μοντέλου χρήστη. Ο αλγόριθμός μας εκπαιδεύτηκε χρησιμοποιώντας όλα τα άρθρα της Wikipedia που περιλαμβάνουν περιγραφές βιβλίων. Επίσης, παρέχουμε έναν ολοκληρωμένο αναλυτή κειμένου, για την σωστή αποθήκευση και επεξεργασία των κειμένων της Wikipedia. Η πειραματική αξιολόγηση της προσέγγισής μας είναι ενθαρρυντική, καθώς τα αποτελέσματά της δεικνύουν ότι ο διαχειριστής δράματος είναι ικανός να κάνει σωστές συστάσεις στον χρήστη.

Acknowledgements

First of all, I am very grateful to my advisor, Prof. Georgios Chalkiadakis for his guidance and for trusting me with this topic. I would like to also thank for the rest of the committee, Prof. Georgios Giannakakis and Prof. Michail Lagoudakis. For my family, and my close friends, as nothing would be the same without them.

Contents

List of Figures

List of Abbreviations

AI	**A**rtificial **I**ntelligence
CPU	**C**entral **P**rocessing **U**nit
CF	**C**ollaborative **F**iltering
CYOA	**C**hoose **Y**our **O**wn **A**dventure
DM	**D**rama **M**anager
Game AI	**Game A**rtificial **I**ntelligence
LDA	**L**atent **D**irichlet **A**llocation
ML	**M**achine **L**earning
NPC	**N**on **P**layer **C**haracter
PTM	**P**robabilistic **T**opic **M**odels
RAM	**R**andom **A**ccess **M**emory
RS	**R**ecommender **S**ystem

Chapter 1

Introduction

1.1 Game Artificial Intelligence

One of the subdisciplines of Artificial Intelligence (AI) and Machine Learning (ML) is the use of AI in games. The term Game AI covers a wide collection of programming and design practices dealing with creating responsive, adaptive and intelligent agents that act to bring maximum user enjoyment. Most of the research conducted on AI in games, has been about creating opponents that perform well against human players. But there is also work to be done in making the human player's play session better. AI systems should be able to infer and provide the best possible experience within the context of the game[19].

A video game is considered to have two main aspects; the game and the context. Game AI consists of the agents responsible for the actual challenges players face and the problems they may encounter, such as rules and objectives. On the other hand, context AI is referring to the agents that deal with context tasks such as motivating the player and maintaining the global structure of the plot in order for the story to remain coherent.

1.2 Interactive Narrative

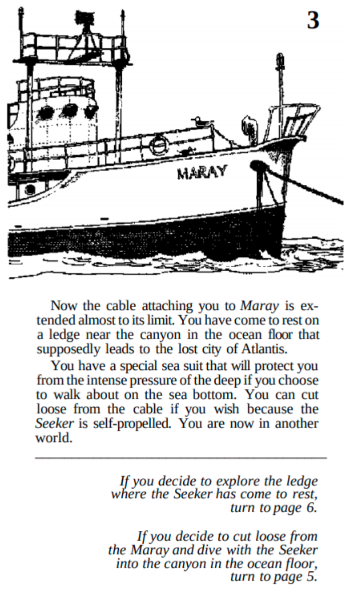
From the early beginning of human civilization the ability to present a story has been of uttermost importance. We humans use narratives to communicate, entertain, teach, etc. In [15], Prince defines a narrative as:

Definition 1 *A narrative is a recounting of one or more real or fictitious events communicated by one or more narrators to one or more narratees.*

In [17], Riedl and Butilko define interactive narrative as:

Definition 2 *(Interactive Narrative). Interactive narrative is a form of digital interactive experience in which users create or influence a dramatic storyline through actions, either by assuming the role of a character in a fictional virtual world, issuing commands to computer-controlled characters, or directly manipulating the fictional world state.*

Interactive personalized story generation, a subdomain of context AI, is achieved by letting a user make meaningful decisions about the fate of characters. An omniscient



3

Now the cable attaching you to *Maray* is extended almost to its limit. You have come to rest on a ledge near the canyon in the ocean floor that supposedly leads to the lost city of Atlantis.

You have a special sea suit that will protect you from the intense pressure of the deep if you choose to walk about on the sea bottom. You can cut loose from the cable if you wish because the *Seeker* is self-propelled. You are now in another world.

If you decide to explore the ledge where the Seeker has come to rest, turn to page 6.

If you decide to cut loose from the Maray and dive with the Seeker into the canyon in the ocean floor, turn to page 5.

FIGURE 1.1: Example of a path choice from a CYOA book (Journey Under The Sea)

background agent frequently called *drama manager* (DM) is responsible for capturing data about the user's decisions. Depending on the system's implementation, the drama manager may provide recommendations about how the user should proceed, or manipulate the non-player characters (NPC) to act accordingly to drive the game to a prefered state of the story space. So, drama managers monitor the virtual world in which the user is immersed and act to determine what happens next in the player's story experience, often coordinating and/or instructing virtual characters[1].

A narrative can be decomposed into a finite set of discrete blocks of the story, called *plot points*. When specific plot points which are coherent with each other are ordered in a sequence, they create a story. There are two fundamental types of narrative: *linear narrative* and *branching narrative*. Linear narrative is the most common form of narration. In linear narrative, all the plot points of a story are sequenced from beginning to ending without the possibility of a user alternating the way the story unfolds. Many books or computer games employ linear narratives. Every player experiences *the same* story. On the other hand, a branching narrative offers the player the ability influence the way the story progresses. At predefined points in the narrative, the user's behavior can alter the plot points that are about to follow.

As Young wonders in [31], *What structures from AI research can most readily accommodate representations of narrative?*

In our work, we use *probabilistic topic modeling* and *reinforcement learning*, in order to address the sequential recommendation problem, and thus influence the narrative a user experiences. The sequential selection problem is about providing recommendations that are dependent on the sequence of prior recommendations. The probabilistic

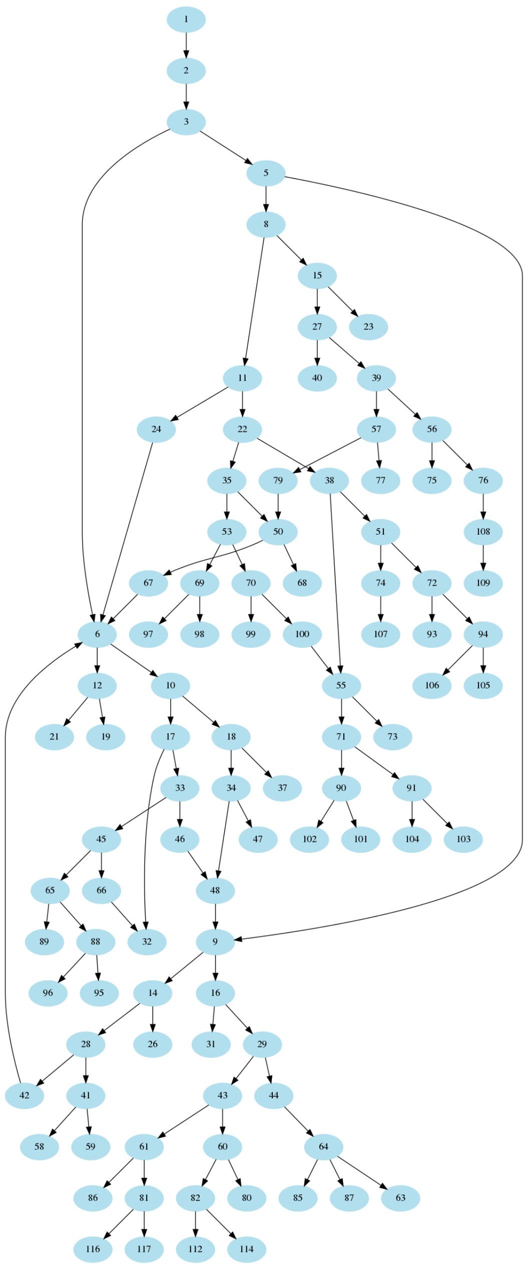


FIGURE 1.2: Example of a branching story graph from a CYOA book (Journey Under The Sea)

topic model used in this thesis is the *Latent Dirichlet Allocation* algorithm. In story-based computer games and also other applications such as education systems, a narrative is used to motivate the user's activity and to create a sense of causal continuity across a series of challenges[21].

In our work, we use *Choose-your-own-adventure* (CYOA), a well-known series of children's gamebooks. The stories are written from a second-person point of view, with the user assuming the role of the protagonist, ultimately deciding the characters' fate and the overall plot outcome. In the end of each page of CYOA book, there are several options the user has to choose from (usually there are two alternatives offered). While the characters are dealing with a situation, the narration stops, and the author lists the available options. An example is shown in figure 1.1, where the user must decide whether to continue reading at page 6 or page 5. The reader chooses the preferred action, and the story unfolds. The full stories contained in CYOA books are few, and also small, as those books are designed for mostly young ages.

The branching narrative can be represented by a branching story graph, that is, a graph in which each node represents a plot point. Any path in this graph, starting from the root node and ending to a sink node (a node with no outgoing edges, with out-degree of 0) is a complete story experience. Figure 1.2 shows the branching story graph representation of a CYOA book - Journey Under The Sea. Each node represents a page in the book. In general, the branching story graph can be authored by a human designer or by an intelligent process or through a collaborative process such as crowdsourcing. In this thesis we focus on the design of the DM, and not on the crafting of complete story graphs. That is why we use CYOA books throughout this thesis. Even though we study the application of our drama manager in CYOA books, the system can adapt to any secret path book. The same techniques can also be applied to any interactive narrative system with preauthored plot points. Thus, the DM can be extended to applications in video games, given descriptive information about each plot point of a game.

1.3 Thesis Contributions

We provide a recommendations-based approach for a *drama manager (DM)* that uses player modeling to personalize the user's story according to her preferences. A recommender system's task in this domain is to make informed guesses as to which story path the user must follow, in order for her to experience maximum enjoyment. We employ topic modelling, one of the most powerful techniques in text mining. Specifically, we use Latent Dirichlet Allocation (LDA), perhaps the most popular algorithm in the topic modeling field, as the key element of the user modeling performed by our DM. Because the LDA algorithm is trained using Wikipedia articles referring to books, we created an efficient parser to only obtain the required articles. Also, we provide some useful methods for processing of digital books. A command line utility is created to split possible merged words due to bad formatting of pdf files. Also, a method is proposed for automatic construction of the branching story graph by using the textual information.

Our DM is capable of providing recommendations regardless of the number of users (i.e., we do not rely on CF for user modeling, like [33] does). We introduced a novel distance metric for DM agents, which is the *expected utility*. Our DM calculates the expected utility of the user for each possible story reachable by the current plot point. We do that using her current user model as an approximation the actual user model, to identify the optimal path the user should follow. The recommended plot point is the one belonging to the story yielding the maximum expected utility. Finally, we evaluate our story generation system using simulated books and simulated users. Whenever the story branches our DM will inform the user which path she is more likely to get engaged to, by exploiting ratings the user provides after every recommendation. Nevertheless, our DM is *non-intrusive*, meaning that the user can disregard the DM's recommendations and choose an alternative option. Our DM adapts into the new situation, and keeps recommending the best choice. Although the system we present is quite simple, it represents one of the fundamental principles of *drama management*, that is discovering new storylines based on a pre-authored library of legal stories. Our DM yields positive results, even though the LDA algorithm is a generic topic modelling algorithm, and does not capture sequential information. As we discuss in Chapter 6, a custom probabilistic topic model can be built, to capture the sequential nature of the CYOA books, and yield even better results.

This work also serves as a framework for similar research, showing how the training data was gathered, providing basic functionality for the preprocessing, the creation of the necessary data structures and the distance metrics used and so on.

Our DM is a novel one, as it is the only one as far as we know, that uses probabilistic topic models for plot point modeling and user model updating. Experimental results show that if there is a path in the story space that the user likes (i.e., a path populated with plot points that the user assigns high ratings), our DM is capable of finding and recommending that path to the user, plot point by plot point. We also show, that even if there is a path in the story space, that is not completely aligned with the user preferences (i.e., with a maximum rating of 5, the path is populated with plot points that the user rates half of them with a rating of 4, and the remaining half with a rating of 5), the DM still manages in most occasions, to recommend a highly-rated path. Finally in D we outline some first steps towards an attempt to account for major user preference shifts. Though this effort is still immature, it lays the ground for further research to that end.

The remainder of this thesis is structures as follows. In chapter 2, we provide some background knowledge needed to understand our DM; in chapter 3, we investigate some related work in the field of DMs. Later in 4 we discuss our approach, starting from the data gathering and preprocessing, moving on to the optimal topic number identification for the LDA algorithm, and finally describing the inner workings of our DM. In chapter 5 we discuss the experimental results, and in chapter 6 we discuss future work.

Chapter 2

Background

In this chapter, we provide background on the key concepts used throughout the thesis. Even though the ideas presented bellow should be sufficient for the reader to grasp the idea, we refer interested readers to [34] and [16]. Also, for a survey on topic modeling algorithms, see [3].

2.1 Probabilistic Topic Modeling

We will briefly discuss the basics of Probabilistic Topic Modeling (PTM). This work is highly motivated and inspired by PTMs. They are a type of statistical model, that discovers latent abstract topics that occur in collections of documents. Hence, they are considered unsupervised learning algorithms.

In text analysis, a corpus is a collection of documents, and documents are collections of words. Topic models are based in the idea that documents are composed by multiple topics. PTMs are capable of discovering with great accuracy the topics that a document discusses. For example in Figure 2.1 the article's title is "Seeking Life's Bare (Genetic) Necessities". It speaks about the application of data analysis to determine the number of genes needed by an organism to survive. After analysis, a topic modelling algorithm will identify the proportions of k_1% about genetics and k_2% about data analysis and so forth. In the following paragraphs, we will discuss the logic behind topic models, as well as their key assumption, the "Bag of Words" paradigm. We focus in the algorithm used in this thesis, Latend Dirichlet Allocation.

Given a corpus of documents, our ultimate goal is to infer the knowlegde contained in this corpus. In order to be able to extract information from a corpus, topic models must be aware of the data representation. A commonly used representation is the 'Bag of Words' (BoW) model. In this data representation, the exact location of each word in a document d_i does not matter [8]. This means that all permutations of a document will yield the same result. This interprets that given several instances of a problem, each instance is referred to as bag, whereas its variables are the words. The above assumption actually means that the model only "remembers" the number of times each word appears in a document. In general, we use this model to describe a problem where a set of discrete data compose a cohesive structure. This representation is based on a fundamental statistical assumption, exchangeability [14], which allows the order of random variables to be neglected by the specific model.

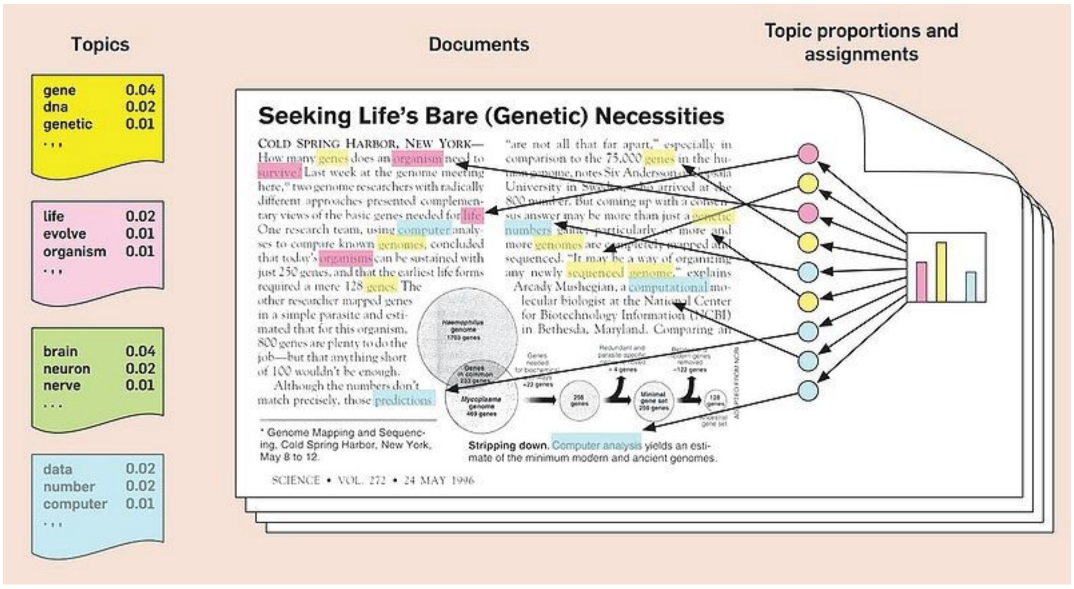


FIGURE 2.1: Example of the intuition of Latent Dirichlet Allocation [3]

Definition 3 *(Exchangeability). A finite set of random variables $z_1, ..., z_n$ is said to be exchangeable if the joint distribution is invariant to permutation. If π is a permutation of the integers from 1 to N:*

$$p(z_1, ..., z_n) = p(z_{\pi(1)}, ..., z_{\pi(N)}) \tag{2.1}$$

The BoW paradigm, works under the assumption that exchangeability applies to both documents and words. This assumption is not a realistic one, but it turns out that it works. In Figure 2.1, by just changing the order of the words in this document, the topic model would assign the same topic to this document, which is, the topic of genetics. In a similar way, we consider that documents are also exchangeable.

2.1.1 Latent Dirichlet Allocation

This section describes the bacis notion of the LDA algorithm [4]. The LDA model has been a strong foundational model, forming a basis for various other topic models. The topic models Latent Semantic Indexing (LSI) [6] and the later Probabilistic Latent Semantic Indexing (pLSI) [7], created the canvas and inspired [4] with the simplest topic model, LDA.

LDA's intuition relies on fact the documents are composed by a variety of topics. We define a topic as a distribution of some words sampled by a fixed vocabulary. Each topic though, needs a human annotator to come up with a correct description. If we consider that a corpus of documents is accurately described by K topics, then each document is composed by different proportions of those topics. For example, a document d is described by $k^{th} \in \{1, ..., K\}$ topic in a proportion p_k%. This assumption makes sense, and comes natural. If a human is asked to read any document, she will identify with ease the topics portrayed in this document. In LDA, the topics are modeled after hidden-variable models.

For example, let us consider that we apply LDA on the article in Figure 2.1. LDA would produce the topics seen at the left side of the figure. A human annotator can easily infer that the first topic is probably referring to genetics, the second topic to evolutionary biology and so on. On the right side of the figure, we see a plot of a discrete probability distribution. This probability distribution encodes the percentage that corresponds to each topic of the document. Concluding the process, we know that this article is a mixture of data analysis, genetics evolutionary biology in different proportions.

Notation

First of all, we need to establish the notation to describe the generative process of the LDA algorithm. Even though the LDA algorithm is not applicable only in the text processing domain, we will follow the notation used by [4]:

- A *word* is the basic unit of discrete data, which is defined as an item from a vocabulary that is indexed by $\{1, ..., V\}$.
- A *document* is an sequence of N words denoted by $w = (w_1, ..., w_N)$, in which w_n denotes the n^{th} word in order.
- A *corpus* is a collection of M documents denoted by $D = \{\boldsymbol{w_1}, ..., \boldsymbol{w_M}\}$

For example, in the article of Figure 2.1 we observe that there are three main topics captured by the algorithm. The genetics topic shown with yellow, the evolutionary biology topic shown with pink and the data analysis topic shown with blue. Each one of those three topics, includes words about the respective concept with high probability.

The Generative Process

So far, we have defined the necessary notation and intuition of topic models and particularly the LDA algorithm. In what follows, we will describe the *generative process* of the model, in order to try to imitate the way the observed data was produced. It is impossible for a PTM algorithm model to infer exactly the way the data was created, but it nevertheless tries to approximate it. By following the process described below, we model the data in order to discover the latent topics. Let K, as mentioned, denote the number of topics we want to discover. The LDA algorithm takes the topic number K as input.

LDA assumes the following generative process:

- For each topic z, where $z \in \{1, ..., K\}$:
  - Draw a distribution over the words of the vocabulary, $\phi_z \sim Dir(\beta)$.
- For each document $\boldsymbol{w_i}$ where $i \in \{1, ..., M\}$ in a corpus D:

- Choose $\theta_i \sim Dir(\alpha)$.
- For each w_n the n^{th} word in i^{th} document, where $n \in \{1, ..., N\}$
  * Choose a topic $z_n \sim Multinomial(\theta_i)$.
  * Choose a word $w_n \sim Multinomial(\phi_z)$, a multinomial probability conditioned on the topic z_n.

We define some notation to be used in the following sections and chapters. Each topic z is a distribution over a vocabulary, denoted by V. We also denote this distribution, ϕ_z which is a Dirichlet distribution with parameter β. β is a KxV matrix, whose elements are populated by $\beta_{i,j} = p(w^j = 1|z^i = 1)$. θ_{d_i}, a K-dimensional Dirichlet random variable, encodes the topic proportions for document d_i. The topic proportion variable θ is sampled from a Dirichlet distribution, with parameter α, also a K-vector. For $alpha, (\alpha_j > 0, forj = 1, ..., K)$. We denote with z_n the topic corresponding to the n^{th} word, while z is a N dimensional vector which defines which topic each of the words of the vocabulary belongs to. Following the notation in [4], equation 2.2 provides the joint distribution of a topic mixture θ.

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{2.2}$$

where α, β are the model parameters for a set of N topic z and a set of N words w. By integrating over θ and summing over z:

$$p(w|\alpha, \beta) = \int p(\theta|\alpha) \Bigg(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n, |z_n, \beta) \Bigg) d\theta \tag{2.3}$$

while for all documents, the probability of a corpus occurring given the model parameters is:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \Bigg(\prod_{n=1}^{N_d} \sum_{z_{d_n}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \Bigg) d\theta_d \tag{2.4}$$

The LDA algorithm represented in plate notation is shown in Figure 2.2, where the grey node corresponds to the observed variable, while the white nodes represent the hidden variables which we need to infer. The plates represent variables that repeat in a graphical model. As shown in the graphical model, the parameters α and β are related with the structure of the corpus. The variable θ_d is generated once for each document, and z_{dn} and w_{dn} are generated for each of the words in a document. This can is shown in the graphical model by the N plate and the D plate.

Posterior Inference

In general, the goal of the LDA algorithm is to discover the latent topics from a corpus. The documents are treated as observed data, or evidence, while the topics, the distribution of topics per document and the topic term distributions are treated as hidden

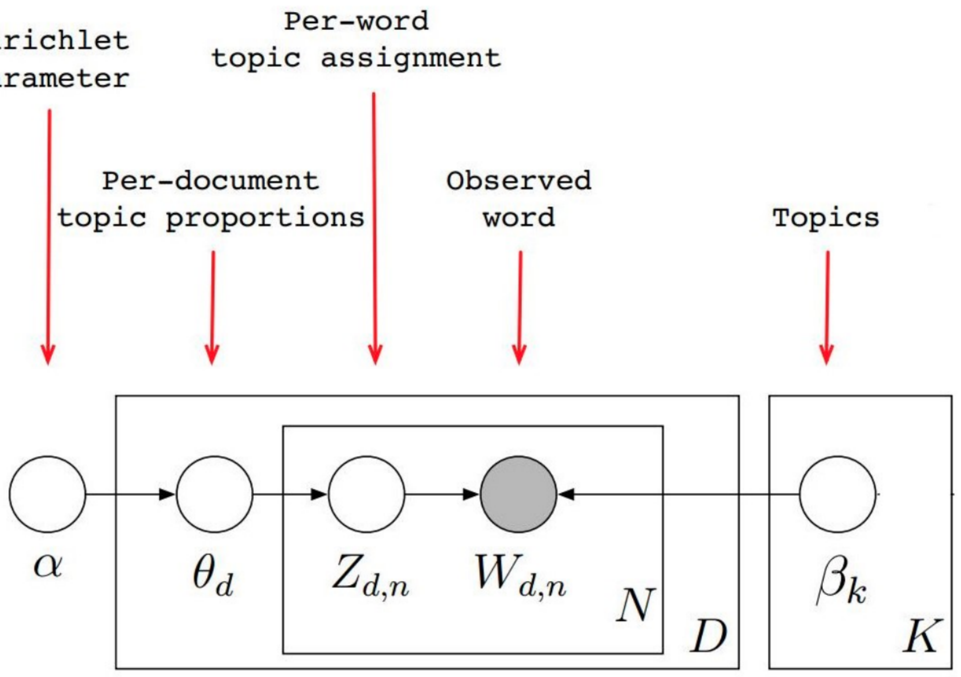


FIGURE 2.2: Probabilistic Graphical Model of Latent Dirichlet Allocation [12]

variables. The computation problem that arises is to find a computationally efficient way of approximating the hidden variables. This can be achieved by reversing the generative process. The process of inferring distributions includes learning:

- the set of topics
- their associated word probabilities
- the topic of each word
- the particular topic mixture of each document

This is a problem of Bayesian Inference. The following equation defines the posterior distribution of the hidden variables given the evidence (a document):

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(z|\alpha, \beta)} \tag{2.5}$$

The equation 2.5 is intractable to compute, but there are available methods to approximate it. Widely used approximations fall into two categories; sampling based algorithms and variational algorithms. Sampling based algorithms collect samples from the posterior to try to approximate the equation with an empirical distribution. A commonly used algorithm is *Gibbs sampling*. Variational methods instead of approximating the posterior, try to find a family of distributions that best fits the observed data. Thus, the approximation problem transforms into an optimization problem. For more details, we refer interested users to [4].

2.2 Learning Rate

In the ML literature many algorithms have been suggested for the update of a *learning rate*. The learning rate parameter appears in various key areas of ML and AI, like in neural networks, reinforcement learning, and others. Wherever some form of learning rate is used, it represents a *tuning parameter* in an optimization algorithm. That parameter determines the step size according to which the algorithm moves toward a minimum defined by the loss function. This often suggests some iterative process with an update of a subset of the parameters [35]. Intuitively, the learning rate determines the step size and hence how quickly the search converges [30].

Most algorithms that use a given learning rate parameter depend on the system designer to manually choose it, given a specific application. To combat this issue, many techniques have developed that automatically change the learning rate parameter, based on the performance according to a metric set by the practitioner.

We focus in a hill-climbing technique named WoLF ("Win or Learn Fast") [5]. The essence of this technique is that a learning agent must learn and adapt quickly when it is performing worse than expected. On the other hand, it must make small changes in the learning parameters, when performing better than expected. Due to the sequential nature of the problem we are aiming to solve, the updates to the learning rate are made considering the relation between the overall performance and the last k recommendations. We utilize two variations for the learning rate, δ_{win} for the case the agent is performing better and δ_{lose} for the case it is performing worse [28].

The variable learning rate used in our work, behaves in the following way:

$$\Delta_{t+1} = \begin{cases} min(\Delta_t + \delta_{lose}, \Delta_{max}) & Losing \\ max(\Delta_t - \delta_{win}, \Delta_{min}) & Winning \end{cases} \tag{2.6}$$

where, Δ_{max} and Δ_{min} are predefined parameters indicating the maximum and minimum value of the learning rate. The conditions *Losing* and *Winning* are defined in the algorithm 1, in chapter 4.

Chapter 3

Related Work

In general, a lot of research has been conducted to construct a successful Drama Manager, i.e., a DM that selects plot points that enhance user experience.

Drama manager agents have been widely applied to the interactive storytelling domain, assuming the role of human designers. There are two approaches to drama management, search based drama management [29][11][23] and declarative optimization based drama management [2][20]. The two aforementioned approaches transform the plot selection problem into a search problem where the DM searches for the next plot points based on an evaluation function set by the human designer. This technique, even though it is dynamic, only responds to player actions in a way partially or completely conceived by a human designer. This evaluation function does not take into account the evolving preferences of a user.

Riedl and Young have worked on the correlation between planning based narrative mediation system and the branching story graph [32][18]. The narrative mediation system, when needed, is capable of generating narratives that preserve the storyline. During the story, the user can interact with the virtual world (i.e., computer controlled agents). If the system detects that the actions of the user are such that the story starts to deviate from the originally planned one, it generates alternative stories from the point of the deviation. They also show, that their system can work with branching story graphs as well, because any acyclic branching story graph can be transformed into a branching story tree which can be transformed into a mediation tree. Figures 3.1, 3.2 and 3.3 show the graphs used in the narrative mediation system. Their approach acts to maintain coherence. They do not mention how the optimal linear story is generated. Also, their work focuses on real-time adaptation of the story in order to maintain coherency, whereas in our work, we focus on the optimal plot point recommendation, from a pool of preauthored plot points. Hence in our work, we do not deal with story coherency, as every story generated is coherent. Our goal is to recommend to each user, the story she will be more likely to like.

The PaSSAGE (Player-Specific Stories via Automatically Generated Events) system [27] observes the player's actions and extracts his preferences through observations. Using the extracted model, the DM dynamically selects the branches of a CYOA style story graph. PaSSAGE uses Robin's Laws five game player types: Fighters, Power Gamers, Tacticians, Storyteller, and Method Actors. The user model is constructed by using each of the five dimensions as the strength of each type. As the player interacts with the game and the user model is updated by the DM, dimensions are updated in accordance to the five game player definitions. Peinado and Gervas [13] also use the

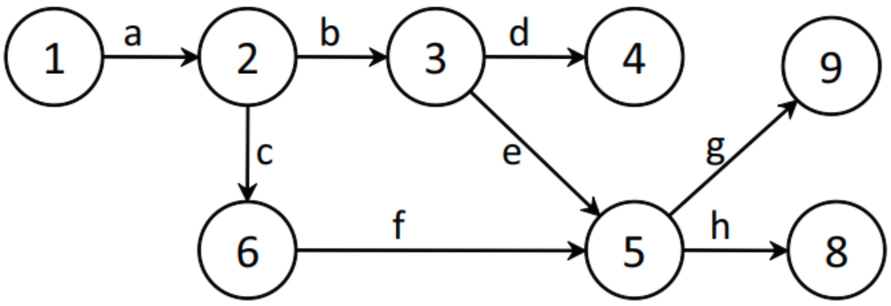


FIGURE 3.1: A sample branching story graph

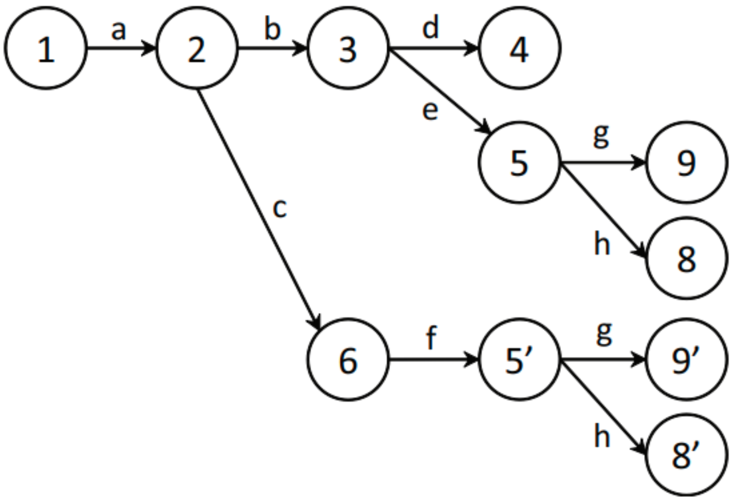


FIGURE 3.2: The branching story tree representation for the branching story graph in Figure 3.1

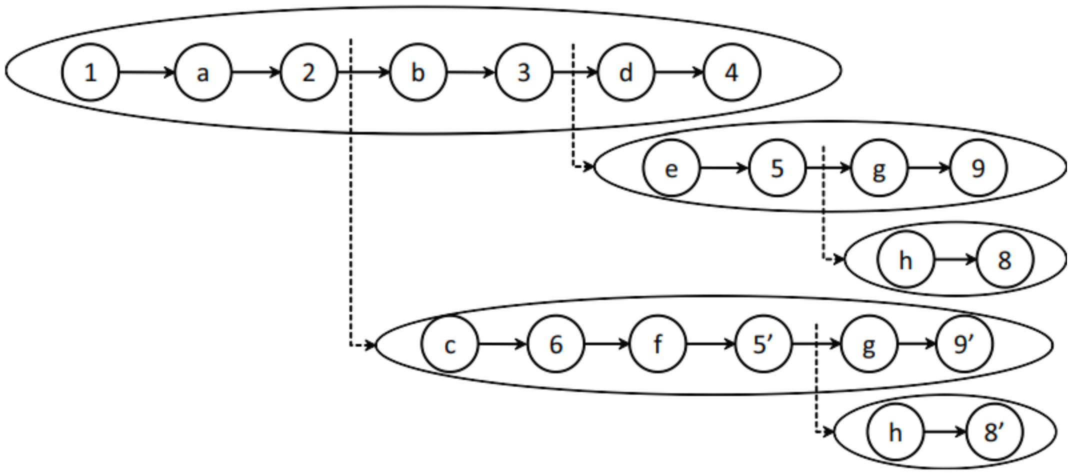


FIGURE 3.3: The narrative mediation tree for the branching story tree of Figure 3.2

same player types as PaSSAGE. They use a knowledge intensive case based reasoning approach to generate interactive stories, based on the game state and the user model. Seif El-Nasr [10] created Mirage [9], which uses a four-dimension player model: heroism, violence, selfinterestedness, and cowardice. Mirage uses a preauthored rule-based system to associate player behaviors to the four dimensional models.

All those methods create player models that can only be classified according to a few discrete play styles. These systems also assume that the predefined player types are sufficient for any choice encountered. Also, none of those methods employ probabilistic topic modelling techniques to tackle the interactive story generation, and the sequential recommendation problem.

This thesis is inspired by the work of Tripolitakis & Chalkiadakis [28], who employed topic modelling and reinforcement learning for movie recommendations and [33], by Yu & Riedl who implemented a DM which uses collaborative filtering to suggest the next plot point on CYOA books. The beginning of 4, makes more clear how these two aforementioned papers helped shape our work in this thesis.

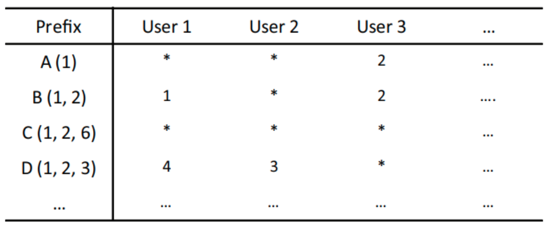
Chapter 4

Our Approach

In this chapter, we describe our recommender DM agent. Its main component is a data-driven player modeling algorithm that predicts the user's preferences over successive plot points and offers recommendations as to which choice the user should make. The chapter is structured as follows. First, in Section 4.1 we examine the work of [28] and [33], and how those two aforementioned papers gave us the concepts this thesis studies; then, in Section 4.2 we discuss about the dataset download while in 4.3 we show the preprocessing methods used. In Section 4.4 we explain how we decided the optimal topic number K of our LDA model. In Section 4.5 we show how the plot point model and the user model correlate, and how the user model update is performed. In Section 4.6 we show the prefix tree which is the main data structure that our DM uses. In Section 4.7 we discuss the recommendation phase of our DM. Finally, in section 4.8 we show our DM's user interface.

4.1 Problem Definition

In [33], Yu and Riedl create a recommender system (RS) for branching narrative, which uses a collaborative filtering (CF) approach. They introduced a prefix based collaborative filtering algorithm based on the input given by the users. In this way, they address the sequential selection problem, i.e., the problem of recommending items that are dependent on the sequence of prior recommendations. In their work items under recommendation are plot points. In this occasion, a plot point is the sub-story between



Prefix	User 1	User 2	User 3	...
A (1)	*	*	2	...
B (1, 2)	1	*	2	....
C (1, 2, 6)	*	*	*	...
D (1, 2, 3)	4	3	*	...
...	...	...	...	...

FIGURE 4.1: Illustration of the prefix-rating matrix. A, B, C and D represent the prefixes. The larger the digital number, the higher the preference. The stars represent those missing ratings[33]

two consecutive story branching points; the branching point that led to this plot point, and the branching point that lies ahead. Exceptions to that definition are the starting plot point, and the ending plot points as one branching point is replaced by the starting and the ending of the book. In CYOA books, a plot point is usually a page of the book, respectively. For example, in Figure 1.2, page 6 is a plot point as is lies between the branching of plot point 3 (choice between 6 and 5) and the branch leading to either page 12 or 10.

Figure 4.1 illustrates a simple prefix-user matrix. At the *Prefix* column we can see some of the prefixes, which contain all the plot points until a specific point. For example, prefix A(1) is a substory containing plot point 1, prefix B(1,2) is a sub-story containing plot points 1 and 2, and so on. The main goal of Yu and Riedl is to guess the missing values of the matrix. To this end, they employ CF. CF is a family of algorithms that try to detect users' rating patterns. By using those patterns, they can make predictions of new user's ratings based on similar users. They test two CF learning algorithms: probabilistic Principal Component Analysis and Non-negative Matrix Factorization. Specifically, they model the users as five dimensional vectors, with each dimension representing a specific player type (fighters, power gamers, tactitians, storytellers and method actors). Each entry of a vector ranges from 0 to 1. This approach, while yielding interesting results, only works when there are multiple users that use the system. Before the prefix-rating matrix is sufficiently populated with data, their algorithm will perform poorly. CF algorithms are known to be prone to the notorious *cold start* problem. Cold start is a problem in recommender systems, stating that recommendations are of very low quality until the system gathers sufficient information. Also, another disadvantage of their method, is that the plot points are labeled by humans. The method we propose automatically assigns plot point descriptions based on the LDA model.

In [28], Tripolitakis and Chalkiadakis create a movie recommender system by employing probabilistic topic modeling intertwined with ideas taken from the field of reinforcement learning. They model *both* the user and the items as mixtures of latent topics following a distribution with Dirichlet priors; this can be achieved by exploiting the robustness of crowdsourced information for each item. Their method is immune to the "cold start" problem, and it can also cope with changing user preferences.

This thesis comes to unite the work of the two aforementioned projects. Our work shows that a similar DM to the one in [33] can be built, which can work even if only one user uses it (i.e., we do not rely on CF for user modeling). We show that by employing probabilistic topic models we can extract meaningful information from book plot points. Inspired by [28], we also model the user and the items as mixtures of latent topics.

4.2 Getting the Data from Wikipedia

Before the application of any ML algorithm, some kind of data preprocessing must be performed. In order for LDA to capture a wide spectrum of topics, we decided to train the algorithm with Wikipedia articles describing books. Specifically, we collect information on every single book found in Wikipedia. We chose Wikipedia because it is one of the bigger online communities producing peer-reviewed, unbiased information.

4.2.1 Downloads

Wikipedia offers copies of its database for free to interested users. It provides public dumps located at various mirror sites, though the most common is *Wikimedia Downloads*[1]. Wikipedia is a dynamic environment, relying on a croudsourcing model for the constant update of available information. Because anyone can issue changes to any article at any time, the articles are constantly being updated. Subsequently, the Wikipedia data dumps must be also updated so the interested practitioners can have up-to-date information. The wikipedia dump we used in our work was that of 04/20/2020[2]. This dump contains the Wikipedia database split into 59 files, all of which were downloaded.

4.3 Preprocessing

4.3.1 Parsing the Data

Even though the 59 mentioned files that were downloaded are compressed, they occupy 34.3 GB of disk space. It is estimated that the size of the uncompressed files is about 51 GB[3]. So, uncompressing and then preprocessing is infeasible for a desktop computer. Our system processes the files by decompressing them one by one, and one line at a time, and thus rendering the processing computationally feasible. The uncompressed data is in XML format, so all available information is enclosed in XML tags. This is the key idea behind line-wise preprocessing of the downloaded files. In our case, we are only interested in the data encapsulated between *title* and *text* tags; denoted in XML format as <*tag*> (starting tag) and </*tag*> (ending tag). This means that we only need the *title*, and the *text* of each article. Every time the parser encounters the starting tag of one of these two tags, it will save characters to a buffer until it encounters the corresponding end tag. Articles are identified by *page* tags. In general, content dumps of wikipedia articles have many tags for information handling like *ns*, *id*, *revision*, *contributor*, *comment*, *model*, *format*, *text xml*, *sha1*, *parentid*, *username*, and so on. Since none of those XML tags provide us with useful information for the DM, we do not process nor store the data encapsulated in them.

Our goal is to determine which articles are about books. We use the python package *mwparserfromhell*, which is a powerful python parser for *Wikimedia Downloads* (i.e. the site we downloaded our data from). Using functionality provided by this package, we are able to filter out articles based on an attribute of Wikipedia pages: *The Infobox*. Each category of articles on Wikipedia, such as films, books, radio stations, etc has its own type of *Infobox*. Since we are designing a parser that gets all book articles, the desired

[1]`https://dumps.wikimedia.org/`

[2]`https://dumps.wikimedia.org/enwiktionary/20200420/`

[3]`https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`

Infobox template is named *Infobox book*. For every book found, we collect its title, text, and length. We gathered 39627 articles.

Now that we have gathered all the books, we continue with with some further text cleanup of the books corpus: The following functions were created:

- *text_one_liner*: Removes \n (new line) characters and replaces them with space characters.
- *real_exp_remover*: Removes all XML tags left in the texts. With the use of regular expressions, all the tags got removed. The most common tag removed is <*ref*>.
- *filler_remover*: Wikipedia articles are split in various parts called *article elements*. Each article element has a title to make in distinguishable by the other elements. Those titles do not provide useful information and are thus removed. A few examples of article elements are References, Plot, Contents, Notes, etc.
- *possessive_fixer*: In English, we add *'s* to show possession. Even though this contains information about a sentence's meaning, it is not useful to topic modelling algorithms, which we will employ to create our personalised recommender system.
- *website_remover*: Links are removed from the articles.
- *space_reducer*: Abundant space characters between words are reduced to one space character.

4.3.2 Dealing with the Nature of the Data

It is meaningful to examine the length distribution of the articles. In figures 4.2 and 4.3 (which is a zoomed version of 4.2) we observe that most of the articles are quite small. In figure 4.3, we observe how the documents are distributed as their length gets larger. By examining manually some of the smallest articles, we observed that they do not contain much meaningful information. For example, some of the smallest articles obtained are:

- *Dorothea Dreams*: "Dorothea Dreams is a 1986 novel by American author Suzy McKee Charnas. 1986 American novels."
- *I nattens tystnad*: "I nattens tystnad is a novel by Margit Sandemo. 1998 novels Novels by Margit Sandemo."
- *The Einstein Girl*: "The Einstein Girl (2009) is a novel written by Philip Sington. 2009 British novels."
- *Sarkofag*: "Sarkofag is a novel by Slovenian author Dušan Merc. It was first published in 1997. List of Slovenian novels Slovenian novels 1997 novels."

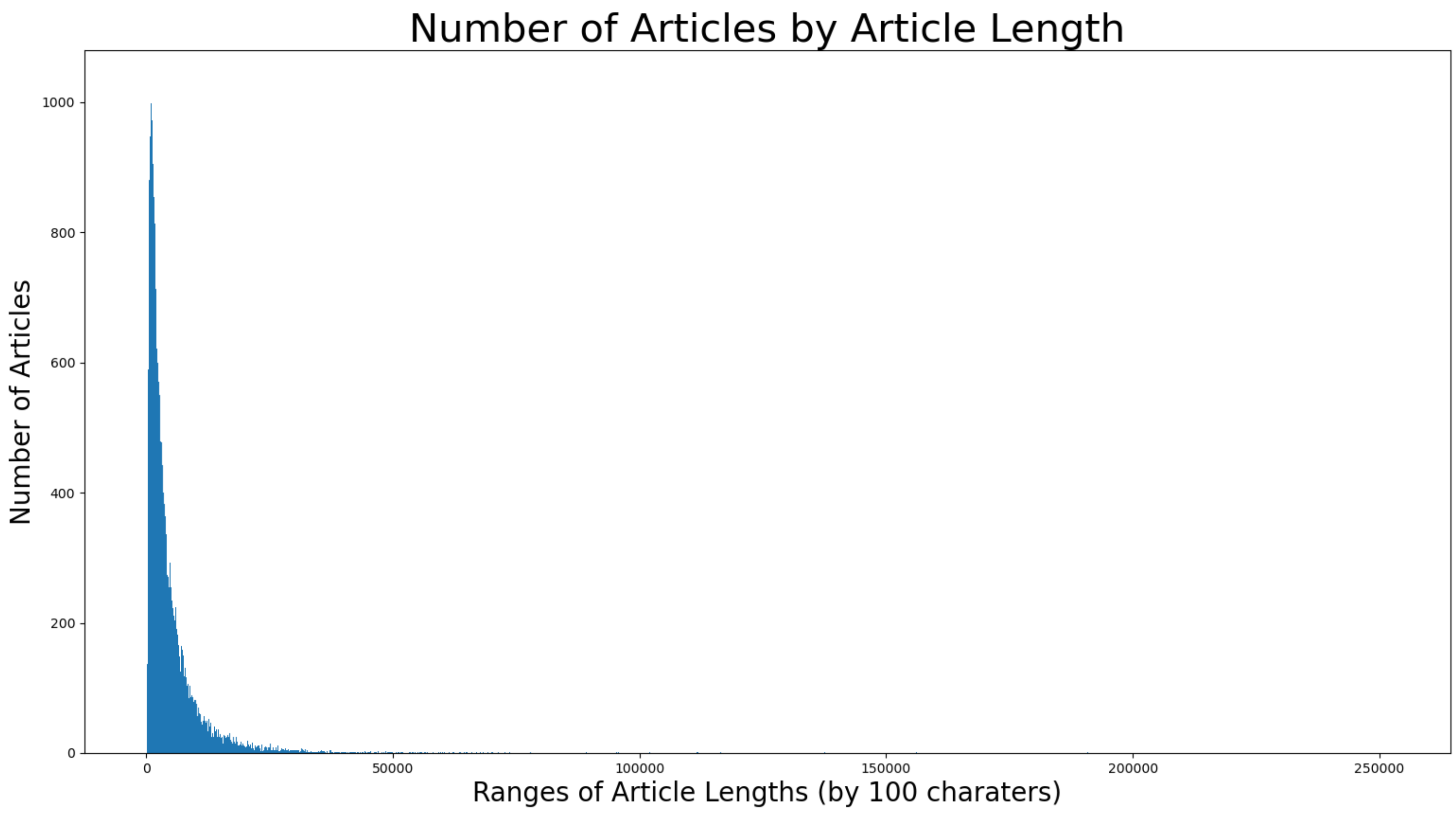


FIGURE 4.2: Graph showing the distribution of tokens per article. The lengths are rounded up to hundreds.

- *Tarot ReVisioned*: "Tarot ReVisioned is a 2003 book and Hermetic Tarot deck by Leigh McCloskey. The foreword is written by Stanislas Klossowski de Rola. 2003 non-fiction books Tarot decks."

We observe that the shortest of articles only contain information about the year the book was published, and the name of the author. This information is essentially useless to a topic modeling algorithm. We also observe some text that we did not manage to preprocess. For example in *I nattens tystnad*, there is a phrase "1998 novels Novels by Margit Sandemo" or in *The Einstein Girl*, we see "2009 British novels". This is residue text that is very hard to remove via parsing, so we leave it as is.

We sorted the articles by their length, and then examined the dataset by hand, to attempt to locate a point at which the articles begin to yield useful information about a book. After further examination, we concluded that there is not a specific article length that beyond it, the book descriptions become meaningful. So, we introduce a heuristic value represented by the variable *low* to 0.3 (30%), meaning that from the 39627 discovered articles, we omit the 11888 shortest ones. Figures 4.2, 4.2 and 4.4 show the article length distribution before we apply the pruning based on the *low* heuristic.

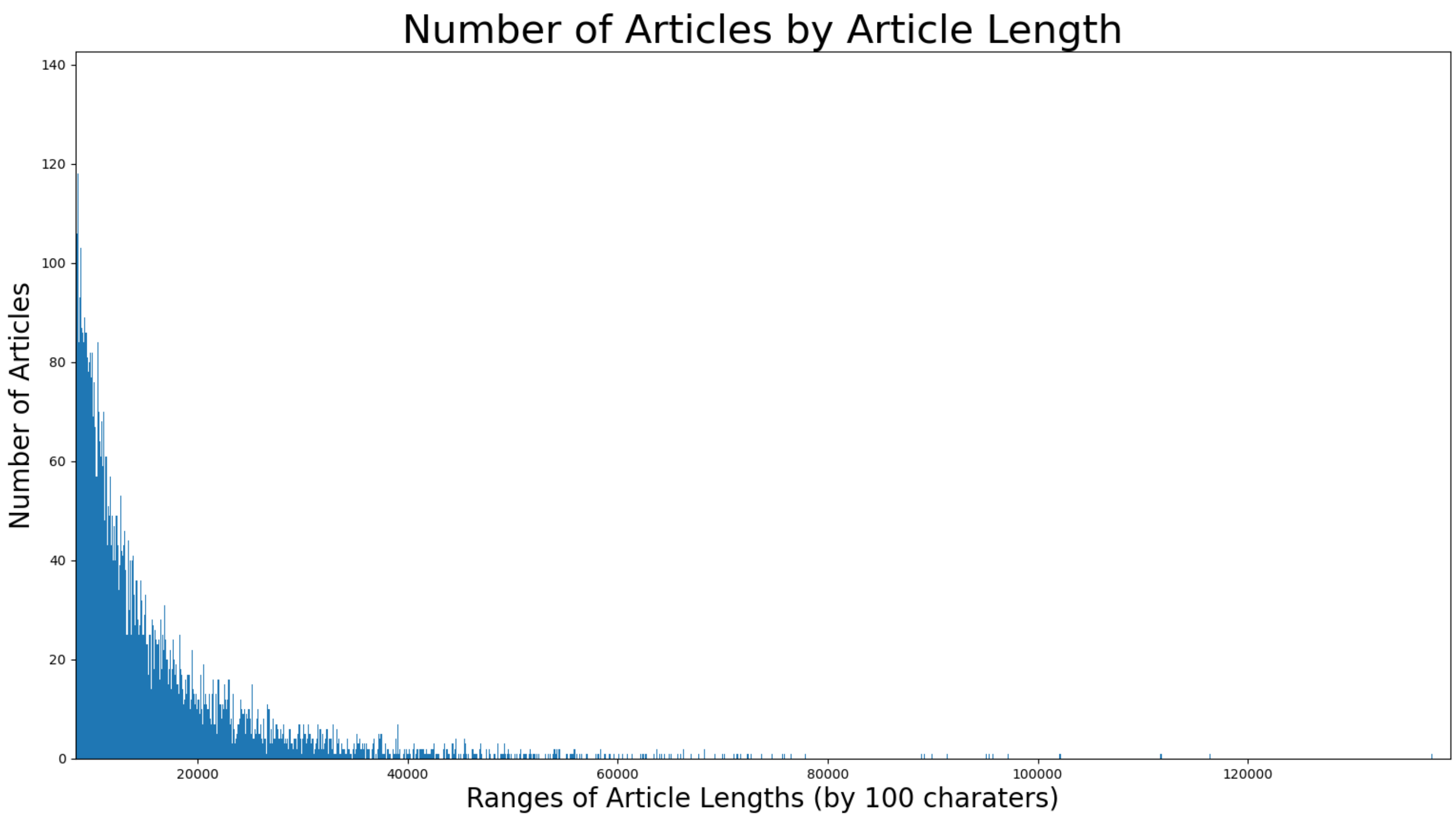


FIGURE 4.3: Graph showing the distribution of tokens per article (this is a zoomed version of 4.2). The lengths are rounded up to hundreds.

Also, from each article obtained from Wikipedia, we removed the first names, as they do not provide semantic information about a topic.

4.3.3 Towards the Bag of Words Representation

In the previous section, we dealt with various characteristics of our dataset, concerning mostly the Wikipedia's dataset format. Before applying the LDA algorithm to the dataset, we must further preprocess our data.

The next preprocessing step is the tokenization. Tokenization is the process by which a document is transformed from a sequence of sentences containing words, to a sequence of words (reffered as tokens) without punctuation. Also, the tokens are all converted to lowercase letters. The tokenization applied in our case, ignores tokens that are too short or to strong. We chose minimum length of accepted tokens to be 2, and maximum length of 15. Even though our system gives an option to perform lemmatizing and stemming, we chose to not perform those tasks in our dataset. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes

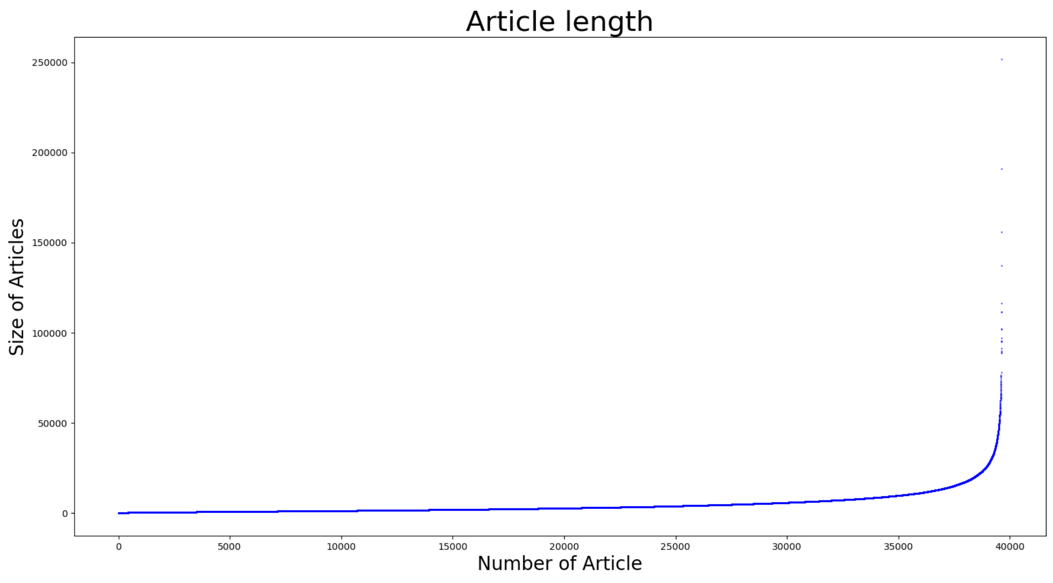


FIGURE 4.4: Increasing size of articles

derivationally related forms of a word to a common base form.[4] But as suggested by [22], in English, morphological conflation treatments such as stemmers and lemmatizers can worsen topic model quality, while LDA turns out to be quite good at combining morphological variants by itself. So, in this work, we choose to not use stemming and lemmatizing, as they pose a potential cause of damage to our model.

The final step before applying the LDA algorithm to the dataset, was to allow bigrams to exist in our tokenized lists. In computational linguistics, n-grams are contiguous sequences of n items. In our case, we allowed 2-gram sequences (which are defined as bigrams) to be counted as a token if this 2-gram sequence tends to appear a lot in the dataset.

[4]https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

4.4 Choosing the Optimal Topic Number

The disadvantage of LDA is that it does not infer the optimal number of topics. So it relies on the a-priori definition of topics by the practitioner. For determining the optimal topic number, we used three model quality metrics; *perplexity*, *topic coherence* and *Jaccard similarity*. Except for those three metrics, we also used human judgement to evaluate the interpretability of the inferred topics.

In order to get the best results out of the selected metrics, the following procedure was applied:

- We split the dataset into $k = 5$ equal sized buckets. That is, the Wikipedia corpus of $M = 27739$ crowdsourced documents was partitioned into k buckets of $M_{test} = 5548$ documents.
- For the calculation of perplexity, we iterated over the dataset k times. For each iteration, we used a different permutation of the $k - 1$ buckets for training the model, and the remaining bucket served as validation corpus.
- For the calculation of Jaccard similarity and topic coherence, the concept of evaluation over a held-out dataset does not apply. Nevertheless, we calculated the models k times by removing one bucket out of the dataset. With this procedure, we try to capture any anomalies caused by a subset of the dataset.

4.4.1 Perplexity

In the field of information theory, perplexity is a metric of how well a probability model predicts a sample. It is not useful as an absolute number, but rather in comparison of perplexities of many probability models. Low perplexity indicates that a probability distribution is good at predicting the sample. The perplexity of a probability distribution p is

$$perplexity(p) = b^{-\sum_x p(x) log_b p(x)} \tag{4.1}$$

where b is usually 2 or e, even though perplexity is independent of the base, in condition that the entropy and the exponentiation use the same base.

Perplexity is often used as an example of an intrinsic evaluation metric. It is used widely in the language modeling community, and aims to capture how unsuprised a model is of new data it has not seen before. This is commonly measured as the normalised log-likelihood of a held out test set. Perplexity is monotonically decreasing in the likelihood of the test data, and is algebraicly equivalent to the inverse of the geometric mean per-word likelihood. A lower perplexity score indicates better generalization performance [4]. Perplexity is algebraically equivalent to the inverse of the geometric mean per-word likelihood. For a test set of M documents, the perplexity is:

$$perplexity(D_{test}) = exp(-\frac{\sum_{d=1}^{M} logp(\boldsymbol{w}_d)}{\sum_{d=1}^{M} N_d}) \tag{4.2}$$

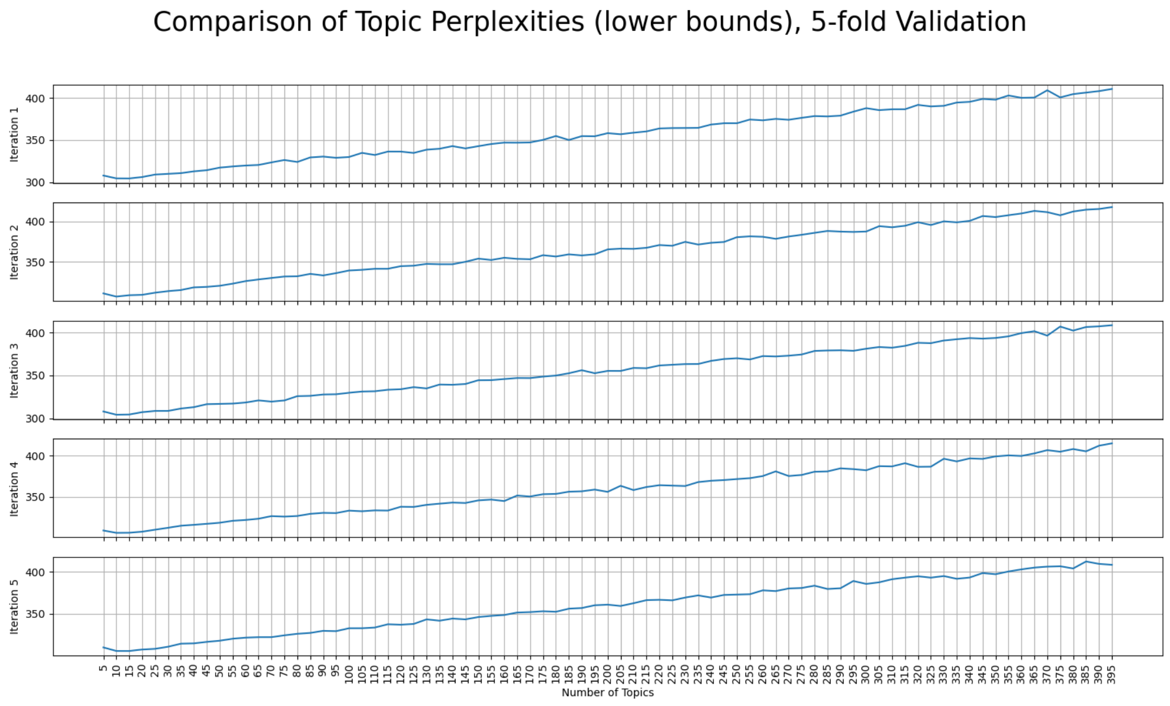


FIGURE 4.5: Perplexity values of 5 evaluations

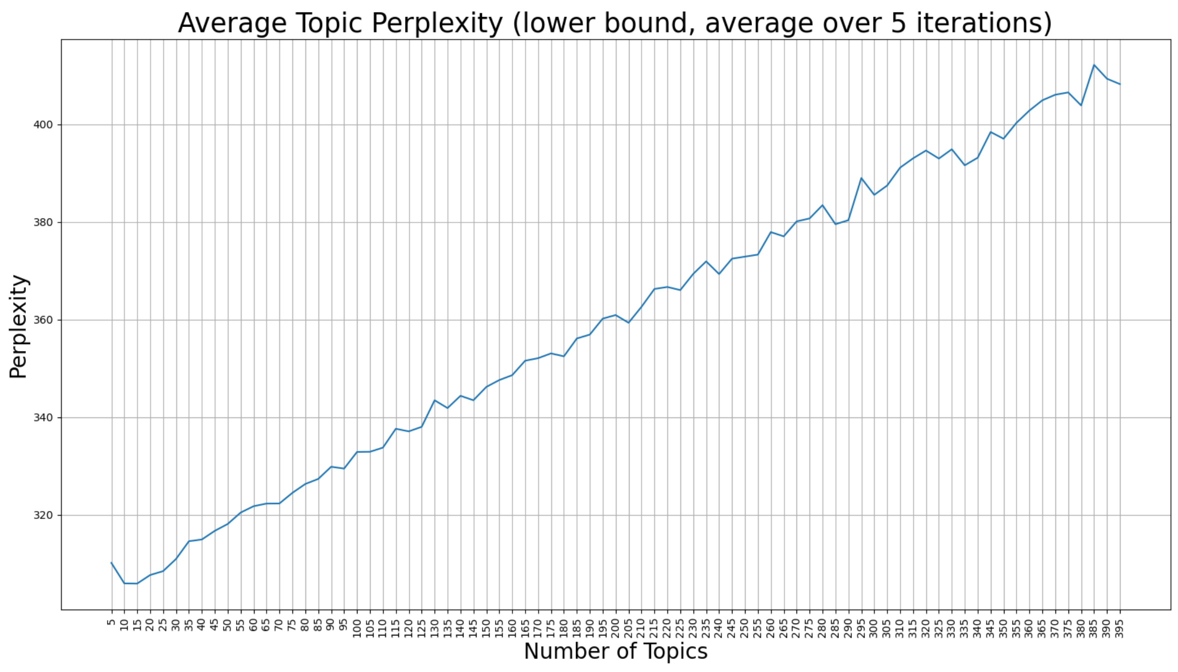


FIGURE 4.6: Average perplexity values

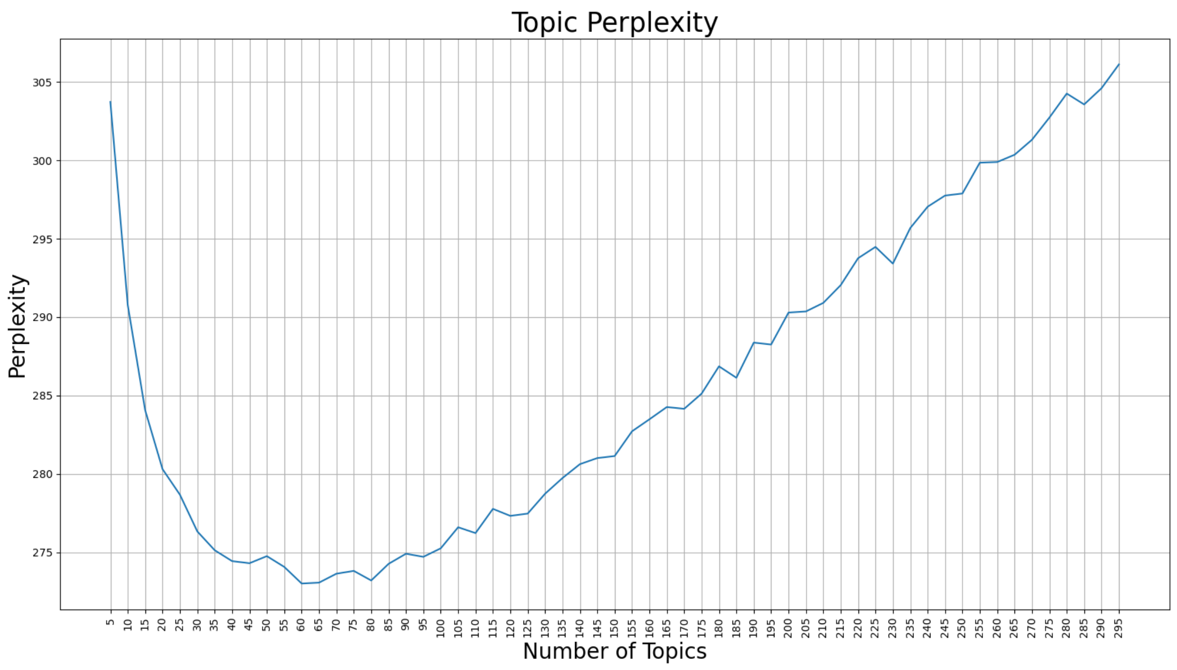


FIGURE 4.7: Perplexity values (Obtained by the training data)

In this work we used *gensim's*[5] function *log_perplexity* which computes a per-word likelihood lower bound. Although the bounds obtained are not as informative as we would hope, they do provide us some useful guidance. In Figure 4.5 we see the plots of perplexity bounds, as the number of topics increase. Those perplexity bound curve values tell us that the perplexity will not be lower than the curve. In figure 4.7, we observe that there is a perplexity minimum. Given the fact that the inferred topics have been optimized based on the training set, we observe that the perplexity upper bound obtains its minimum value for 30 to 100 topics. This is a strong indication that our model's predictive strength maximizes at this range of topics. The global minimum of the perplexity's lower bound is at $K = 60$ topics. We applied the elbow criterion to identify the optimal number of topics K. Elbow criterion is a heuristic used in determining the value of parameters in various data driven models, such as principal component analysis among other clustering algorithms. In Figure 4.7 the elbow criterion indicates that the optimal topic number K must be chosen between $K = 45$, $K = 60$, $K = 80$.

4.4.2 Topic Coherence

Topic coherence measures whether the words in a topic tend to co-occur together. It adds up a score for each distinct pair of top-ranked words. The score is the log of the probability that a document containing at least one instance of the higher-ranked

[5]https://radimrehurek.com/gensim/

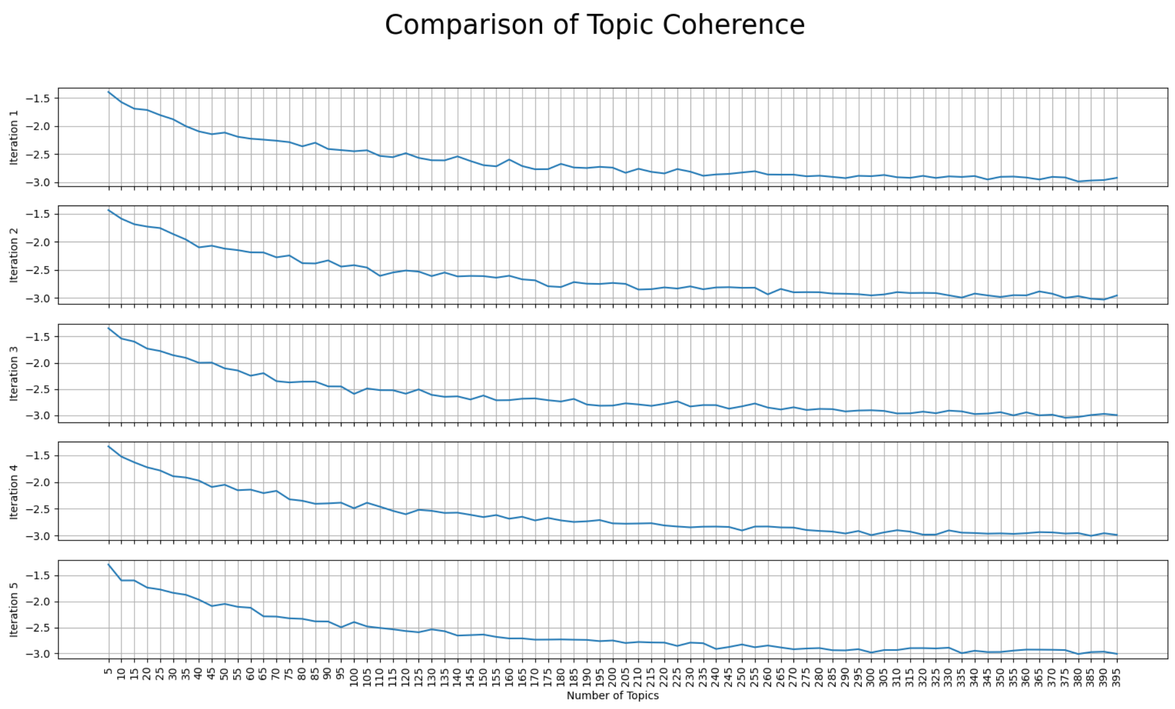


FIGURE 4.8: Topic coherence values of 5 evaluations

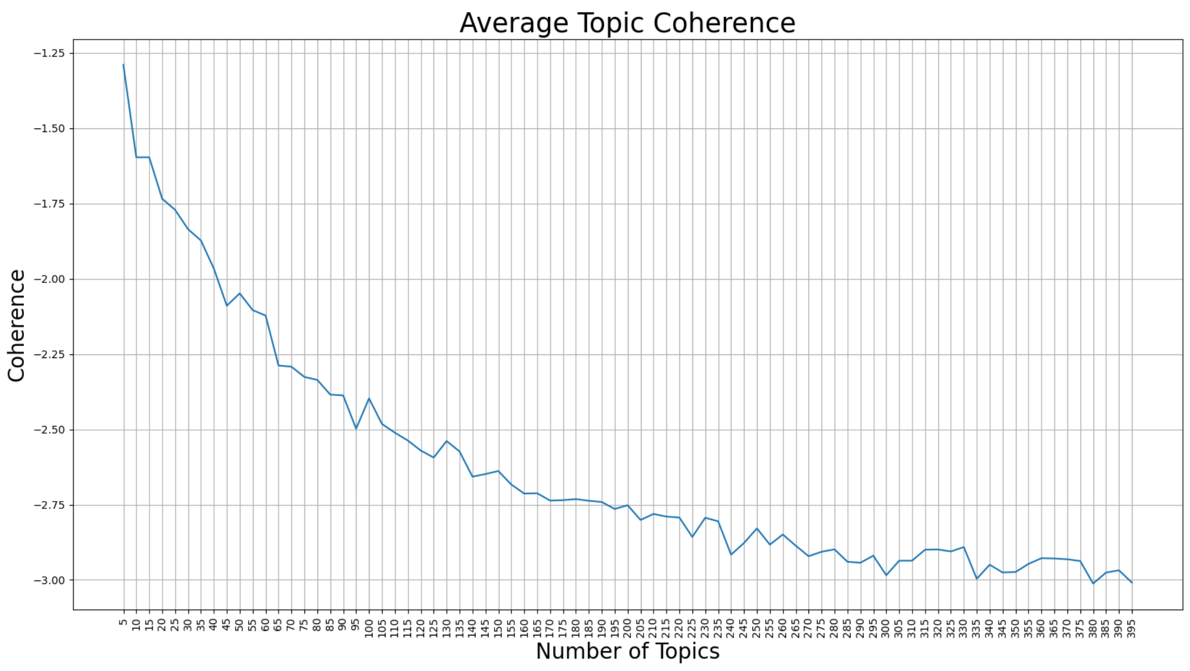


FIGURE 4.9: Average topic coherence values

word also contains at least one instance of the lower-ranked word. Simply, they score a single topic by measuring the degree of semantic similarity between high scoring words in the topic[25]. Topic coherence is one of the methods used to decide whether the inferred topics are interpretable. In simple words, we can use topic coherence to help us distinguish good and bad topics. Topic coherence measures compute the sum of pairwise distributional similarity scores over the set of topic words V. So, this generalizes as:

$$coherence(V) = \sum_{(v_i, v_j) \in V} score(v_i, v_j, \epsilon) \tag{4.3}$$

where V is the set containing the words of a topic. In this work, the coherence measure *UMass* was used:

$$score(v_i, v_j, \epsilon) = log \frac{D(v_i, v_j) + \epsilon}{D(v_j)} \tag{4.4}$$

where $D(x, y)$ counts the number of documents containing words x and y and $D(x)$ counts the number of documents containing x. Also, ϵ indicates a smoothing factor which guarantees that *score* returns real numbers. In this work, a value of $\epsilon = 1^{-12}$ was used. UMass measure is a way of confirming that the model actually learned the data that it was trained with. Topic coherence uses the n_{top} words of each topic. n_{top} refers to the n words that have the highest weight in each topic.

In Figures 4.8 and 4.9 we see the evolution of topic coherence as the topic number increases. We observe that the topic coherence tends to be monotonically decreasing, even though in theory, we would like the coherence to be as high as possible. A possible explanation for this outcome, is that while the topic number increases, we observe that the 'junk' topics (topics that did not capture a concept, and to a human seem like random words), may increase. So, this decreases the value of the coherence metric. In any case, a non-interpretable topic is not a problem for our system, because we can define a-priori which topics are meaningfull, and exclude the rest from the process.

Thus, we have a measure to compare the topics of a model. But we require a measure to compare between different models, rather than individual topics from a specific model. So we average the topic coherences from each model.

4.4.3 Jaccard Similarity

The *Jaccard similarity* coefficient is a statistic used for measuring the similarity between two finite sets. It is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.5}$$

If $A = \emptyset$ and $B = \emptyset$ then we define $J(A, B) = 1$. In general, $0 \leq J(A, B) \leq 1$. For example, if we have two topics, $topic_1$ and $topic_2$, with:

$$topic_1 = \{police, murder, crime, case, death, mystery, detective, killed, later, dead\}$$

and

$$topic_2 = \{police, murder, case, crime, detective, death, trial, prison, evidence\}$$

the Jaccard similarity is computed as:

$$J(topic_1, topic_2) = \frac{A}{B} = \frac{6}{13} = 0.461$$

where

$$A = |police, murder, crime, case, death, detective|$$

$$B = |police, murder, crime, case, death, mystery, detective, killed, later, dead, trial, poison, evidence|$$

We compute the Jaccard similarity of an LDA model (with K topics) by taking the average Jaccard similarities between all its combination of topics. We do not include $J(A, A)$ (the similarity between the same topic) in the computation. So, for a model with K topics, we compute $(K-1)^2$ similarities. Formally, we compute for each LDA model with K topics LDA_K:

$$J_{LDA_K} = \frac{1}{(K-1)^2} \sum_{i=1, j \neq i}^{K} \sum_{j=1}^{K} J(LDA_{Ki}, LDA_{Kj}) \tag{4.6}$$

When $J(A, B)$ is above a certain threshold, a high correlation between the models is implied. Jaccard similarity is the simpler of methods used to determine the number of topics of our final model. We would like the average Jaccard similarity to be as low as possible. As we observe in Figures 4.10 and 4.11, as the topic number K increases, Jaccard similarity converges to a value of 0.0021. This turns out to be a weak indication that the bigger the K, the better. We want our topics to be semantically far from each other, to cover a wide spectrum of topics. Note however that two similar topics that nevertheless do not share any common word, have a Jaccard similarity of zero, but may be quite similar.

4.4.4 Human Judgement

We expect from the LDA algorithm to discover as many (meaningful) topics as possible from a large variety of Wikipedia articles. If our dataset was only composed by a few Wikipedia articles, we could use a human annotator for the document-topic assignments. A human could infer topics of equal, or better interpretability than the LDA algorithm. Obviously, for nearly 40.000 Wikipedia articles, that would be impossible. Regardless, even though we studied many techniques to determine the optimal topic number for our model, a human has the final say. Thus, given the obtained insights from the evaluations with *perplexity*, *coherence* and *Jaccard similarity* discussed in the end of Sections 4.4.1, 4.4.2 and 4.4.3 we will explore some of the models and their respective topics manually.

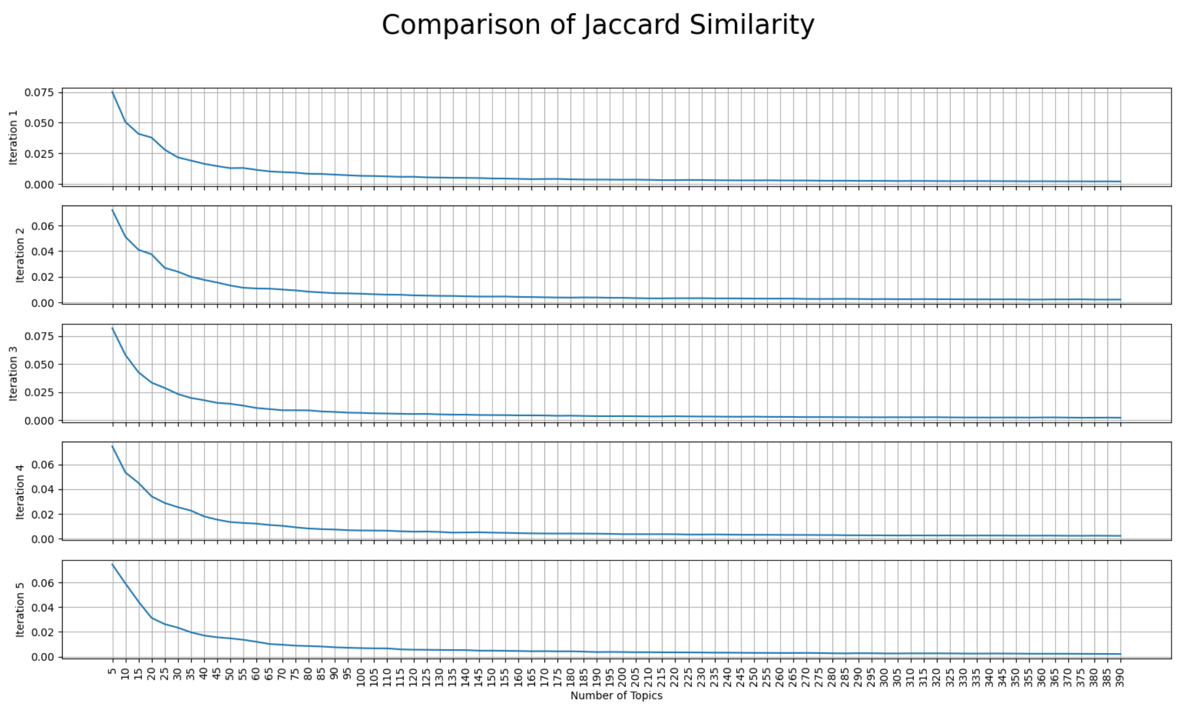


FIGURE 4.10: Jaccard similarity values of 5 evaluations

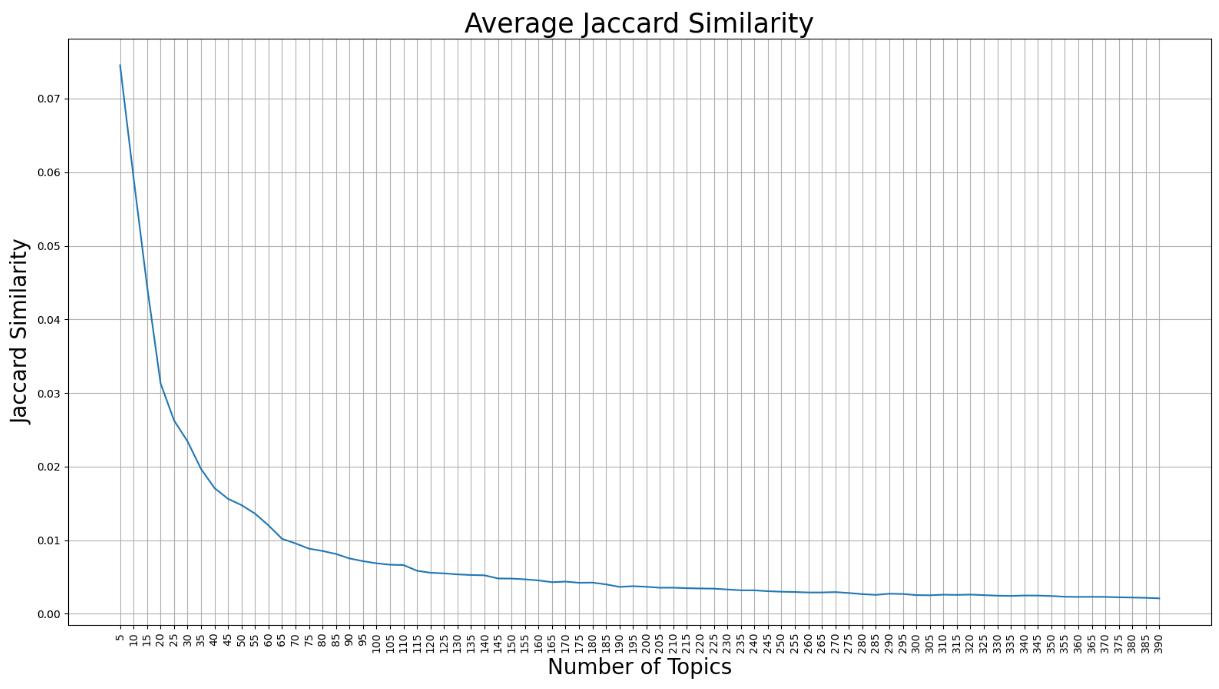


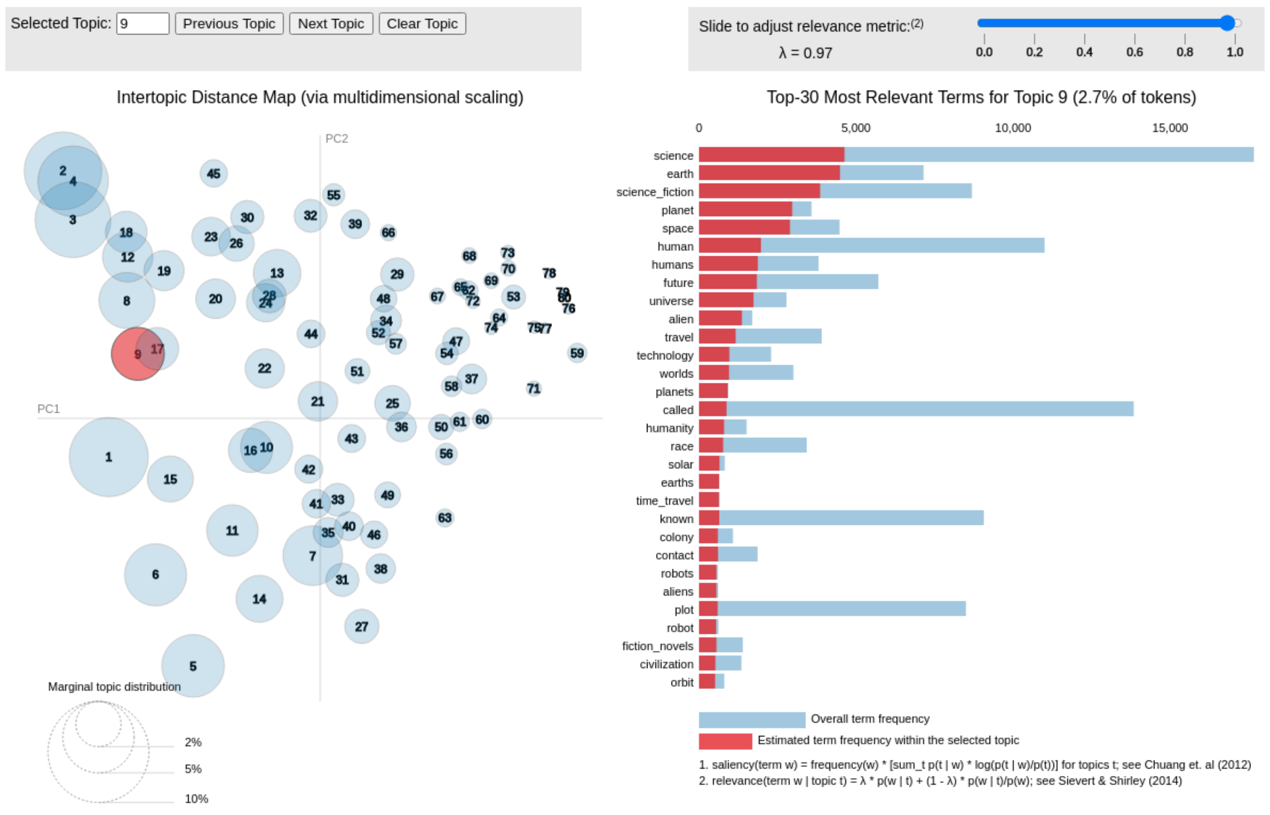
FIGURE 4.11: Average Jaccard similarity values

We use *pyLDAviz* [24] to visualize the topic term distributions of the various topics. As we discuss in appendix A, in a LDA model with $K = 80$ topics, 59 of those topics showcase a very clearly defined topic, with the remaining 21 being either completely composed of unrelated words, or composed of words that can have a connection, but not a clear one. We chose to continue our work with $K = 80$ topics.

Apart from the default ordering of words as seen in Figure 4.12, the practitioner can also examine different orderings of words per topic, by experimenting with the relevance parameter λ. In [24], the authors define the relevance of term w to topic k given a weight parameter λ (where $0 \leq \lambda \leq 1$) as:

$$r(w|t) = \lambda p(w|t) + (1 - \lambda)\frac{p(w|t)}{p(w)} \tag{4.7}$$

where w is a word (or bigram) from the topic vocabulary, and t is the topic. With $\lambda = 1$, the results are ranked by the familiar order of their topic-specific probability. With $\lambda = 0$, the results are ranked solely by their lift. Lift is defined as the ratio of a term's probability within a topic to its marginal probability across the corpus.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

FIGURE 4.12: Visualization of a topic (science fiction) in pyLDAvis

In conclusion, we discuss the insight given by each metric. The *Jaccard similarity* metric suggests that the we should choose the largest K possible, but it is a rather simple evaluation metric, not taking into consideration the topic quality, but only the topic terms. The Figures showing the *Jaccard similarity* give us an expected behaviour, and are mostly used to make sure that the model produced topics consisting of a variety of terms. The *perplexity* metric helped us find a good range of topics to examine. The *coherence* metric did not prove useful for identifying optimal topic number, but nevertheless proves that some topics of the LDA models must be removed from the final model; insight given by the examination of topics by a human.

4.5 Our Drama Manager Model

We use the LDA algorithm to model both the user and the items under recommendation. Following [28], we consider them both as mixtures over topics with a Dirichlet prior. The number of topics in both user and item models is set to the same value K. For this model to apply, some assumptions must be met:

- *There is descriptive information freely available.* We trained our DM with all the book pages from Wikipedia. A practitioner can choose different datasets to train the LDA algorithm. Of course, the dataset must be chosen carefully, in order to cover a wide spectrum of topics.

- *The dataset texts are objective, and not biased.* This assumption is made because we do not know the profile of the users using our DM. In any case where the practitioner is sure about the profile of the target group of the system, this assumption can be relaxed.

4.5.1 Plot point modeling

The items under recommendation in our approach are the story's plot points. The plot points $y_i \in Y, |Y| = M, 1 \leq i \leq M$ are represented by documents belonging to a corpus D, containing M documents. There are N words in total in the corpus vocabulary. The number of topics in both user and item models is set to the same value K. For each document, we choose:

- $\theta_i \sim Dir_K(\boldsymbol{\alpha})$, where θ_i is the distribution of topics in document i, and $Dir_K(\alpha)$ is the Dirichlet distribution of parameter $\boldsymbol{\alpha}$.

- $\phi_k \sim Dir_N(\boldsymbol{\beta})$, where ϕ_k is the word distribution for topic k, and $Dir_N(\beta)$ is the Dirichlet distribution of parameter $\boldsymbol{\beta}$

$\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the Dirichlet parameters of topic distributions per document and word distributions per topic, respectively.

4.5.2 User modeling

In a similar way, we consider that each user u_j can be represented by a "document". This document has the ability to evolve over time. It is a mixture of topics with a Dirichlet prior. The topic distribution mixture follows a Dirichlet distribution, like the plot points. So, $\theta_j \sim Dir_K(\alpha)$, where θ_j is the distribution of topics in the single document that models the user, $Dir_K(\boldsymbol{\alpha})$ is the Dirichlet distribution of parameter $\boldsymbol{\alpha}$, and K the number of topics.

4.5.3 Plot point model and user model updating

A plot point is a fraction of the story that lies between story branching points. When a user reads a plot point, she needs to provide a rating. Using Bayesian updating, this rating will alter her topic mixture. As the topic distribution of the document that models the user is unknown, we utilize Bayes rule to take into account evidence (user ratings), a likelihood function and a marginal probability, in order to derive a posterior distribution.

The likelihood function associates the prior with the observations, while preserving the form of the overall model. The posterior which is produced represents the updated belief for the prior, given the evidence. Using the posterior beliefs, we are able to update our unknown model. In our case we use the *Dirichlet* and the *multinomial* distributions, which are conjugate. This is a useful property, which allows us to perform easy updates to the prior's *hyperparameters*, using a closed form equation, which we will show below.

Hence, given documents $\mathbf{y}$ and with topics mixtures $\boldsymbol{\theta}$, we have:

$$\boldsymbol{\theta} \sim Dir(\boldsymbol{\alpha} = \langle a_1, ..., a_K \rangle) \tag{4.8}$$

$$\mathbf{y} \sim Mult(\boldsymbol{\theta} = \langle \theta_1, ..., \theta_K \rangle) \tag{4.9}$$

In detail, the topic mixtures $\boldsymbol{\theta}$ are described by:

$$Dir(\boldsymbol{\theta}|a_1, ..., a_K) = \frac{\Gamma(a_1 + ... + a_K)}{\Gamma(a_1)...\Gamma(a_K)} \prod_{i=1}^{K} \theta_i^{a_i - 1} \tag{4.10}$$

Given the evidence, consisting of a document y with Dirichlet prior:

$$f(\boldsymbol{\theta}|y) \propto f(\boldsymbol{\theta}, y) = f(\theta_1, ..., \theta_k|\alpha_1, ..., \alpha_k) f(y|\theta_1, ..., \theta_k) \propto \prod_{j=1}^{K} \theta_j^{a_j - 1} \prod_{j=1}^{K} \theta_j^{y^{(j)}} = \prod_{j=1}^{K} \theta_j^{a_j - 1 + (y^{(j)})} \tag{4.11}$$

Hence the updated hyperparameters of the Dirichlet prior are:

$$a_j{}' = a_j + y^{(j)} \tag{4.12}$$

where $a_j{}'$ is the updated user model, a_j is the user model prior to the update, and $y^{(j)}$ is model of the plot point that was just consumed. Thus, we can update the user model by simply adding to counts of each topic θ_k of the user "document", the topic counts from the plot point "document" that was just consumed and rated. That is, we perform an element-by-element addition of the user model and the plot point model. Also, we take the user's t-th rating r into account by updating the user model n times. Each rating the user provides, refers to the *story-so-far* (i.e., the story from the beginning until the current plot point). The value of n is updated as shown in the following equation:

$$n = \lfloor {\Delta_t}^r - 1 \rfloor \tag{4.13}$$

where Δ_t is the variable learning rate for the t-th recommended item, ranging from $\Delta_{min} = 1$ to $\Delta_{max} = 4$. Δ is assigned various values throughout the operation of our DM, according to the "Win or Learn Fast" (WoLF) method. Also r, which is the user rating, takes a value of $0, 1, 2, 3, 4$ or 5.

For example, let us define a prior user model $a_j = (0.1, 0.4, 0.2, 0.3)$, and a plot point model $y^{(j)} = (0.5, 0.5, 0, 0)$. The topic number in this example is $K = 4$ that is, the length of the vectors. For a learning rate $\Delta = 1.5$ and a rating of 3 out of 5, the update is performed as follows. First, the value of n is computed as $n = \lfloor 1.5^3 - 1 \rfloor = \lfloor 2.375 \rfloor = 2$. Hence, the addition $a_j + y^{(j)}$ is performed $n = 2$ times to compute the final updated user model. So, the first update is:

$$(0.1, 0.4, 0.2, 0.3) + (0.5, 0.5, 0, 0) = (0.6, 0.9, 0.2, 0.3)$$

and the second update is

$$(0.6, 0.9, 0.2, 0.3) + (0.5, 0.5, 0, 0) = (1.1, 1.4, 0.2, 0.3)$$

. The result is normalized, hence the updated user model is

$$a'_j = (0.36667, 0.46667, 0.06667, 0.1)$$

. This example showcases that the update shifts the user model towards the plot point that was just rated. With higher rating, it would have shifted even more towards the plot point model.

Therefore, items that have been positively rated by the user, can be thought as having greater *influence* on the overall user preferences. Further, the intuition behind the above equation is that items rated by 0,1 and 2 should have minimal or no influence on the evolution of the user's model. Contrary to that, items rated with 3 to 5 should contribute proportionally to their significance. We empirically found suitable values of Δ_t to lie between 1.1 and 3.0, and we thus allow Δ_t to range between these values in our implementation.

4.6 Prefix Tree Representation

In CYOA books, the user can affect the way in which the story unfolds. The first step towards creating an algorithm that tackles the sequential recommendation problem is to transform the branching story graph into a prefix tree, or prefix graph (we use the terms prefix graph and prefix tree interchangeably, as they are the same object). We define a story as a path from the root to a terminal node (leaf) of the branching story graph. Figure 4.13(a) shows a story graph. In Figure 4.13b it is transformed into a prefix tree. In Figure 4.13b every node is a prefix of a possible generated story. The children of a node in the prefix tree are prefixes that can directly follow the parent

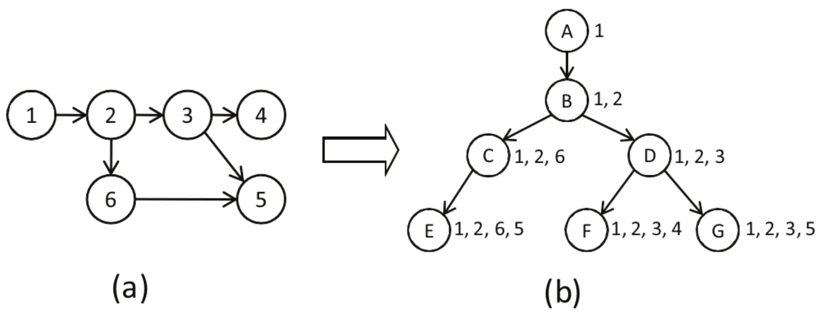


FIGURE 4.13: (a) Branching story graph of a simple story library which contains three stories. (b) The prefix graph of the story library. obtained by [33]

prefix. In Figure 4.13 we see three possible complete stories: {1, 2, 6, 5}, {1, 2, 3, 4} and {1, 2, 3, 5}. While the transformation between the two representation may seem obvious, the employment of a data structure like the prefix tree is crucial for a drama manager like the one we discuss. With the prefix tree, the drama manager does not need to worry about the past nodes, because all the past nodes are incorporated into the prefix tree themselves. It is a tradeoff between simplicity in the implementation process, and some overhead in memory, which we are aware of. We use the python package *treelib* to create the prefix graph. Treelib offers a function *leaves*, that given a node, returns all the leaves reachable from this node.

The stories will be presented to the user plot point by plot point. After each plot point, the drama manager will collect a rating for "the story so far". The collection process is performed with a command line utility as explained in Section 4.8. The rating is used for the user model update, as shown in Equation 4.13. We believe this collection method is preferred to collecting a rating of the previous plot point, because any plot point does not make sense without the previous ones. Also, it is more difficult for a user to isolate the feeling she has developed for a specific part of the story, and to not consider the context of the prior story. Finally, there is no need for the drama manager to solve the credit assignment problem as in reinforcement learning to determine the proportion of a final rating each plot point is responsible for [26].

In Figure 4.14 the architecture of an interactive system is shown. Our DM also follows this architecture. The DM has access to a story library, which contains every possible plot point. The user interacts with the DM through an interactive system interface. The system takes as input the user ratings and the current plot point of the story, and as output the next plot point. Also, the player model depends on the player feedback.

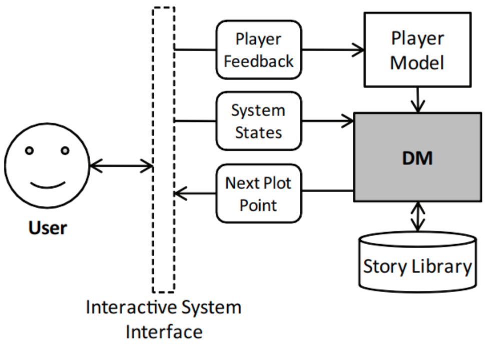


FIGURE 4.14: The architecture of the interactive story generation system. obtained by [33]

4.7 Recommendation phase

The recommendation phase consists of two main functions: a) The querying of the available low-dimensional representation of items for the most similar and b) the monitoring of the system performance and adjustment of the learning rate. The recommendation process is described in the pseudocode of algorithm 1.

4.7.1 Overview

Lets assume that the user starts using the system. At some point, she will face the first branching of the story. The DM will collect a rating of the story-so-far, which is the first plot point, since the story just started. We start the DM with the maximum value of Delta, $\Delta = 4$, to force the user model to adapt fast into a user model close to the actual one. At this point, where we have our first evidence (first rating) about the user preferences, we can start recommending plot points based on that user model. Every leaf node of the prefix graph has the information about all the plot points needed to reach the leaf. For each complete story reachable from a node, we compute the projected user model if the user follows that path. As prior knowledge, we use the current user model. When we have computed all the possible final user models, we select the best, using one of the three methods discussed below. Our DM supports the addition of multiple distance metrics, to make the experimentation for the practitioner easier. In our case, the expected utility metric yielded the best results, though we also describe cosine similarity and Shannon Jensen divergence, for the sake of completeness.

4.7.2 Shannon Jensen Divergence

We study the application of the Shannon Jensen divergence as one of the distance metrics. Jensen Shannon divergence between probability vectors p and q is defined as:

$$JSD(p,q) = \sqrt{\frac{KL(p||m) + KL(q||m)}{2}} \tag{4.14}$$

where m is the pointwise mean of p and q and KL is the Kullback-Leibler divergence. The Jensen–Shannon divergence is bounded by 1 for two probability distributions.

$$0 \leq JSD(P||Q) \leq 1 \tag{4.15}$$

In turn, the Kullback-Leibler (used to calculate the JSD) divergence between two discrete probability distributions P and Q is defined as:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) log \frac{P(x)}{Q(x)} \tag{4.16}$$

4.7.3 Cosine Distance

Inspired by the work of Tripolitakis & Chalkiadakis [28], we also test their recommendation approach. Given the fact that both users and items are represented by the same distribution, we can assess their similarity by employing the cosine distance D_{cosine} metric:

$$D_{cosine}(P, Q) = 1 - \frac{\sum_{i=1}^{n} P_i \times Q_i}{\sqrt{\sum_{i=1}^{n} P_i} \times \sqrt{\sum_{i=1}^{n} Q_i}} \tag{4.17}$$

where:

P, Q are distributions of the same type and same size.

The lower the D_{cosine} metric, the greater the proximity between the two distributions.

4.7.4 Expected Utility

The cosine distance metric, while proven to work well under the circumstances studied in [28], does not perform so well as explained in Chapter 5. To optimize our DM for the sequential recommendation problem we try to tackle, we introduced the notion of expected utility. At any given plot point, the user is modeled as a mixture of latent topics with a Dirichlet prior. By experimental evaluations discussed in Chapter 5, we define the user utility given the user model and a plot point as the dot product between the current user model and the plot point's topic proportions given by the item model:

$$U(\boldsymbol{u}_j, \boldsymbol{y}_i) = \boldsymbol{u}_j \cdot \boldsymbol{y}_i \tag{4.18}$$

To be able to address the sequential selection problem, for each possible story produced from the current plot point our DM calculates the utilities for each plot point. Then, the average utility is stored for each path. Finally, the DM recommends the next page belonging to the path with the maximum average utility.

For example suppose that during a session, a user is at node D of the story graph of Figure 4.15. At this point, the DM knows what the possible stories are, by finding all the leaf nodes accessible from D. So, leaf node E is omitted from the recommendation

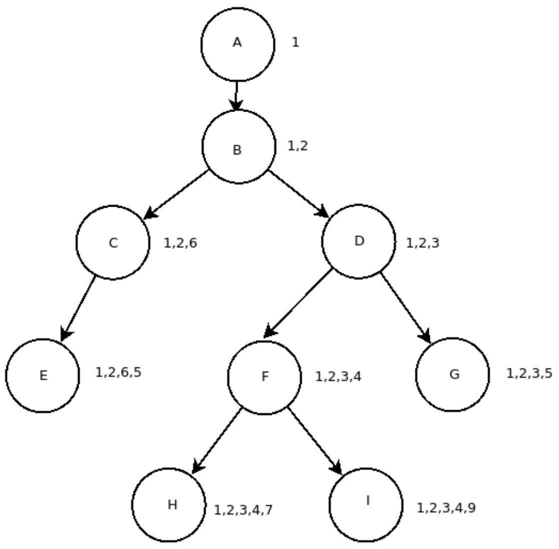


FIGURE 4.15: Example of a prefix graph

process, and nodes H and I and G will be examined. The information about the complete story path is contained in each one of the leaf nodes, so the DM only needs those three nodes to make a decision.

4.7.5 Drama Manager Algorithm

The overall description of our DM is found in algorithm 1. The cosine distance and maximum expected utility are explained in Subsections 4.7.3 and 4.7.4, respectively. The *Update user-model* state uses the Equations 4.12 and 4.13. We observe the update of the learning rate, explained in 2.6.

Finally, in line 9 of the algorithm, we observe the condition that determines *Losing* and *Winning* of the Equation 2.6. The global average of user ratings $\overline{r_{u_j}}$ is compared to the latest ξ ratings, $\overline{r_{u_{j\xi}}}$ multiplied by a factor c. In our experiments shown in 5, we use a value of $c = 1.1$ and a value of $\xi = 3$. An table containing all the parameters used by our DM, is shown in Appendix C.

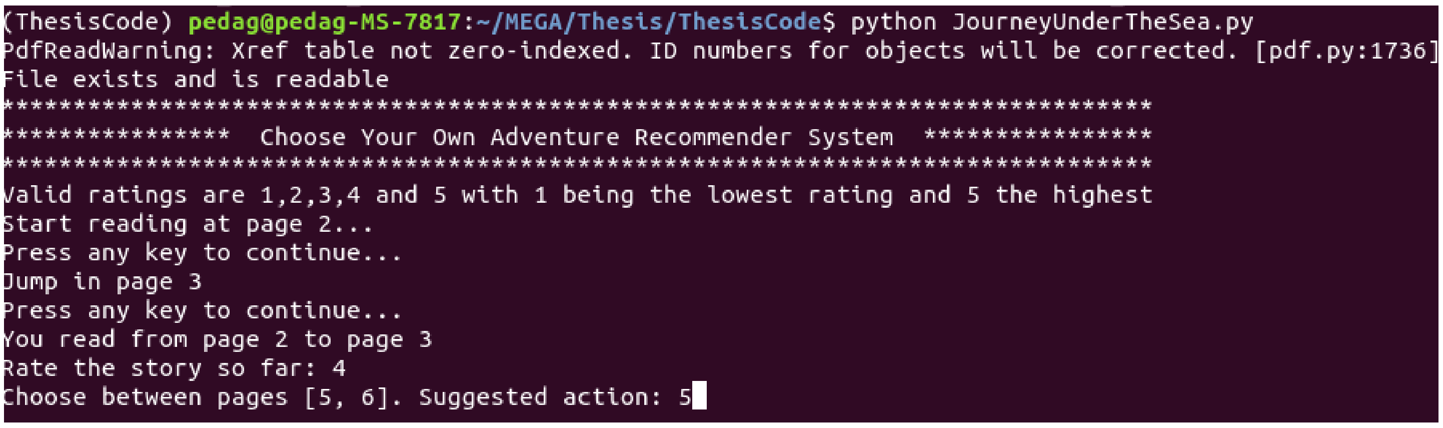
Algorithm 1 Plot point recommendation

```
1: procedure RecommendNextPlotPoint
2:     for each item $y_i \in D$ do
3:         Calculate the maximum utility between the item and the user $u_j$ and store it in an array
4:     end for
5:     Ask user for a rating of the story so far
6:     updateUser:
7:     Update the average of user ratings $\overline{r_{u_j}}$ and the average rating for the latest $\xi$ recommendations $\overline{r_{u_{j\xi}}}$
8:     if $\overline{r_{u_j}} > c\,\overline{r_{u_{j\xi}}}$ then
9:         $\Delta_{t+1} = min(\Delta_t + \delta_{win}, \Delta_{max})$
10:    else
11:        $\Delta_{t+1} = max(\Delta_t + \delta_{lose}, \Delta_{min})$
12:    end if
13:    Update user-model
14: end procedure
```

4.8 User Interface

Our drama manager is in a form of a command-line utility, that the user runs when he starts reading the story. When the user faces a decision she checks the drama manager, and the drama manager yields the suggested option. The drama manager will present its recommendation, but the user will be the one to decide how the story will proceed. When the user makes a decision, she enters one of the available options back to the command line utility and then she continues reading the story.

In Figure 4.16, we see an example of our DM in action. We showcase the first choice of the CYOA book *Journey Under The Sea*. We observe that in page 2 user input was not asked, because there in no choice to make. Thus, the DM promts the user to continue



```
(ThesisCode) pedag@pedag-MS-7817:~/MEGA/Thesis/ThesisCode$ python JourneyUnderTheSea.py
PdfReadWarning: Xref table not zero-indexed. ID numbers for objects will be corrected. [pdf.py:1736]
File exists and is readable
********************************************************************************
****************  Choose Your Own Adventure Recommender System  ****************
********************************************************************************
Valid ratings are 1,2,3,4 and 5 with 1 being the lowest rating and 5 the highest
Start reading at page 2...
Press any key to continue...
Jump in page 3
Press any key to continue...
You read from page 2 to page 3
Rate the story so far: 4
Choose between pages [5, 6]. Suggested action: 5
```

FIGURE 4.16: Example of a choice in our DM

reading to page 3. In page 3, a choice must be made, so the DM collects a rating of the story so far, and then calculated the best path according to the user model.

Chapter 5

Experimental Setup - Evaluations

Following our discussion in Chapter 4 of how to process the Wikipedia dumps and train the LDA model, in this Chapter we present in depth our simulation results.

5.1 Workstations and Training Times

We had two workstations in our disposal for this work. A desktop PC equipped with an Intel i5-4440@3.3 GHz (4 cores, 4 threads) and 8GB of RAM, and a laptop equipped with an Intel i7-6500U@2.5 GHz (2 cores, 4 threads) and 8 GB of RAM. In Chapter 4 we discussed the methods used in order to process the Wikipedia dataset. Here we will discuss the computational setup for our experiments. To utilize the available resources optimally we applied multiprocessing techniques to process our data more quickly. The laptop was chosen to make all the heavy computation even though it has 2 cores compared to 4 cores of our desktop, due to a cooling problem of our desktop PC CPU (when the desktop CPU was running at 100%, it was overheating). 59 compressed files were downloaded from the Wikipedia dumps. Using the python *multiprocessing* package, we mapped the 59 files to the 4 available threads. For each one of the 59 mentioned files, our system creates a new *.ndjson* file[1], containing only the articles refering to books. The workload was high, rendering the CPU usage at 95-100% continuously. The wikipedia dump processing was completed after five days of processing. We managed to reduce the wikipedia dataset size from 34,3 GB to 197,9 MB, by only keeping the articles refering to books.

Following a similar scheme, the training of the LDA algorithm was also performed on the laptop, with multiprocessing applied. For most experiments, we trained 78 LDA models (from $K = 5$ to $K = 390$ with a step of 5). Once again, we mapped the training of the models to the 4 threads. The 78 models are complete after 24 hours of training. Because we performed 5-fold cross validation, the final models were computed after five days.

5.2 Using Real CYOA books

In order to evaluate our system, we had to find some CYOA books to experiment with. The CYOA book the code was mostly built on, was *Journey Under The Sea*, published

[1]http://ndjson.org/

in May, 2006. We found several CYOA books online, but none of them had enough main topics in order to use it as a reference book for our experiments. This is why we created simulated books and users for our experiments, as explained in Section 5.3. Nevertheless for sake of completeness, in Section 5.2.1 we describe the process we followed in order to use real books, in case a practitioner founds suitable books for experiments.

5.2.1 Gathering Information from a PDF

Construction of the Story Graph

Given a CYOA book, our DM must be able to obtain the connections between the various plot points. To create a story graph, we must iterate over all the pages of a book and search for spedific keywords. We provide a function called *getDestination*, that uses a page's text as input, and returns the pages the user can be redirected to. This is currently only tested on CYOA books. Of course, a practitioner can customize this function to apply to any other secret story book. To craft a function that uncovers the structure of a CYOA-like book, we must carefully study the ways the author uses to guide the user. In our case we identify 5 distinct cases that each page falls into:

- The page contains a variation of the phrase *"Turn to page x"* one or more times.
- The page contains the phrase *"The End"* with no other phrases indicating alternate endings.
- The page contains the phrase *"The End"* and a variation of the phrase *"if you do not like this ending, turn to page x"*.
- The page contains a variation of the phrase *"Go on to the next page"*.
- None of the above, so the user just continues reading to the next page.

Finally, after we have iterated over all the pages of the book, the required information is gathered to create the story graph, and continue as described in this thesis. That is, for every page of the book, we know whether it is a terminal page or not, and to which pages it is connected to.

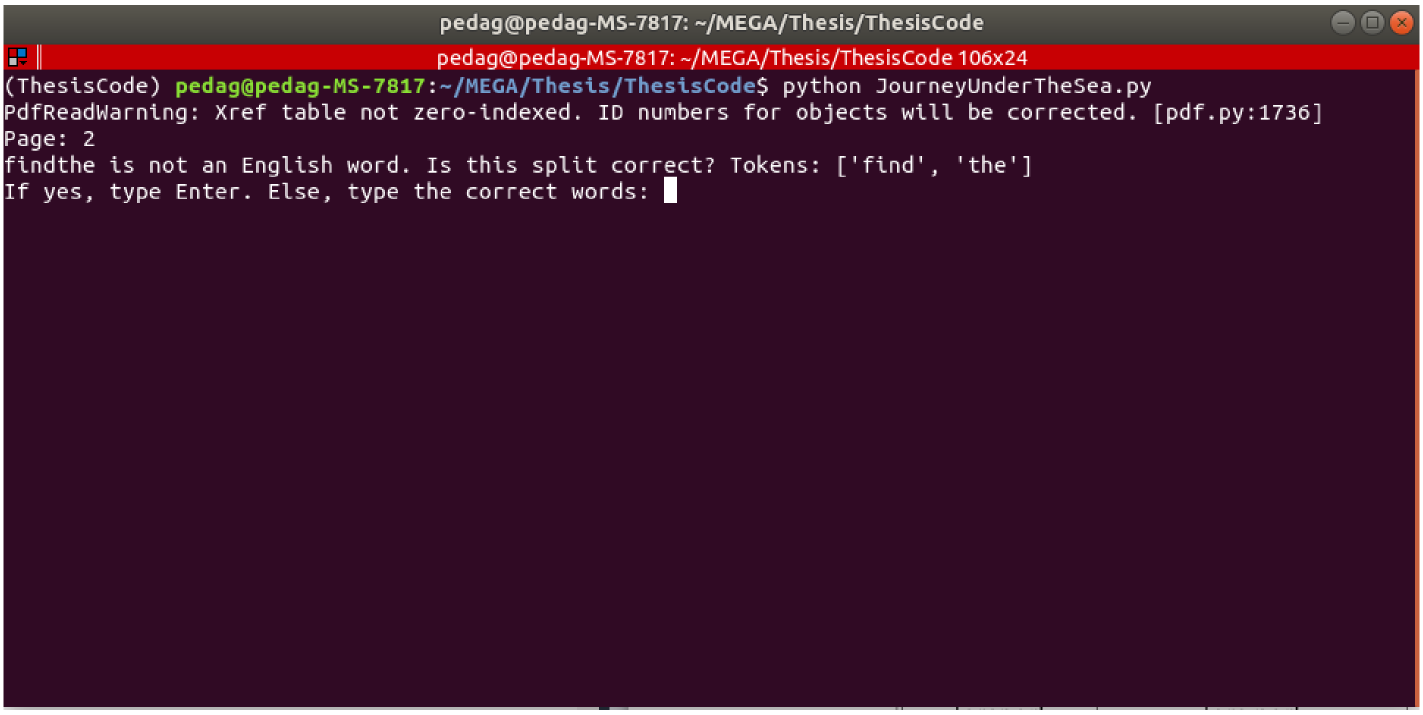
Merged Words

The most common format of digital books is the PDF format. We used the python package *PyPDF2* to obtain the full text of the book. The .pdf file format is widely used and known because pdf files can be viewed on any platform. Now, pdf files have drawbacks. PDF has a locked layout, meaning that every element of the file (such as letters, images etc.) has its own place in the layout. Even though many pdfs look fine when reading them, they may have a very unstructured internal layout. Unfortunately, our CYOA books also had some problems. PyPDF2 offers functionality that can extract the text from a given page of the pfd file. The obtained text though, did not include the new line characters, resulting in the final word of a line, and the first word of the next line, to be merged together. Even though there are some packages like python's *compound-word-splitter*, they do not have 100% accuracy. This means that when there is more than one possible split of the two merged words, there is a chance the result is incorrect. Our algorithm's accuracy is based solely on the textual information of the books, so we do not want to risk losing potentially important words from the book. To prove that this package does not work optimaly, we provide some wrongly splitted words from CYOA, Journey Under The Sea:

Merged Words	compound-word-splitter	Original Meaning
theresearch	Therese arch	the research
ofthe	oft he	of the
youthink	youth ink	you think
toolate	tool ate	too late
youthat	youth at	you that
toleave	tole ave	to leave
toreport	tore port	to report
otherspaces	others paces	other spaces
toescape	toes cape	to escape
asmall	as mall	a small

A full list of the ambiguous merged words can be found in appendix B. The problem of splitting merged english words is hard. In order to become automated, it requires the application of natural language processing methods which goes beyond the scope of this thesis. Though, this could be a future extension. As a consequence, we implemented a command line utility to split the merged words manually. The user probably has the pdf or the printed book in her possession. So, during the first time our DM is executed, a command line pops up, requesting for the manual inspection of merged words. We exploit the fact that the uses possesses the book in some format, so she can refer to the book to complete the process easily. In figure 5.1, we see a screenshot of this process. For every ambiguous word the DM shows the inferred split, and the user can either type the correct split, or press Enter if the proposed split is correct.

Apart from merged words, the command line utility of figure 5.1 also deals with informal words. Because we do not know the words that are merged, all the words of the book are checked, one by one. Every word is checked against an English dictionary. If the word exists, then the process continues. But if the word is not contained in the English dictionary, there is a potential candidate for two merged words. The command line utility then lists the token and asks for the correct split. There is a possibility that the word is not a merged word produced by two other words, but rather a word that is not included in the English dictionary. Some examples from the CYOA book, Journey Under The Sea are: Atlantis, Atlanteans, Nodoors (referring to a tribe living in Atlantis, etc). In this case, the user presses Enter, and the process continues.



pedag@pedag-MS-7817: ~/MEGA/Thesis/ThesisCode
pedag@pedag-MS-7817: ~/MEGA/Thesis/ThesisCode 106x24

```
(ThesisCode) pedag@pedag-MS-7817:~/MEGA/Thesis/ThesisCode$ python JourneyUnderTheSea.py
PdfReadWarning: Xref table not zero-indexed. ID numbers for objects will be corrected. [pdf.py:1736]
Page: 2
findthe is not an English word. Is this split correct? Tokens: ['find', 'the']
If yes, type Enter. Else, type the correct words:
```

FIGURE 5.1: Command line utility for merged words inspection

5.3 Simulating Users and Books

The CYOA books we have in our disposal, being created for children, do not contain a wide variety of topics. Most books we examined contained few topics. Also, the majority of stories are composed by those few topics. As a consequence, we cannot extensively test our DM with such books. Thus, to create reliable results, we created synthetic users and synthetic books.

To create the synthetic books needed, we need to create a branching story graph. In order to save time, we used the branching story graph of the CYOA book, *Journey Under The Sea*, as seen in Figure 1.2. Any other existing or artificial story graph would be applicable. Following the structure of Figure 1.2, we suppose that each node in the graph, is a plot point of the book. For our later experiments, we simulate 500 independent runs of the DM, with different user preferences, and different topic proportions for each plot point. The only common attribute between those independent runs is the structure of the branching story graph, which remains the same for simplicity reasons. The common branching story graph, does not affect the independence between the DM simulations. In the evaluation section, each simulation is referred as *episode*. The term *episode* must not to be confused with its meaning in the reinforcement learning domain.

For the LDA model, we choose $K = 80$ topics. Though, in our experiments evaluating the performance of the DM, we use $L = 20$ as the topic number. For L:

$$0 \leq L \leq K \tag{5.1}$$

The DM, knows what content is available by accessing the story library (Figure 4.14). In the CYOA domain, and in secret path books in general, 20 topics are sufficient to capture a multitude of content. So, the DM can access the calculated LDA model, and omit the topics that are not used in a particular interactive session. We made this decision because the performance drops when the topic number increases, as described in Section 5.4.

Populating the Plot Points with Topics

In order to accurately simulate our DM, we must create plot point topic proportions that could belong to an actual CYOA book. In order to prove our DM's ability to find the optimal path, the book under recommendation must have a considerable amount of possible stories. The book we experiment with, *CYOA-Journey Under The Sea* has 186 distinct stories the user can read. The process of synthetic user and synthetic book creation is as follows.

We create a random user model by a random permutation of elements of the vector $\boldsymbol{utility\_values} = [2, 3, ..., L-1, L, L+1]$, with L being the number of topics. In our experiments, we use a value of $L = 10$. We name this permutation of the vector $\boldsymbol{utility\_values}$ as $\boldsymbol{user\_preferences}$. Then, we normalize $\boldsymbol{user\_preferences}$, in order to represent a discrete probability distribution. The vector $\boldsymbol{user\_preferences}$ represents the hidden user model. To define how this user rates plot points, we defined his *utility function*, which is a scalar value dependent of each plot point the user encounters. The higher the utility value between a user and a plot point, the more the user likes the plot point. The utility is defined as follows:

$$U(\boldsymbol{u_j}, \boldsymbol{y_i}) = \boldsymbol{weight} \cdot \boldsymbol{y_i} \tag{5.2}$$

where $\boldsymbol{u_j}, \boldsymbol{y_i}, \boldsymbol{weight} \in \mathbb{R}^K$. $\boldsymbol{u_j}$ is the user model, $\boldsymbol{y_i}$ is the item (plot point) model, and $\boldsymbol{weight}$ is a vector with weights, indicating which topics the user likes. The vector $\boldsymbol{y_i}$ is a discrete probability distribution, and represents the topic proportions a plot point is described of.

In the $\boldsymbol{weight}$ vector, each index corresponds to a topic. For example, let us define the following weight vector:

$$\boldsymbol{weight_{example}} = [L+1, L, ..., 3, 2]$$

The weight vector $\boldsymbol{weight_{example}}$ suggests that the user adores topic 0, and the preference slowly declines till topic L-1, which the user hates. It is useful to note, that for $\boldsymbol{weights_{example}}$, $U_{min}(\boldsymbol{u_j}, \boldsymbol{y_i}) = 2$, while $U_{max}(\boldsymbol{u_j}, \boldsymbol{y_i}) = L+1$. $U_{min}(\boldsymbol{u_j}, \boldsymbol{y_i})$ is obtained by $\boldsymbol{y_i} = [0, 0, ...0, 1]$, while $U_{max}(\boldsymbol{u_j}, \boldsymbol{y_i})$ by $\boldsymbol{y_i} = [1, 0, ...0, 0]$. All item models in our system are probability distributions, because they are outputs of the LDA model discussed earlier. This explains why the aforementioned vectors are the ones that corresponds to the maximum and minimum utilities. In general, $2 \leq U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq L+1$. Item model $\mathbf{y_i}$ for U_{min} and U_{max} is as described, because it represents topic proportions and thus, it is a probability distribution. The $\boldsymbol{user\_preferences}$ vector mentioned before encodes the user's topic preferences in a similar way as the $\boldsymbol{weights}$ vector.

Using this model, we can translate the $U(\boldsymbol{u_j}, \boldsymbol{y_i})$ into a rating $r \in [0, 1, 2, 3, 4, 5]$. We do that by splitting the interval $[2, L+1]$ into buckets as follows:

- When $C \leq U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq C + \frac{(L+1)-(C)}{6}, r = 0$
- When $C + \frac{(L+1)-(C)}{6} < U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq C + 2\frac{(L+1)-(C)}{6}, r = 1$
- When $C + 2\frac{(L+1)-(C)}{6} < U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq C + 3\frac{(L+1)-(C)}{6}, r = 2$
- When $C + 3\frac{(L+1)-(C)}{6} < U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq C + 4\frac{(L+1)-(C)}{6}, r = 3$
- When $C + 4\frac{(L+1)-(C)}{6} < U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq C + 5\frac{(L+1)-(C)}{6}, r = 4$
- When $C + 5\frac{(L+1)-(C)}{6} < U(\boldsymbol{u_j}, \boldsymbol{y_i}) \leq L + 1, r = 5$

where C is the minimum value of the $\boldsymbol{weights}$ vector. In our case, $C = 2$, and $L = 20$. We have $|R| = 6$ utility intervals ($|R|$ denotes the cardinality of R i.e., the number of elements of R).

Practitioners can modify the aforementioned intervals for different values of the $\boldsymbol{weight}$ vector or rating values.

The aforementioned utility limits can be adjusted to every utility function proposed by the practitioner. We chose to model the user's preferences with a linearly increasing utility between the various topics. Also, we split the range of plot point utilities in six equal intervals in order to map plot points to ratings.

Using the user preference model, we can populate the plot point topic proportions in the following manner. We select two out of the 186 possible stories at random, and declare one of them to be the story the user likes, and the other one to be the story the user dislikes. The first path referred as "the liked path" and the second path is referred as "the disliked path", respectively. We check if they are different paths. If they happen to be the same path, we repeat the process until the two paths are different. For each node in the liked path we generate random vectors until we have generated a random vector that the user will rate with a rating of $r = 5$. Those random vectors, are of size L, and have from three to seven (chosen at random) topics activated, whose values are also random. Each activated topic's value is sampled from a uniform distribution. After that, the random vector produced is normalized. We repeat the process for the disliked path, with the difference that now the user must rate this path's plot points with a rating of $r = 0$. (If the two paths have plot points in common, we assign them ratings of $r = 5$). The rest plot points of the story graph are populated with random vectors, to ensure that the rest of the story will have random ratings from the user.

5.4 Searching for the Optimal Path

In this section, we may use the terms *hidden user model* and *actual user model* interchangeably. This also same applies to the terms *DM simulation* and *DM episode*.

The most crucial aspect of a drama manager is to be able to identify the best possible story path (in terms rating) from a pool of preauthored stories. In each one of the Figures shown later in this Section, we observe the average rating users give at each one of 500 independent simulations of our DM. We use the term *average rating* because for each story, the user will provide as many ratings as there are plot points. So a reliable performance metric is the average rating for each story. As explained in Section 5.3, the term *independent simulation* means that in each one of the 500 DM simulations, a uniform prior user model is assumed. We will call one such independent simulation, *an episode*, for short. Also, in each simulation the user preferences change at random, and so do the book topic proportions. However all the simulations share the same branching story graph. To summarize, each independent DM simulation tells us how one *random user* will rate a *DM recommended story*. In each simulation, the book's topic proportions change. Overall, in each of the Figures below, we see the rating behaviour of 500 *different users*, each one involved in a different episode. Also, the term *episode* seen in the following figures must not to be confused with its meaning in the reinforcement learning domain. In this thesis, the term episode has a meaning of an independent trial.

In each episode of our DM two random paths are chosen to represent the "liked path" and "disliked path", and are given appropriate topic proportions, based on the actual user model. Specifically, by introducing the "liked path" we aim to model the optimal path our DM must discover. The "liked path" is a full story, which is crafted based on the hidden user model. We must clarify, that the actual user model differs from the user model we use in our updates. The actual user model vector is produced only in order to provide the simulated users' ratings. Our DM is obviously prohibited to access this vector.

We observe that in the vast majority of the simulations, the DM manages to guide the user through the optimal path. In the simulation set of Figure 5.2, the "liked path" is populated with topics giving the user high utility (thus the user will rate each plot point with 5). Respectively, the "disliked path" is populated with topics giving the user the lowest utility, and thus the user will rate each plot point in the story with 0. Overall, in this setting our DM managed a overall average rating of 4.8. The absence of our DM (which means whenever the user is faced with a choice, she picks one at random) yielded an average rating of 2.8. Also, we observe that at 485 out of 500 episodes, the use of our DM resulted in higher average ratings. This means that our DM has an 0.97 rate of recommending good quality stories, when the user's "liked path" consists of plot points that all will be assigned the maximum rating (which is 5 in our experiments).

In Figure 5.3, we observe a different set of results. In this experiment, we designed the plot points of the story such that half plot points of the user's "liked path" will be assigned a rating of 5, and the remaining half will be assigned a rating of 4. We observe that the majority of average user ratings lie at 4.5. This is the best we can hope for, as the average of the true ratings (according to the actual user model) is 4.5. Also, we observe that at 431 out of 500 episodes the use of our DM resulted in higher average ratings. Thus, in this setting our DM has an 0.86 rate, of recommending good quality stories.

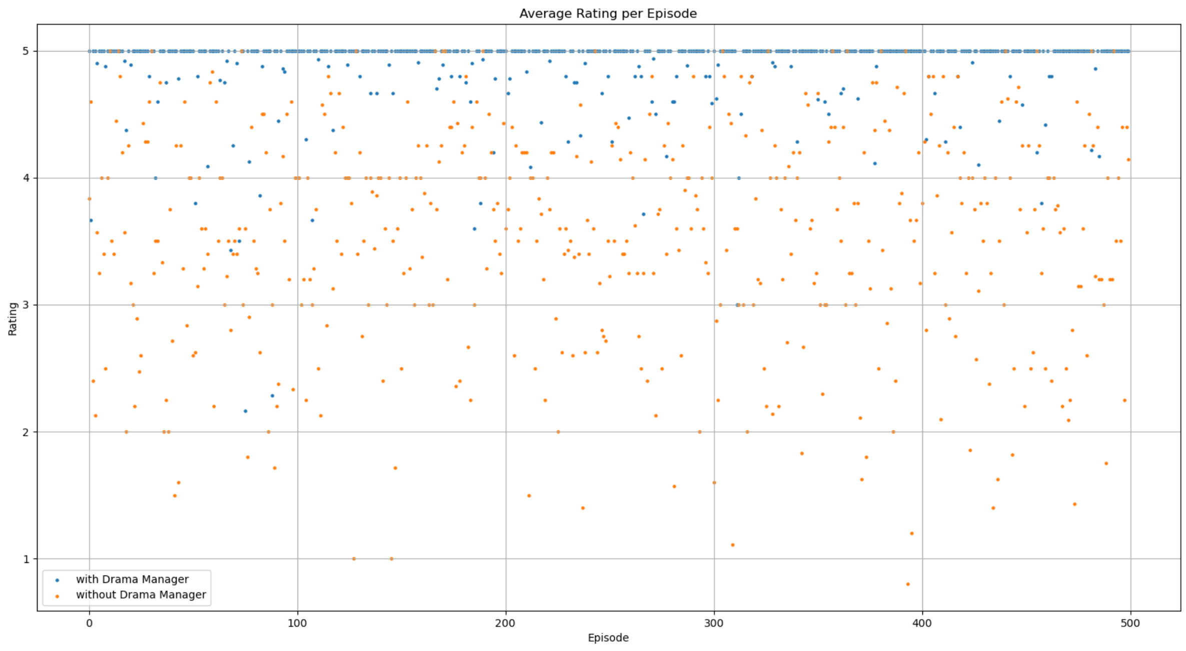


FIGURE 5.2: Average rating of stories (a 5-rated path is available)

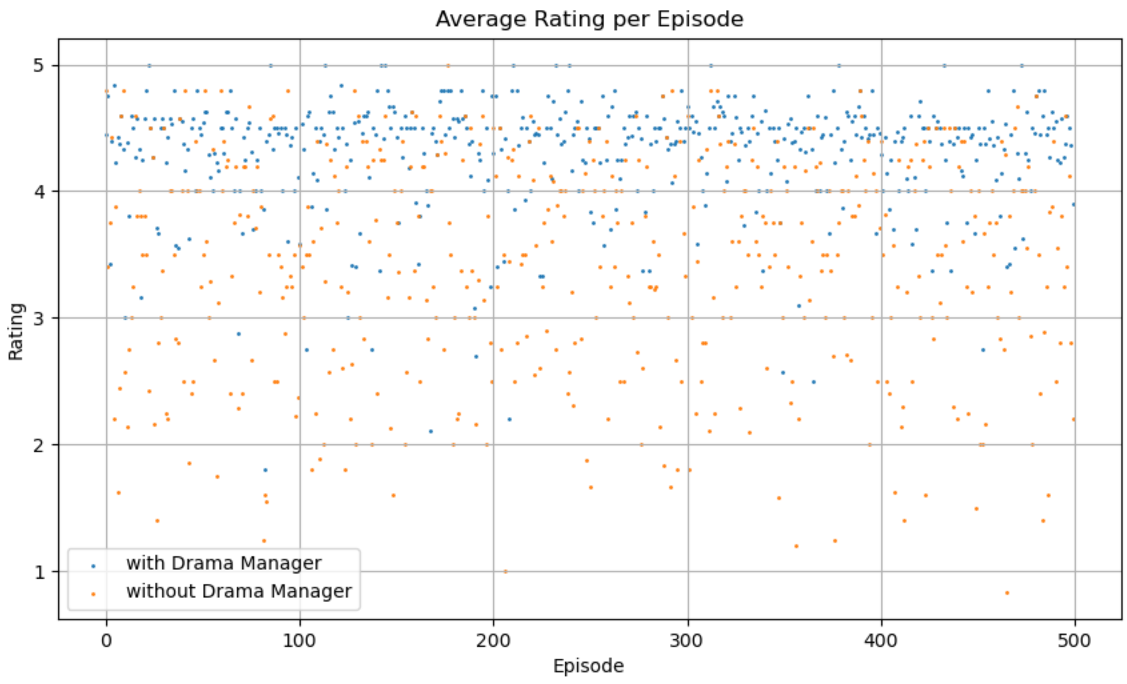


FIGURE 5.3: Average rating of stories (The user's liked path is populated with 50% 5, and 50% 4 rated plot points)

We previously mentioned that the performance of our DM drops as the topic number increases, which is why we are not using the full LDA model calculated, but only the topics which are present each book we are dealing with. We use $L = 20$ topics for the experiments of Figures 5.2 and 5.3. In Figures 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 we show the performance drop for different topic numbers (i.e., different values of L).

In Figure 5.4, the average rating in the presence of our DM is 4.46, while in absence of DM the obtained average rating has a value of 3.1. 430 out of 500 episodes yielded a higher average rating when using the DM.

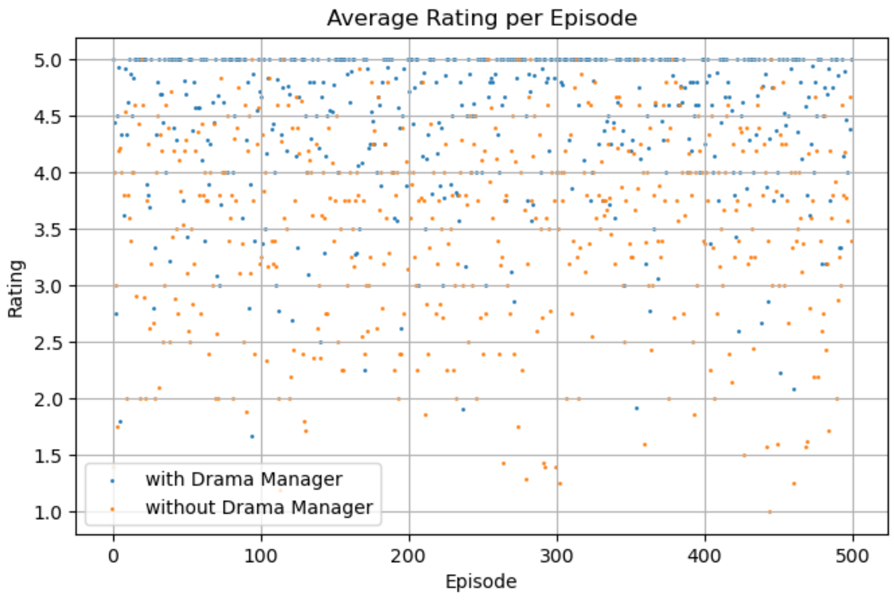


FIGURE 5.4: Average rating of stories (The user's liked path is populated with ratings of 5), L=30 topics

In Figure 5.5, the average rating in the presence of our DM is 4.21, while in absence of DM the obtained average rating has a value 2.9. 395 out of 500 episodes yielded a higher average rating when using the DM.

In Figure 5.6, the average rating in the presence of our DM is 4.1, while in absence of DM the obtained average rating has a value of 2.85. 383 out of 500 episodes yielded a higher average rating when using the DM.

In Figure 5.7, the average rating in the presence of our DM is 4.15, while in absence of DM the obtained average rating has a value of 2.95. 380 out of 500 episodes yielded a higher average rating when using the DM.

In Figure 5.8, the average rating in the presence of our DM is 4.11, while in absence of DM the obtained average rating has a value of 3.1. 384 out of 500 episodes yielded a higher average rating when using the DM.

In Figure 5.9, the average rating in the presence of our DM is 4.00, while in absence of DM the obtained average rating has a value of 3.05. 355 out of 500 episodes yielded a higher average rating when using the DM.

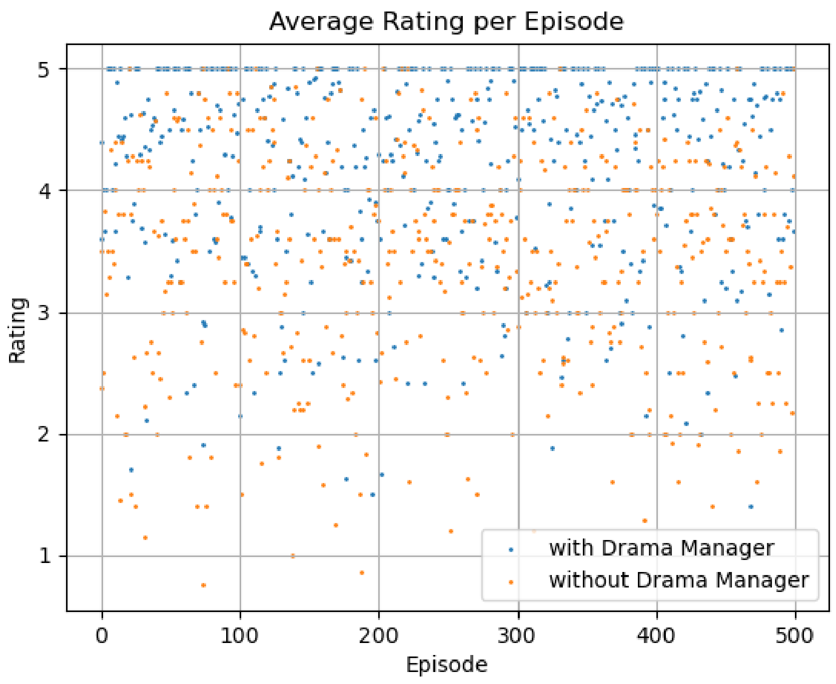


FIGURE 5.5: Average rating of stories (The user's liked path is populated with ratings of 5), L=40 topics

In the Figures above, we observe a decrease in the performance of our DM over time. This is why we chose to tackle this issue, by using only the L most important topics for each book, and omitting the rest $K - L$ topics. The decrease in the performance of our DM is visible in the plots 5.4 to 5.9, as the average rating decreases. As K increases, more and more ratings corresponding to the use of our DM tend to move away from higher rating values.

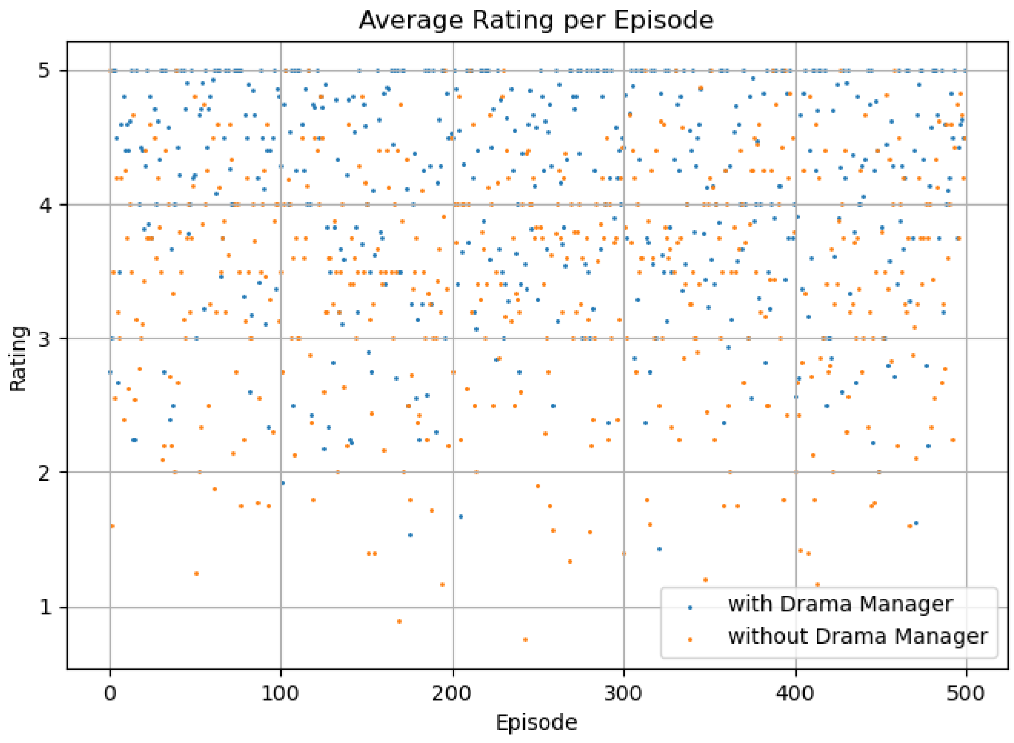


FIGURE 5.6: Average rating of stories (The user's liked path is populated with ratings of 5), L=50 topics

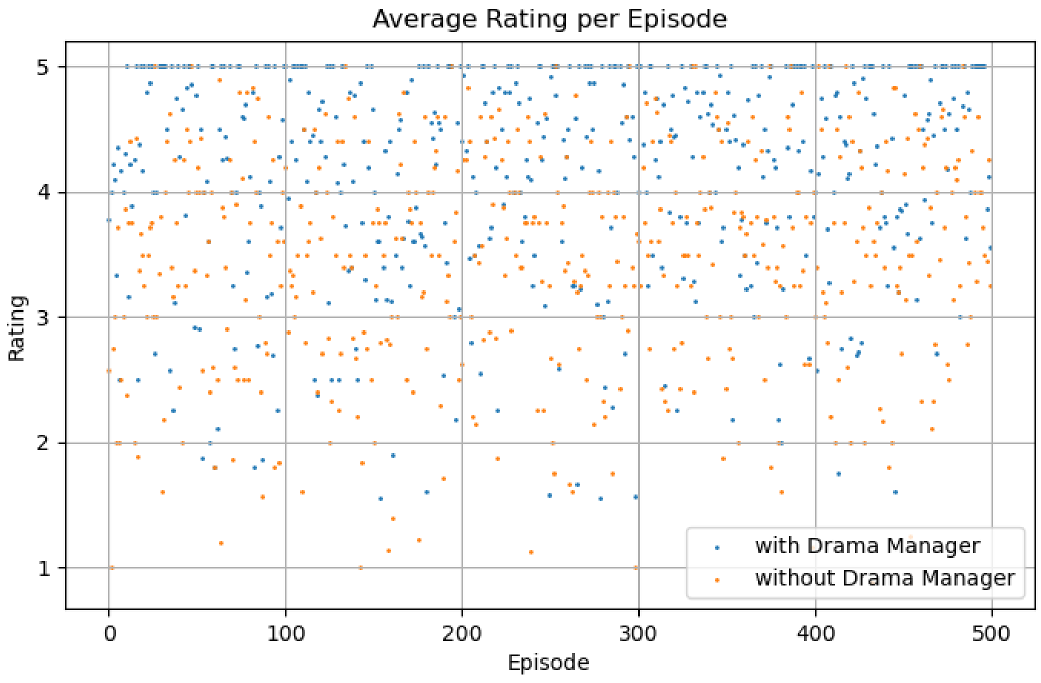


FIGURE 5.7: Average rating of stories (The user's liked path is populated with ratings of 5), L=60 topics

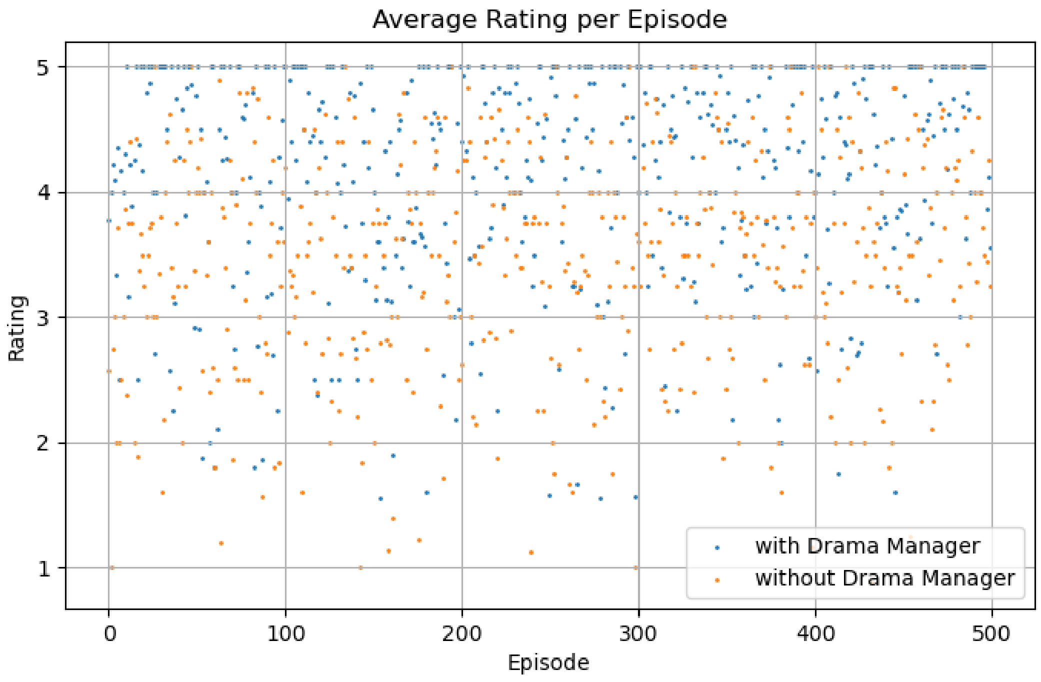


FIGURE 5.8: Average rating of stories (The user's liked path is populated with ratings of 5), L=70 topics

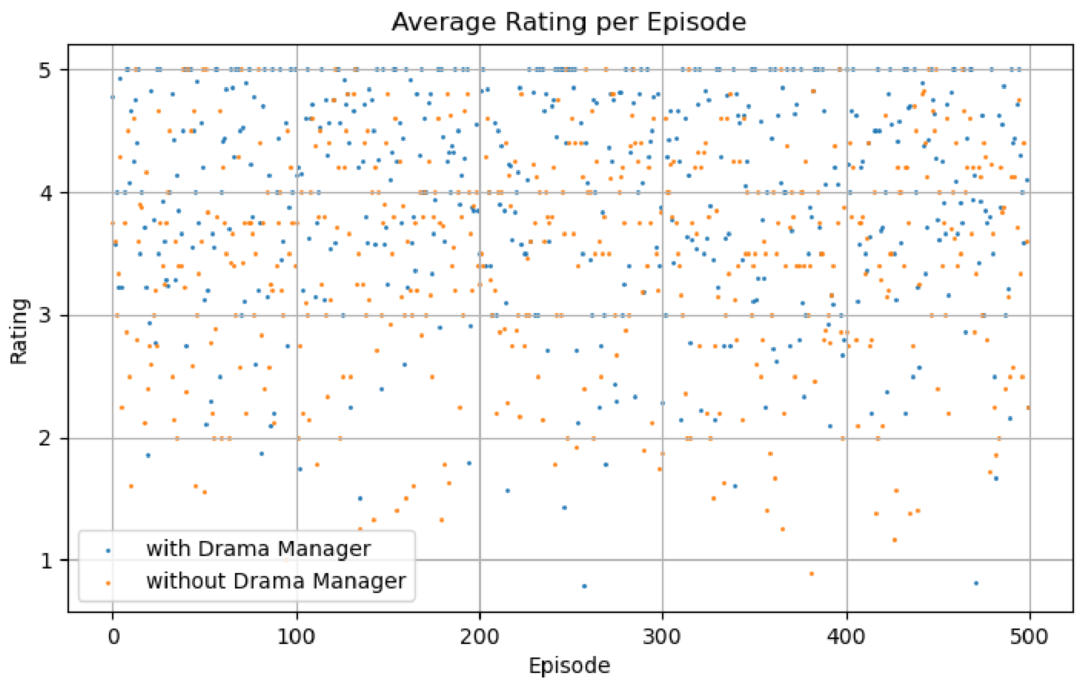


FIGURE 5.9: Average rating of stories (The user's liked path is populated with ratings of 5), L=80 topics

5.5 About Other Drama Manager Evaluation Methods

We heavily rely on the ratings of the simulated users as a performance metric for our system. User ratings are the best method to evaluate our system. In the current section, we explain why we did not use other methods.

As described in the previous sections, in our DM we model the user as an evolving document, which is a probability distribution over topics. Also, in order to create the simulated users, a hidden user model is generated for each user, which is used for providing the ratings of the simulated users. Both the user model inferred by the DM, and the hidden user model are probability distributions. Intuition says that probability distribution distance metrics (Kullback-Leibler divergence, Jensen-Shannon divergence) are useful in this setting, in order to discover the distance between the hidden user model and the user model the DM has inferred. It is reasonable to assume that in this way we can examine our DM's ability to approximate the user model.

This assumption has only limited application in our settings, however. The plot points of the users' liked paths are created based both on the user's hidden model and the user utility model (based on the ***weight*** vector described in Section 5.3). But the user's hidden model only guarantees that the user will rate the generated plot point with a high rating. For each one of the six utility intervals of our DM described in Section 5.3, there is a wide variety of topic proportions that is accepted. This is a realistic way to model the plot points, because a user is not capable of distinguishing small differences between two similar plot points. A real user's rating depends on the presence of the topics she likes, rather than the topic proportions themselves. We could create the liked path plot points in such a way, that the deeper into the story, the closer they are to the hidden user model, but this is a rather unrealistic way to construct a story. As we shown in the previous Sections, by using our DM a user will end up with a story that is within a greater story set of stories she likes. Distribution similarity metrics cannot be always accurate or capture the true distance from the real user preferences.

We examine a simple example. Assume the following two models (it is irrelevant if they are user or item models):

$$model_1 = [0, 0.1, 0, 0.2, 0.1, 0.4, 0.2, 0, 0, 0] \tag{5.3}$$

$$model_2 = [0.05, 0.05, 0.1, 0.1, 0, 0.3, 0.3, 0.1, 0, 0] \tag{5.4}$$

We calculate the Jensen Shannon divergence between $model_1$ and $model_2$.

$$JSD(model_1, model_2) = 0.37 \tag{5.5}$$

We remind the reader that JSD is bounded by 0 and 1.

If the models above correspond to two different plot points, the user would probably assign them the same rating, even though the JSD metric suggests that they are at a distance of $JSD(model_1, model_2) = 0.37$. This claim is based on the fact that a human user cannot distinguish a difference of 0.05 (topic 0 difference between models $model_1$

and $model_2$) or 0.1 (topic 5 difference between models $model_1$ and $model_2$). We remind the reader that each index of the vectors of the models above, corresponds to a topic i.e., index 0 to topic 0, index 1 to topic 1 and so on. With this example we aim to demonstrate, how conventional probability distribution similarity metrics fail to display the desired behaviour.

In Figure 5.10 we observe the average Jensen-Shannon divergence between the hidden user model, and the plot points the user likes (story rated with only ratings of five). Each of the 100 episodes shown is an independent simulation of our DM, with different user model and different plot points. We see that the distances are big, even though the user will rate those plot points with high ratings.

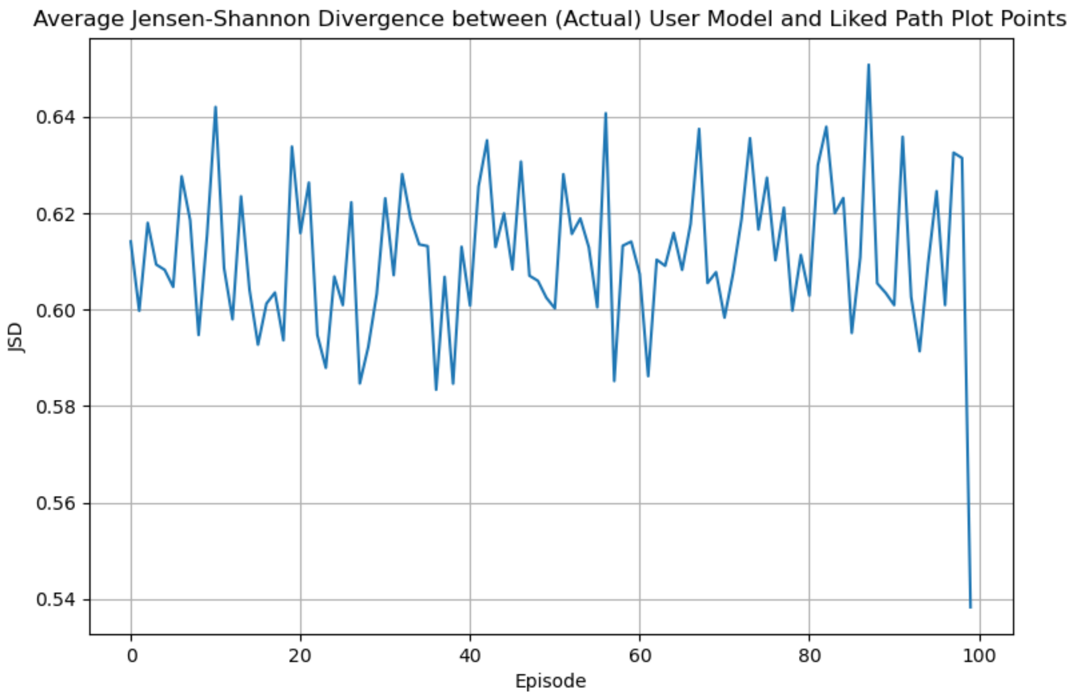


FIGURE 5.10: Average JSD between the true user model and the plot points of the user's likes path

In conclusion, because user ratings depend on a variety of topic proportions for each utility interval, rather than a single instance of topic proportions, the concept of tracking the probability distribution convergence does not readily apply.

Chapter 6

Conclusions & Future Work

In this thesis, we created a novel drama management approach. We employed probabilistic topic modeling for modeling items under recommendation. Also, in a novel approach inspired by [28], we also model an evolving user model as a mixture of latent topics. We treat the user as a document, whose topic proportions change according to the items the user consumes. We achieved this by using techniques from the reinforcement learning field. That is, by taking into account the user rating, we accelerate or decelerate the rate at which the user model changes. We provide an efficient parser for Wikipedia articles useful for the download and processing of the necessary training data for a PTM. Our DM does not rely on CF, and we proved it works with even one user. We introduced a novel metric for user-item model distance, *expected utility*. A command line utility is created tosplit possible merged words due to bad formatting of pdf files. Also, a method is pro-posed for automatic construction of the branching story graph by using the textual information.

As future work, we aim to study different datasets, to see if the performance of our dataset changes. Also, a more specific PTM algorithm can be implemented, designed exactly for the sequential recommendation problem. Moreover, we aim to automatically split merged words with high accuracy, in case the future work still includes CYOA books. There is also room for experiments with other than linearly increasing utility values between topic proportions of plot points. In would be interesting to also experiment with more than 6 (0, 1, 2, 3, 4, 5) rating values. The system should also be tested on human users, given the fact that adequate secret path books are found.

In our experiments, we assume that the user model prior to the interaction with our DM, is uniformly distributed. In future work, we aim to study if it is realistic to assume a prior user model that is closer to the actual user model than the uniformly distributed prior. This is not a difficult requirement, because as system designers, we know the topics discussed across the possible stories from our pool of stories. So, in a small interactive session the user can rate a small amount of items (chosen by the DM, in order to be relative to topics existing in the story space). The construction of such interactive session, along with the experimental result evaluation, is left for future work. Also, we do not experiment with multiple DM sessions of the same user (i.e., to use an already calculated user model as a prior model for other DM simulations).

Finally, we aim to tackle the sequential recommendation problem for larger story spaces. One such book showcasing a challenging problem of scalability for our DM, is the book *Infected: A Click Your Poison Book*. Its branching story graph is shown in Figure 6.1. The bottleneck of our DM is the transformation from a branching story graph into a prefix graph. The problem of finding all the possible paths of a graph (which is what we do in order to calculate the prefix graph), is NP-Hard. In future work, we aim to explore other graph transformation techniques, to be able to handle bigger stories.

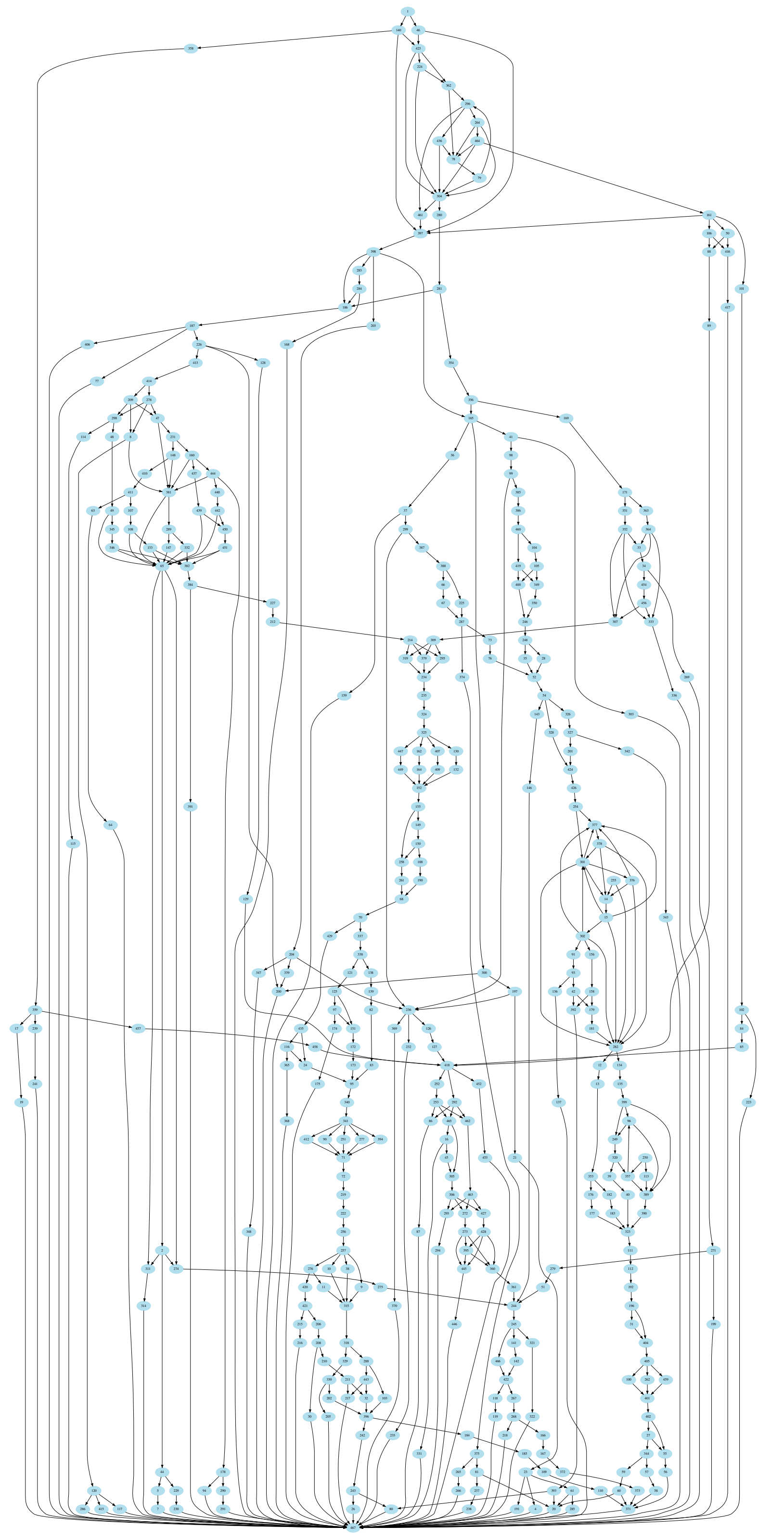


FIGURE 6.1: Branching story graph of the book *Infected*

Appendix A

Manual Topic Inspection of an LDA model with 80 topics

A.1 Model Inspection

In this appendix, we will examine a model with $K = 80$ topics.

In each bullet, we provide the most significant words, along with a description that we suggest:

Topics are listed by decreasing percentage of tokens with respect to the dataset

1. **Book writing**, 6.1%: characters, character, literary, narrative, literature, themes,narrator, style, theme
2. **Undefined**, 5.9%: tells, house, finds, night, goes, home, away, gets, room, takes, asks
3. **Battles**, 5.6%: city, escape, group, death, killed, help, power, kill, takes, return, attack, fight, battle, captured, leader,
4. **Family**, 4.9%: father, mother, love, family, daughter, brother, sister, woman, young, wife, friend, child, death, husband, relationship, marriage, born
5. **Philosophy-Argumentation**, 3.8%: argues, theory, view, wrote, history, argued, described, study, social, criticized, evidence, claims, argument, critisism, philosopher,
6. **Management**, 3.8%: chapter, social, society, change, development, mastery, drift, problems, problem, future, example, based, theory, technology
7. **Book publishing**, 3.5%: edition, volume, text, history, editions, pages, second, page, version, original, volumes, chapter, chapters, english
8. **British novels**, 3.1%: london, british, lady, wife, house, england, young, money, british novels, husband, english, friend, marriage, servant
9. **Science fiction**, 2.7%: science, earth, science fiction, planet, space, human, humans, future, universe, alien, travel, technology, humanity, race, solar, time travel, colony, robots, ship

10. **Publishing**, 2.6%: review, reviews, writing, wrote, reception, publishers, critical, positive, weekly, recieved, praised, reviewer, journal, library

11. **Media**, 2.6%: said, media, article, interviews, press, journalist, interview, news, magazine, public, fiction books, wrote, controversy, publication, editor, newspaper, release

12. **Spying**, 2.5%: intelligence, agent, secret, thriller, security, team, officer, kill, powell, information, killed, police, chief, prison, operation, agency

13. **Film adaptation of book**, 2.2%: film, adapted, adaptation, films, television, directed, movie, based, starring, released, novels adapted, version, production, american novels, screenplay, hollywood, television series

14. **Political-Financial (US oriented)**, 2.1%: states, united, political, economic, united states, government, policy, money, capitalism, power, economy, great, america, national, wealth, market

15. **Existentialism**, 2.1%: human, concept, philosophy, love, individual, nature, mind, philosophical, self, experience, good, knowledge, reason, thought, soul, meaning, consciousness, sence, evil, truth, question, reality, spirit, existence

16. **Script (maybe?)**, 1.9%: wrote, writing, letters, manuscript, death, letter, publication, literary, thumb, read, write, according, began, file

17. **Undefined**, 1.8%: anxiety, good, know, says, things, said, want, word, feel, think, person, words, right, wanted, able, voice

18. **City (New York oriented)-American novels**, 1.7%: city, york, narrator, New York, street, drug, suicide, american novels, apartment, hotel, house, gang, angeles, money, drugs

19. **Family-village life (positive feelings)**, 1.6%: family, town, children, village, home, father, house, farm, families, land, child, lives, mother, local, wife, small, community, living, grandfather, young, farmer, journey, country

20. **High school**, 1.5%: school, young, adult, young adult, year, friends, teacher, boys, parents, students, girl, high school, library, college, class, education, teachers, summer, grade

21. **New York Times (NYT) bestsellers**, 1.5%: list, york, times, released, york times, copies, random, sold, house, bestseller, million, seller, year, release, weeks, sales, ranking

22. **Army-Military**, 1.5%: army, military, battle, general, soldiers, forces, soldier, officer, force, commander, major, troops, invasion, attack, command, enemy, fighting, corps, service, warfare, voctory, civil, killed, navy

23. **Crime investigations**, 1.4%: murder, crime, police, detective, mystery, case, death, killer, killed, dead, murders, body, inspector, criminal, evidence, missing

24. **Trip to the country**, 1.4%: land, river, water, travel, journey, tree, great, mountain, valley, lake, birds, fishing, bird, flies, wind, island, forest, wild, north, bear, mountains, wood, expedition

25. **Monthly subscriptions**, 1.2%: times, york, review, retrieved, york times, june, november, december, guardian, diary, daily, wrote, diaries, journal,year

26. **Horror-terror-dark**, 1.2%: horror, dream, great, dark, dreams, greek, lost, dead, bridge, ancient, death, evil, supernatural, mysterious, revival, strange, secret, darkness, mythology, greece, gothic, myth, past

27. **Academia-university press**, 1.2%: university, press, oxford, political, cambridge, philosophy, university press, essays, studies, york, professor, instoduction, politics, college, princeton, society, harvard, academic, literature

28. **Doctor Who**, 1.1%: doctor, episode, serial, television, episodes, doctors, tardis, released, broadcast, british, season, production, original, planet

29. **Collections of books**, 1.1%: stories, short, short story, collection, anthology, edited, introduction, tales, writed, asimov, originally, essay, lovecraft, story collections, isaac asimov, contains

30. **Sailing-adventure**, 1.1%: ship, island, crew, ships, boat, treasure, aboard, british, navy, islands, pirates, pilot, storm, voyage, coast, flying, port, fleet, sail, ocean, adventure, royal

31. **Colonialism**, 1.1%: white, african, culture, race, history, america, native, states, african american, americans, racial, cultural, identiry, community, racism, africa, colonial, movement, violence

32. **Children books**, 1.1%: children, children books, little, illustrations, picture, illustrated, animals, literature, characters, circus, fish, frog, children literature, young, girl, home, parents

33. **Religious affairs**, 1.1%: church, religious, religion, christianity, catholic, faith, century, protestant, christ, ancient, spiritual, rome, history, miracles, holy, spirit, bishop, temple, bible, divine, pope, theology, saint, priest

34. **Magazines (But it is not clear)**, 0.9%: miss, edition, club, magazine, york, dust, issue, title, viking, publication, editions, press, appeared, hardcover

35. **World War 2 (Holocaust oriented)**, 0.9%: jewish, polish, germany, jews, history, nazi, israel, holocaust, poland, europe, massarce, historical, jerusalem, peace, hitler, israeli, nazis, european, international, eastern

36. **World War 2 (alternate history)**, 0.9%: british, united, british novels, london, britain, kingdon, united kingdom, depictions, minister, cultural, history, cultural depictions, prime, england, states hitler, germany, france, leader, alternate history, great britain

37. **Book Translations**, 0.9%: french, english, translation, translated, language, paris, spanish, italian, france, dutch, enslish translation, title, languages, swedish, literature, original, translator

38. **Communism-Russia**, 0.9%: russian, soviet, revolution, political, union, russia, class, revolutionary, communist, soviet union, moscow, socialist, party, history, power, government, regime, fascism, left, marxist, propaganda, marxism, republic, movement, anarchist

39. **Fantasy-Dungeons and Dragons**, 0.8%: fantasy, magic, fantasy novels, magical, sword, fantasy novel, trology, grey, american fantasy, magician, plot, dar, city, quest, powers, magicians, mage, wizards, spider, gods, powerful, evil, fantastic

40. **Legislation**, 0.8%: state, court, states, laws, constitution, justice, legal, public, government, judge, united, rights, united states, civil, authority, rules, congress, rule, supreme, cases, members

41. **Natural selection-biology**, 0.8%: human, species, evolution, natural, humans, animals, selection, nature, plants, evolutionary, enviromental, animal, biological, biology, genetic, plant, natural history, population, genes, biologist

42. **Medicine**, 0.8%: health, medical, mental, hospital, brain, medicine, disease, patients, patient, aids, cancer, mind, body, care, phychiatrist, treatment, psychology, doctors, disorder, research, illness, memory, meditation

43. **Homosexuality-Sex**, 0.8%: women, sexual, woman, female, feminist, male, gender, homosexuality, sexuality, feminism, relationship, lgbt, rape, love, homosexual, marriage, lesbian, desire, roles, rights, relationship, erotic, masculinity, power, beauty

44. **Cities of U.S. (related to sports?)**, 0.8%: boston, chicago, california, team, road, baseball, florida, ohio, county, sports, pennsylvania, football, massachusetts, league, state, york, corn, town, national, season

45. **Wizardry**, 0.7%: castle, witch, stone, wizard, wiches, horse, tower, wicked, lion, statue, heart, woman, magic, magical, glass, emerald, witchcraft, dead

46. **Scientific work**, 0.7%: science, scientific, research, theory, professor, physics, scientists, scientist, institute, newton, mathematics, design, university, universe, intelligent, history, knowledge, sciences, physicist, mathematical, light, experiments, astronomy, motion, discovery, experiment, quantum

47. **Unrecognized**, 0.7%: science, science fiction, originally, fantasy, magazine, originally published, galaxy, campbell, worlds, edition, astounding, american writer

48. **Music scene**, 0.7%: music, song, band, rock, album, songs, singer, musical, food, golden, lyrics, dance, jazz, tune, record, popular, artists, composer

49. **China-Australia relations (Book:Silent Invasion)**, 0.7%: chinese, japanese, australian, china, australia, japan, asia, south, west, western, history, country, domination, kong, east, hong kong, tokyo, melbourne, eastern, shanghai

50. **Historical novels**, 0.6%: historical, century, history, period, england, historical novels, records, early, events, death, late, kings, chronicle, royal, eighteenth, british, rulers

51. **Dungeons and Dragons**, 0.6%: dragon, dragons, monster, dungeons, game, rules, players, guide, characters, campaign, role, playing, history, edition, setting, realms, adventure, masters, mage, spells, wizards, forgotten

52. **Poetry**, 0.6%: poem, poetry, poems, poet, irish, collection, song, verse, ireland, love, poets, lines, epic, literature, night, dublin, poetic, poetry collections

53. **Awards**, 0.6%: award, awards, winning, year, nominated, prize, book award, winner, award best, winning works, poll, shortlisted, award winning, finalist, nominations, literary, association, awards nominations, placed, nominee, nebula award, writers, choice

54. **Theatre**, 0.5%: play, theatre, stage, plays, shakespeare, musical, opera, production, performed, theater, broadway, london, actor, drama, performance, company, actors

55. **Unrecognized**, 0.5%: blue, page, hair, roberts, eyes, white, skin, born, glass, sister, group, member, rainbow, kansas, special, pretty, characters, looking, wears, website, lone, pink

56. **US relations with the Islamic world**, 0.5%: president, islam, muslim, bush, iran, islamic, iraq, trump, political, washington, presidential, campaign, arabic, election, tessorism, middle, states, iranian, terrorist, afghanistan, muslims, arab, presidency, house, attacks, white house, vice, senator

57. **Games-Africa (cannot relate them though)**, 0.4%: game, games, video, south, park, africa, player, video game, playing, south africa,chess, play, shadow, parody, entertainment, south african, video games, released, zero, african, interactive, lost, original

58. **Nuclear energy (post-apocalyptic oriented)**, 0.4%: nuclear, energy, global, post, bomb, united, states, climate, power,apocalyptic, hook, weapons, rocket, united states, north, atomic, electric, radiation, post apocalyptic, zone, nuclear weapons, survivors, explosion, survival, arctic, bombs

59. **Publishing**, 0.4%: paperback, edition, hardcover, publication, hardback, norton, editions, september, cover, audio, press, publication history, publishing, june, march, july, edition published

60. **Biography**, 0.4%: biography, fiction books, memoir, account, describes, autobiography, biographies, personal, details, early, covers, autobiographies, experiences, events, carrees, childhood, biographical, memories, writing

61. **Slavery**, 0.4%: slave, civil, slavery, slaves, walker, south, southern, rights, north, uncle, civil rights, history, freedom, movement, negro, free, united, sold

62. **Kings and queens**, 0.4%: prince, princess, kingdom, royal, knight, saint, palace, kings, crown, knights, court, throne, heaven, hell, noble, ruler, priest, lady, royal family, medieval, heir

63. **Colonialism**, 0.4%: india, indian, wells, oregon, pakistan, hindu, society, history, zealand, delhi, caste, indias, colonial, massacre, language, haven, independence, press, rule

64. **Galactic empire (korean, korea unrelated)**, 0.3%: empire, emperor, master, moon, north, ranking, imperial, korean, korea, grand, masters, apollo, mission, human, republic, male, commander, leader, state, lunar, force, alliance, rebels, galactic

65. **Star Trek-Canada**, 0.3%: star, canadian, canada, graphic, wars, trek, star trek, graphic novel, persian, enterprise, federation, canon, prequel, generals, fictional, pocket

66. **Vampires**, 0.3%: blood, vampire, vampires, midnight, kill, human, beast, werewolf, horror, hole, atlas, turn, bloody, cold, night, turned, american horror

67. **Unrecognized**, 0.2%: camp, tale, rabbit, potter, tales, mouse, sprague camp, whale, camps, fisher, wine, elephant, doll, racing, grass, gollancz

68. **Unrecognized**, 0.2%: queen, brothers, egypt, queens, brother, foster, egyptian, thief, bacon, eden, arthurs

69. **Workers of Mexico**, 0.2%: ghost, mexico, train, station, mexican, factory, ghosts, workers, iron, railway, ward, hunger, border, construction, latin, bread, strike, industrial, canal, invisible, realism, working, savage, worker, rain, trains

70. **Unrecognized**, 0.2%: brown, jones, wolf, fairy, little brown, snow, rogers, greene, wolves, miles, little, tale, fairy tale

71. **Prize about a book for Vietnam (Unclear)**, 0.2%: prize, vietnam, national, adams, foundation, christmas, pulitzer, knopf, vietnamese, national book, alfred knopf, winning, shift, awarded, prize fiction, america, veterans

72. **Unrecognized**, 0.2%: bond, animals, animal, bone, cape, mountain, bones, jonathan cape, brooks, hodder, bonds, dinosaurs, hodder stoughton, dinosaur

73. **Unrecognized**, 0.2%: holmes, comic, comics, adventure, sherlock holmes, adventures, watson, tiger, comic book, chase, marsh, shadows, marvel, strip

74. **Unrecognized**, 0.2%: hill, report, commonwealth, hills, cloud, coal, review, ranger, mining, based, mark twain, silent, reports, collision, reviews
75. **Unrecognized**, 0.1%: earth, ring, texas, hunting, smiths, giant, rings, flood, genesis, middle, wagner, rising
76. **Unrecognized**, 0.1%: bible, thompson, pseudonym, luther, georgia, code, baker, print, confessions, testament, martin luther, sata, codes, examiner, biblibal
77. **Unrecognized**, 0.1%: hart, influential, cross, bell, moore, style, salt, merry, handbook, history, noter, person, huffington
78. **Unrecognized**, 0.1%: card, clan, dogs, cards, cats, warriors, clans, sheep, charity, shepherd, bang, pets, called
79. **Unrecognized**, 0.1%: arts, martial, philips, colony, seed, harvest, devils, band, fight, pact, mass, coincide, matrins press, blade
80. **Unrecognized**, 0.1%: machine, machines, rivers, miracle, pilgrimage, baltimore, miraculous, phantom, hyperion, cotton, remake, booth, barrier, rewrite, abraham lincoln, embark, critically acclaimed

Appendix B

Dealing with merged words from the PDF parsing

B.1 Merged Words

In this section, we provide all the merged words we encountered, along with the word split offered by the compound-word-splitter python package, and also the original word split.

Merged Words	compound-word-splitter	Original Meaning
vesselseeker	vessels eek er	vessel seeker
theresearch	Therese arch	the research
maray	maray	mara y
ofthe	oft he	of the
astrong	as tron g	a strong
the seeker	there eke r	the seeker
adark	Adar k	a dark
theairlock	Thea IR lock	the airlock
tothe	tot he	to the
giantsquid	giants quid	giant squid
inthe	int he	in the
themoray	them Ora y	the moray
onthe	ont he	on the
theseeker	these eke r	the seeker
yousome	yous om e	you some
asubmarine	(blank)	a submarine
avulerable	(blank)	a vulnerable
youthink	youth ink	you think
theabyss	Thea by s	the abyss
ahalf	ah Al f	a half
guage	(blank)	guage
torise	tor is e	to rise
toolate	tool ate	too late
insimple	ins imp Le	in simple
youthat	youth at	you that

theatlanteans	Thea Tl ante ans	the antlanteans
atlantean	(blank)	atlantean
forthe	forth e	for the
english	Eng Li sh	english
yoursea	yours ea	your sea
toleave	tole ave	to leave
thewreck	thew rec k	the wreck
toreport	tore port	to report
youstill	yous till	you still
cometo	comet o	come to
haveno	haven o	have no
otherspaces	others paces	other spaces
thenodoors	the no doors	the nodoors
abeautiful	(blank)	a beautiful
achance	ac Han Ce	a chance
asensing	as ens in g	a sensing
haveskin	haves kin	have skin
aglasslike	Ag lass like	a glass-like
maraythat	ma ray that	maray that
themaray	them Ara y	the maray
toescape	toes cape	to escape
ahole	ah ole	a hole
theearth	thee art h	the earth
goback	gob ac k	go back
toreturn	tore turn	to return
foranother	fora not her	for another
nodoors	no doors	nodoors
asmall	as mall	a small
amistake	am is take	a mistake
yoursenses	yours ens es	your senses
advisor	adv is or	advisor
theactors	Thea ct or s	the actors
marayis	ma ray is	maray is
nowthat	nowt hat	now that
theelectrons	thee LE ctr on s	the electrons
bethe	Beth e	be the
tojoin	Tojo in	to join
soclose	soc lose	so close
willseek	wills eek	will seek
arich	(blank)	a rich
workschedules	works Che Du Le s	work schedules
largesalary	larges Alar y	large salary
thework	thew or k	the work

togo	tog o	to go
facethat	facet hat	face that
notworkers	notwork er s	not workers
thesea	these a	the sea

Appendix C

System Parameters

In this section, we provide a table containing all the parameters of our system.

LDA related parameters	
Parameter	Value
Lemmatizing	False
Stemming	False
LDA models trained	5 to 390 with step of 5
no_below [1]	100
no_above [2]	0.35
low	0.3
n-fold validation for topic quality	5
Topic Coherence	True
Jaccard Similarity	True
Perplexity	True
allow bigrams	True
allow n-grams with n>2	False
remove first names from dataset	True
multiprocessing for quick preprocessing	4 threads
Drama Manager related parameters	
Number of topics	80
Number of ratings	6 (0,1,2,3,4,5)
Delta	4
δ_{lose}	0.3
δ_{win}	0.05
c	1.1
ξ	3
user utility function	2 to K with step of 1
linear user utility	True

[1]Keep tokens which are contained in at least no_below documents for the LDA model training

[2]Keep tokens which are contained in no more than no_above documents (fraction of total corpus size, not an absolute number) for the LDA training

Appendix D

Modeling and Tackling Preference Shifts

In general, a DM must have the ability to quickly identify the eventual preference shifts of a user. To prove our point, we created a separate testbed.

We experimented in a test case consisting of a sequence of 240 plot points, and we choose the number of topics L for each plot point to be $L = 80$ (the number L is unimportant in the preference shift tackling, and can be easily discovered with a trial and error procedure). We suppose that each plot point is an amalgamation of three to seven (for each iteration, this is chosen randomly) topics. So, we created 240 random vectors, each of which has from three to seven non-negative values summing to one, ensuring that the item model can be treated as a discrete probability distribution. This way we model the topic proportions for each plot point.

We model the user's initial preferences with the vector

$$\boldsymbol{weights}_{pre\_shift} = [K+1, K, ..., 3, 2]$$

whereas we model the user's preference shift as

$$\boldsymbol{weights}_{post\_shift} = [2, 3, ..., K, K+1]$$

We consider the following scenario. A user is interacting with our DM in a session of 240 plot points. In the first 60, we model the convergence of the user model to the actual user preferences. After the 60^{th} plot point, every 60 plot points a major preference shift occurs. Specifically, we create the user $\boldsymbol{weights}$ accordingly such that

$$D(\boldsymbol{weights}_{pre_{shift}}, \boldsymbol{weights}_{post_{shift}}) > T$$

where D is the L_2 norm, and $T = 380$. The value of T was a result of several experiments, and is dependent of the number of topics, K. With this setting, we create a major preference shift, meaning that the user starts hating the previously adored topics, and vice versa. By examining the top two subplots of D.1, we can infer that the user indeed likes what is being recommended to him. But we cannot identify if the items recommended are similar. To tackle this problem, we keep the last W known user models, and calculate their mean value. Whenever the user consumes an item, we calculate the *Jensen Shannon divergence* between mean of the last W known user models (excluding

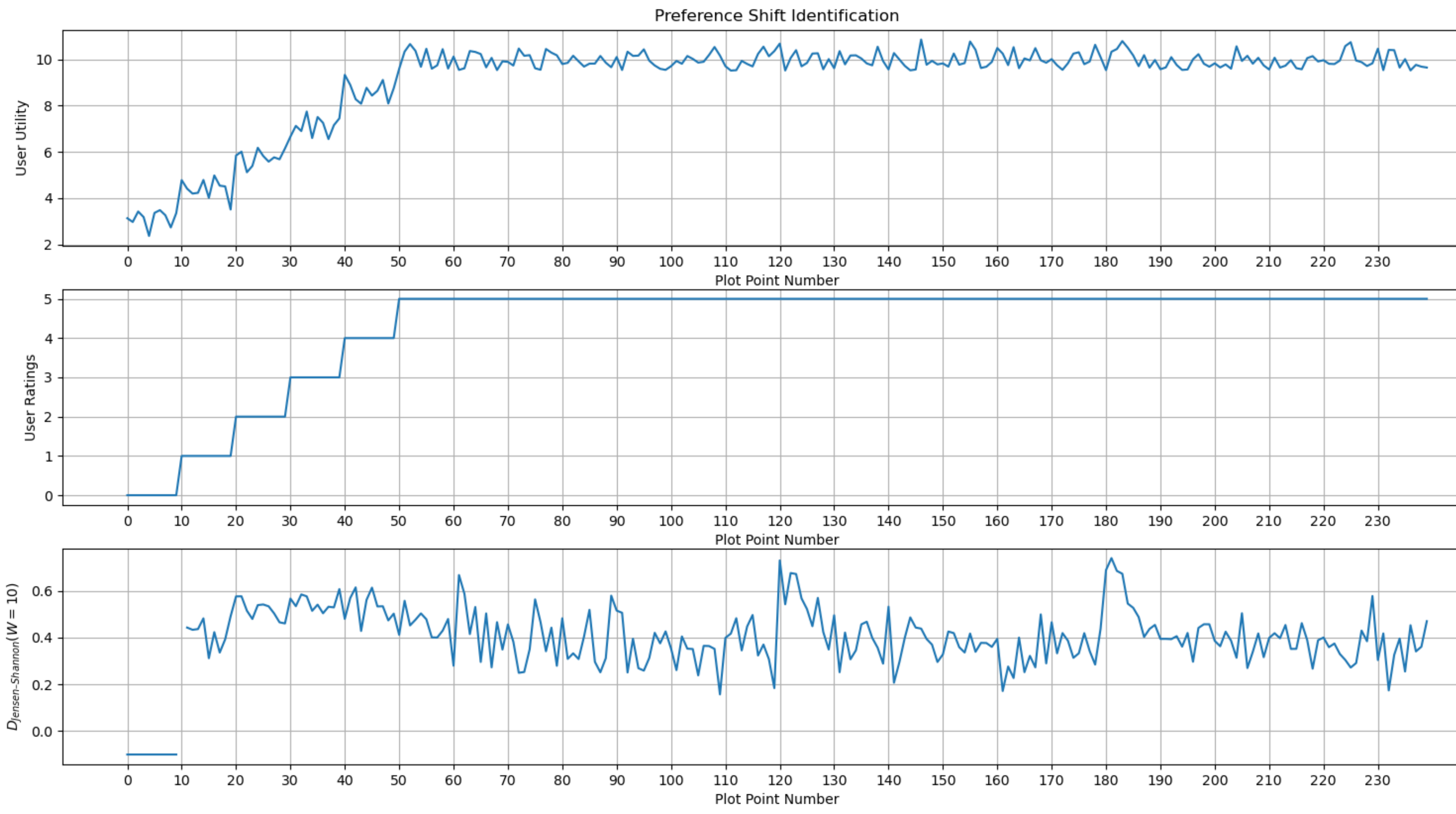


FIGURE D.1: Identification of major preference shifts

the last user model), and the last user model. In the third subplot of figure D.1, we notice three spikes, at the 60th, 120th and 180th plot point. Those are the plot points where we placed the major preference shifts. So, with this technique, we were able to identify changes in the user model, even though the rating pattern of the user did not change.

Bibliography

[1] Joseph Bates. "Virtual reality, art, and entertainment". In: *Presence: Teleoperators & Virtual Environments* 1.1 (1992), pp. 133–138.

[2] Sooraj Bhat et al. "A globally optimal algorithm for TTD-MDPs". In: *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. 2007, pp. 1–8.

[3] David M Blei. "Probabilistic topic models". In: *Communications of the ACM* 55.4 (2012), pp. 77–84.

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[5] Michael Bowling and Manuela Veloso. "Multiagent learning using a variable learning rate". In: *Artificial Intelligence* 136.2 (2002), pp. 215–250.

[6] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[7] Thomas Hofmann. "Probabilistic latent semantic indexing". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999, pp. 50–57.

[8] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008. ISBN: 0521865719.

[9] Magy Seif El-Nasr. "Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories". In: *Interaction Studies* 8.2 (2007), pp. 209–240.

[10] Magy Seif El-Nasr. "Interactive Narrative Architecture based on Filmmaking Theory." In: *Int. J. Intell. Games & Simulation* 3.1 (2004), pp. 29–36.

[11] Mark J Nelson and Michael Mateas. "Search-Based Drama Management in the Interactive Fiction Anchorhead." In: *AIIDE*. 2005, pp. 99–104.

[12] Thaleia Ntiniakou. "A Framework for Employing Probabilisc Topic Models on Gene Expression Data". In: *Technical University of Crete, Institutional Repository* (2019).

[13] Federico Peinado and Pablo Gervás. "Transferring game mastering laws to interactive digital storytelling". In: *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*. Springer. 2004, pp. 48–54.

[14] Jim Pitman. "Exchangeable and partially exchangeable random partitions". In: *Probability theory and related fields* 102.2 (1995), pp. 145–158.

[15] Gerald Prince. *A dictionary of narratology*. U of Nebraska Press, 2003.

[16] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. "Sequence-aware recommender systems". In: *ACM Computing Surveys (CSUR)* 51.4 (2018), pp. 1–36.

[17] Mark Owen Riedl and Vadim Bulitko. "Interactive narrative: An intelligent systems approach". In: *Ai Magazine* 34.1 (2013), pp. 67–67.

[18] Mark Riedl, Cesare J Saretto, and R Michael Young. "Managing interaction between users and agents in a multi-agent storytelling environment". In: *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. 2003, pp. 741–748.

[19] Mark Riedl, David Thue, and Vadim Bulitko. "Game AI as storytelling". In: *Artificial intelligence for computer games*. Springer, 2011, pp. 125–150.

[20] David L Roberts et al. "Targeting specific distributions of trajectories in MDPs". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 21. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, p. 1213.

[21] Andrew Rollings and Ernest Adams. *Andrew Rollings and Ernest Adams on game design*. New Riders, 2003.

[22] Alexandra Schofield et al. "Understanding text pre-processing for latent Dirichlet allocation". In: *Proceedings of the 15th conference of the European chapter of the Association for Computational Linguistics*. Vol. 2. 2017, pp. 432–436.

[23] Manu Sharma et al. "Player modeling evaluation for interactive fiction". In: *Proceedings of the AIIDE 2007 Workshop on Optimizing Player Satisfaction*. 2007, pp. 19–24.

[24] Carson Sievert and Kenneth Shirley. "LDAvis: A method for visualizing and interpreting topics". In: *Proceedings of the workshop on interactive language learning, visualization, and interfaces*. 2014, pp. 63–70.

[25] Keith Stevens et al. "Exploring topic coherence over many models and many topics". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 2012, pp. 952–961.

[26] Richard S Sutton. "Temporal Credit Assignment in Reinforcement Learning." In: (1985).

[27] David Thue et al. "Interactive Storytelling: A Player Modelling Approach." In: *AIIDE*. 2007, pp. 43–48.

[28] Evangelos Tripolitakis and Georgios Chalkiadakis. "Probabilistic topic modeling, reinforcement learning, and crowdsourcing for personalized recommendations". In: *Multi-Agent Systems and Agreement Technologies*. Springer, 2016, pp. 157–171.

[29] Peter Weyhrauch and Joseph Bates. *Guiding interactive drama*. Carnegie Mellon University Pittsburgh, PA, 1997.

[30] Ian H Witten and Eibe Frank. "Data mining: practical machine learning tools and techniques with Java implementations". In: *Acm Sigmod Record* 31.1 (2002), pp. 76–77.

[31] R Michael Young. "Creating interactive narrative structures: The potential for AI approaches". In: *Psychology* 13 (2000), pp. 1–26.

[32] R Michael Young et al. "An architecture for integrating plan-based behavior generation with interactive game environments." In: *J. Game Dev.* 1.1 (2004), pp. 1–29.

[33] Hong Yu and Mark O Riedl. "A sequential recommendation approach for interactive personalized story generation." In: *AAMAS*. Vol. 12. 2012, pp. 71–78.

[34] Hong Yu and Mark Owen Riedl. "Data-Driven Personalized Drama Management." In: *AIIDE*. Citeseer. 2013.

[35] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).