NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Movie Recommendation System based on Dempster-Shafer theory

Ilias B. Papasotiriou

Supervisor: **Isambo Karali,** Assistant Professor

ATHENS

JUNE 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα συστάσεων ταινιών βασισμένο στην θεωρία Dempster-Shafer

Ηλίας Β. Παπασωτηρίου

Επιβλέπων: **Ιζαμπώ Καράλη,** Επίκουρη Καθηγήτρια

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2021

BSc THESIS

Movie Recommendation System based on Dempster-Shafer theory

Ilias B. Papasotiriou
S.N.: 1115201500123

SUPERVISOR: **Isambo Karali,** Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα συστάσεων ταινιών βασισμένο στην θεωρία Dempster-Shafer

Ηλίας Β. Παπασωτηρίου
Α.Μ.: 1115201500123

ΕΠΙΒΛΕΠΩΝ: **Ιζαμπώ Καράλη,** Επίκουρη Καθηγήτρια

ABSTRACT

In this thesis, we study the subject of Handling Uncertainty in Recommendation Systems. We implemented a Collaborative Filtering movie recommendation system using the Python programming language. We use Dempster-Shafer theory to propagate the uncertainties arising from imperfections in user ratings to the decision-making process. By converting ratings into mass functions and pignistic probabilities, we measure the similarities between users and use the Dempster's rule of combination to predict user ratings in movies they have not rated.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: Dempster-Shafer theory, Uncertainty, Recommendation system, Python

ΠΕΡΙΛΗΨΗ

Σε παρούσα πτυχιακή, μελετάμε το θέμα του Χειρισμού της Αβεβαιότητας στα Συστήματα Συστάσεων. Υλοποιήσαμε ένα σύστημα συστάσεων για ταινίες με τη μέθοδο Collaborative Filtering χρησιμοποιώντας τη γλώσσα προγραμματισμού Python. Χρησιμοποιούμε τη θεωρία Dempster-Shafer για να μεταφέρουμε τις αβεβαιότητες που πηγάζουν από τις ατέλειες στις αξιολογήσεις χρηστών στη διαδικασία λήψης αποφάσεων. Μετατρέποντας τις βαθμολογίες σε συναρτήσεις μάζας και pignistic πιθανότητες μετράμε τις ομοιότητες μεταξύ των χρηστών και χρησιμοποιούμε τον κανόνα Dempster's rule of combination για να προβλέψουμε την βαθμολογία των χρηστών σε ταινίες που δεν έχουν αξιολογήσει.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Θεωρία Dempster-Shafer, Αβεβαιότητα, Συστήματα συστάσεων, Python

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Mrs. Isambo Karali for providing me with this inspiring topic and for her consistent support and guidance. This thesis would not have been possible without her help through countless online meetings and emails. Also, I would like to express my appreciation to my family and friends for their support during this time.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In this day and age, with the rapid development of the internet, online sales continue to grow exponentially. The term coined to describe online sales is e-commerce. Small online stores up to large corporations on the internet are flourishing, with companies using every means to better serve their online customers. A problem for these companies is the enormous size of their products and their potential customers. Online stores fight not to be absorbed into the enormous competition of the internet and try to keep their products up to date and aimed at their consumers. In this chaotic context, using huge amount of information, small and large businesses are trying to personalize their online stores for each user who may have very different preferences. For this reason, the so-called Recommendation Systems (RS) [9] have been developed, which are systems that aim offering users products according to their preferences.

Due to the current development of e-commerce, Recommendation Systems are at the hotspot of research. RS's task is to gather information about the general preferences of the users and suggest products that meet these specifications. The most common method that RS display information about users' preferences is through their ratings and reviews. The main approaches are either content-based or collaborative filtering. Content-based methods try to combine users' preferences and match them to existing product categories. While collaborative filtering methods find similar users and assume that in the future they will have similar preferences. However, these methods are not perfect as the scores can lead to uncertain and imprecise results. Thus, systems, in order to handle the uncertainty of user data, make assumptions that impair system performance.

One framework designed to reason with uncertainty is Dempster-Shafer theory [6], [17]. The theory is a generalization of the Bayesian theory of subjective probability, which was firstly introduced by Arthur P. Dempster and later developed by Glenn Shafer. This framework uses a function, referred to as mass function or basic probability assignment, in order to represent the exact body of belief over a set of events. Also, additional measures, belief and plausibility are used to express how likely or plausible an event is; and their range interval expresses the range of uncertainty of the given event. Moreover, Dempster-Shafer theory is widely used due to its ability to combine independent sources of evidence. This tool is called the Dempster rule of combination and it is popular in statistical and classification problems.

One application where Dempster-Shafer theory can be used is Recommendation Systems [13], [11], [21]. Due to the imprecise user input, RS is possessed by uncertainty. For this reason, methods have been proposed using Dempster-Shafer theory to handle uncertainty.

In this thesis, we implement a movie recommendation system using Dempster-Shafer theory. Movie RS is a classic RS problem initially since there are plenty of data with user ratings and, on the one hand, because it is a major problem since there is a huge size of active users and movies. The importance of the RS movie is also evident from the Netflix competition [9] where they offered a prize of one million dollars for the first algorithm that could outperform their recommendation system.

The purpose of our application is to process user data considering the imperfections and uncertainties contained, as well as to propagate them to the final result.

The thesis is structured as follows. The next chapter describes in more detail the Dempster-Shafer theory. The Chapter 3 illustrates the way that RS work as well as some existing RS methods combined with the Dempster-Shafer theory. Next, Chapter 4 describes our implementation and the results we reached. Finally, we present our conclusions and proposals for future work.

2. DEMPSTER SHAFER THEORY

Dempster-Shafer theory (DST) [6], [17], also referred to as evidence theory or theory of belief functions is a general framework that first introduced by Arthur P. Dempster in context of statistical inference and later developed by Glenn Shafer in a general framework of epistemic uncertainty. The main possibilities of Dempster-Shafer theory are a) obtaining degrees of belief for subjective probabilities of a specific question and b) the combination of degrees of belief using Dempster's rule of combining, when they are based on independent evidence.
Dempster-Shafer theory is a generalization of the Bayesian theory of subjective probability and was developed for handling uncertainty. Uncertainty does not mean ambiguity or vagueness but expresses a degree of belief that may arise from a random process or lack of knowledge [18].

2.1 Definitions

We define Universe (U) or frame of discernment (Θ) as the set of all possible states [17], [18], therefore the powerset 2^Θ is the set of all subsets of Θ, containing the empty set ($\emptyset$).

Definition 2.1.1: (Mass Function)
For a frame of discernment Θ a function $m : 2^\Theta \to [0, 1]$ is called mass function or basic probability assignment (bpa) if:

1) $$m(\emptyset) = 0$$

2) $$\sum_{A \in \Theta} m(A) = 1$$

Then for a set $A \in 2^\Theta$ the quantity m(A) is called basic probability number and represents the exact quantity of belief over A.
A set $A \in 2^\Theta$ with $m(A) > 0$ is called a focal element of a belief function over Θ [17].

Definition 2.1.2: (Belief Function)
For a frame of discernment Θ and a set $A \in 2^\Theta$ the belief function $bel : 2^\Theta \to [0, 1]$ represents the total belief over A. The belief function is calculated by the sum of m(B) for all proper subsets B of A:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2.1}$$

From the definition we can easily conclude that the total amount of belief of $\emptyset$ is 0 and the belief for Θ is 1.

Definition 2.1.3: (Plausibility Function)
For a frame of discernment Θ and a set $A \in 2^\Theta$ the plausibility function $pl : 2^\Theta \to [0, 1]$ represents how plausible A is or the amount of belief not committed to the complement of A:

$$pl(A) = bel(\Theta) - bel(\overline{A}) \tag{2.2}$$

Since $bel(\Theta) = 1$:

$$pl(A) = 1 - bel(\overline{A}) \tag{2.3}$$

The amount of belief not commited to Θ is 0 so $pl(\Theta) = 1$ and respectively $pl(\emptyset) = 0$

The plausibility function can also be defined for a set $A \in 2^{\Theta}$ as the sum of the masses intersecting with A, i.e. the intersection of the sets $B \in 2^{\Theta}$ with A that are not $\emptyset$:

$$pl(A) = \sum_{A \cap B \neq \emptyset} m(B) \tag{2.4}$$

It is important to understand the relation between the above definitions. The mass of set $A \in 2^{\Theta}$ is the exact belief that this event occurs, while the belief function expresses the total belief that applies to A. The plausibility function shows how believable A is, i.e. if pl(A) = 0, A is unlikely to happen. Definitions show that $m(A) \leq bel(A) \leq pl(A)$. Furthermore, interval $[bel(A), pl(A)]$ represents the uncertainty of element A, the larger the interval is the more uncertain we are about A.

Definition 2.1.4: (Pignistic probability distribution)
For a frame of discernment Θ and a set $A \in 2^{\Theta}$, the pignistic probability distribution $Bp : 2^{\Theta} \to [0, 1]$ [19], is defined as:

$$Bp(\theta_i) = \sum_{\theta_i \in A \subseteq \Theta} \frac{m(A)}{|A|} \tag{2.5}$$

A pignistic probability distribution [13] is compatible with the mass function m for the same frame of discernment Θ. A probability distribution Pr is said to be compatible with a mass function m when $\forall A \subseteq \Theta$ holds $bel(A) \leq Pr(A) \leq pl(A)$.

2.1.1 Example

In a football game, the possible scenarios for a team are victory, draw, defeat; then the frame of discernment is $\Theta = \{victory, draw, defeat\}$ and the powerset $2^{\Theta} = \{\{\}, \{victory\}, \{draw\}, \{defeat\}, \{victory, draw\}, \{victory, defeat\}, \{draw, defeat\}, \{victory, draw, defeat\}\}$. According to an experienced football analyst with knowledge of mathematics, the following applies:

$m(\{victory\}) = 0.1$

$m(\{draw\}) = 0.3$

$m(\{draw, defeat\}) = 0.2$

$m(\{victory, draw, defeat\}) = 0.4$

The provided information describes the exact amount of belief for each possibility; for instance, the total degree of belief for a draw or defeat is 0.2 and for draw, it is 0.3. These indications are governed by a significant degree of uncertainty and this is obvious from $m(\Theta) = 0.4$.

For every $A \in 2^{\Theta}$ the Dempster-Shafer theory allows us to answer the questions:

1. How likely is event A?
2. How plausible is event A?

For the first question we need to calculate the belief function (Definition 2.1.2):

$bel(\{victory\}) = m(\{victory\}) = 0.1$

$bel(\{draw\}) = m(\{draw\}) = 0.3$

$bel(\{draw, defeat\}) = m(\{draw, defeat\}) + m(\{draw\}) = 0.5$

$bel(\{victory, draw\}) = m(\{victory\}) + m(\{draw\}) = 0.4$

and obviously:

$$\begin{aligned} bel(\{victory, draw, defeat\}) = m(\{victory\}) + m(\{draw\}) + m(\{draw, defeat\}) \\ + m(\{victory, draw, defeat\}) = 1 \end{aligned}$$

For the second question we need to calculate the Plausibility Function (Definition 2.1.3):

$pl(\{victory\}) = m(\{victory\}) + m(\{victory, draw, defeat\}) = 0.5$

$pl(\{draw\}) = m(\{draw\}) + m(\{draw, defeat\}) + m(\{victory, draw, defeat\}) = 0.9$

$pl(\{defeat\}) = m(\{draw, defeat\}) + m(\{victory, draw, defeat\}) = 0,6$

$pl(\{draw, defeat\}) = m(\{draw\}) + m(\{draw, defeat\}) + m(\{victory, draw, defeat\}) = 0.9$

and:

$$\begin{aligned} pl(\{victory, draw, defeat\}) = m(\{victory\}) + m(\{draw\}) + m(\{draw, defeat\}) \\ + m(\{victory, draw, defeat\}) = 1 \end{aligned}$$

From the above calculations we can distinguish the degree of uncertainty for each possibility. Specifically for the event $\{draw\}$, the range $[bel(draw), pl(draw)] = [0.3, 0.9]$ shows that there is enough uncertainty while in the event of a $\{draw, defeat\}$ $[bel(draw, defeat), pl(draw, defeat)] = [0.5, 0.9]$ is more limited.

2.2 Dempster's rule of combination

Dempster's rule of combination (DRC) [6], [17], is a statistical method that allows the combination of evidence from independent sources to reach in one conclusion. Specifically, if different masses exist in the same frame of discernment Θ, DRC allows their combination in order to create a new mass function.

Definition 2.2.1: (Dempster's rule of combination)
Suppose m_1 and m_2 two masses and $A_1, ..., A_k$ and$B_1, ..., B_l$ the corresponding focal elements in the same frame of discernment Θ. We can calculate the combination of the two masses $m_{1,2}$ according to the following formula:

$$m_{1,2}(\emptyset) = 0$$

$$m_{1,2}(A) = \frac{\sum\limits_{A_i \cap B_j = A} m_1(A_i)m_2(B_j)}{1 - \sum\limits_{A_i \cap B_j = \emptyset} m_1(A_i)m_2(B_j)} \quad \text{, for } A \in \Theta, A \neq \emptyset$$

for $K = \sum\limits_{A_i \cap B_j = \emptyset} m_1(A_i)m_2(B_j)$:

$$m_{1,2}(A) = \frac{1}{1-K} \sum_{A_i \cap B_j = A} m_1(A_i)m_2(B_j) \quad \text{, for } A \in \Theta \tag{2.6}$$

The combination of m_1, m_2 is called joint mass and the operation is denoted by $m_{1,2} = m_1 \oplus m_2$

The DRC operation creates a new mass function where the focal elements are a consequence of the focal elements$A_1, ..., A_k$ and$B_1, ..., B_l$ of m_1, m_2 respectively in which there are common elements i.e. $A_i \cap B_j = A$. The constant K is a normalization factor and its purpose is to introduce in equation the all empty intersections of m_1, m_2 focal sets. The smaller K is, the less conflict there is between m_1, m_2 and in the case where $K = 0$ then $m_{1,2}(A) = \frac{1}{1-0} \sum\limits_{A_i \cap B_j = A} m_1(A_i)m_2(B_j) = \sum\limits_{A_i \cap B_j = A} m_1(A_i)m_2(B_j)$

The weight of conflict [17], [15] is defined as:

$$Con(m_1, m_2) = -log(1 - \sum_{B \cap C = \emptyset} m_1(B)m_2(C)) \tag{2.7}$$

2.2.1 Properties of Dempster's rule of combination

DRC theory holds the following properties:

1. Commutative property: the order of operands does not change the result
$$m_1 \oplus m_2 = m_2 \oplus m_1$$
2. Associative property: rearranging the parentheses in an expression of DRC will not change the result
$$(m_1 \oplus m_2) \oplus m_3 = m_1 \oplus (m_2 \oplus m_3)$$
3. Neutral element: when any mass function is combined with $m_0(\Theta) = 1$ it remains unchanged
$$m_1 \oplus m_0 = m_1$$

2.2.2 Example

In the previous example, the football analyst provided us:

$$m(\{victory\}) = 0.1 \quad m(\{draw\}) = 0.3 \quad m(\{draw, defeat\}) = 0.2 \quad m(\Theta) = 0.4$$

Suppose that another football analyst, just as experienced as the previous one, gives the following information for the same match:

$m(\{victory\}) = 0.1$

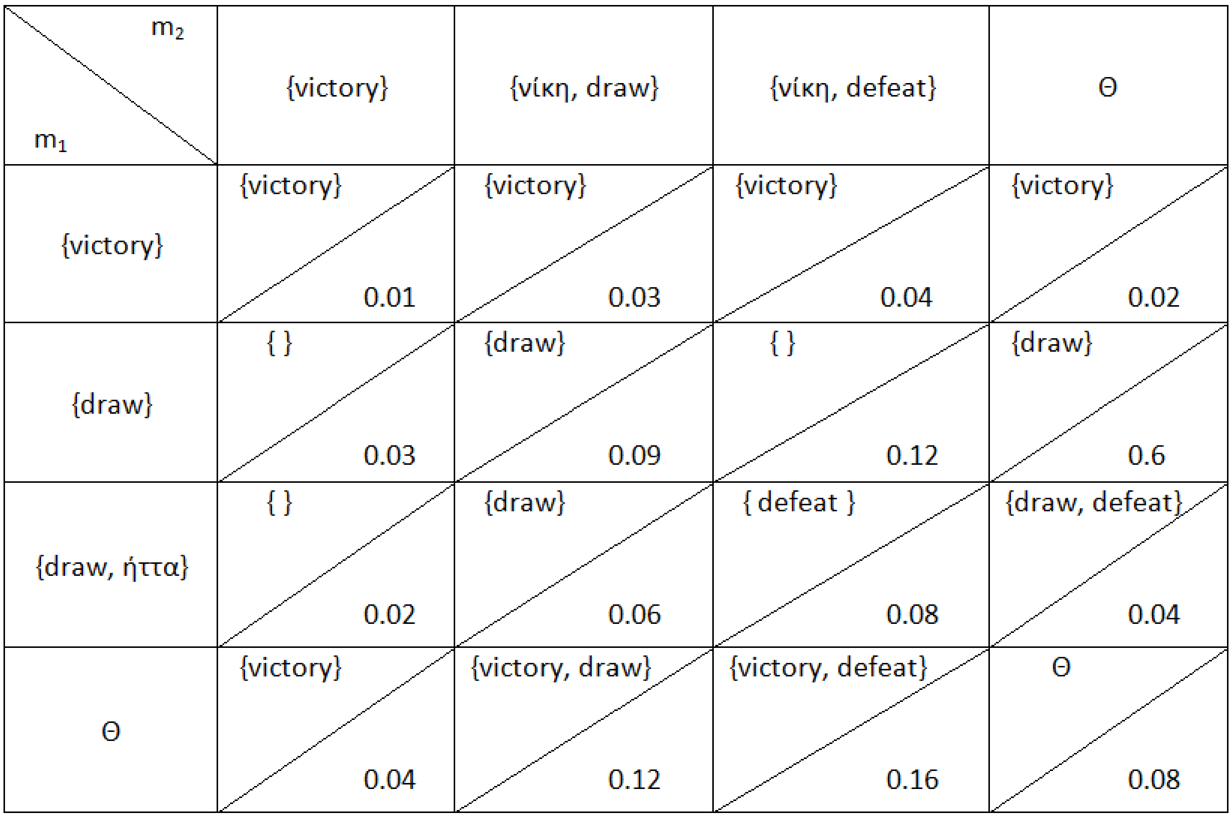
$m(\{victory, draw\}) = 0.3$

$m(\{victory, defeat\}) = 0.4$

$m(\{victory, draw, defeat\}) = 0.2$

We use the DRC to combine the two views and reach a possible result of the match. To calculate $m_{1,2}$ we need to find all the intersections of m_1, m_2 focal sets. The focal sets that are empty are required for the calculation of K and contribute to the conflict, while those that are not empty will be the new focal elements of $m_{1,2}$.

Table 2.1: Focal sets intersaction and their mass values



m_2 / m_1	{victory}	{νίκη, draw}	{νίκη, defeat}	Θ
{victory}	{victory} 0.01	{victory} 0.03	{victory} 0.04	{victory} 0.02
{draw}	{ } 0.03	{draw} 0.09	{ } 0.12	{draw} 0.6
{draw, ήττα}	{ } 0.02	{draw} 0.06	{ defeat } 0.08	{draw, defeat} 0.04
Θ	{victory} 0.04	{victory, draw} 0.12	{victory, defeat} 0.16	Θ 0.08

In each inner cell of table 1 is the intersection of focal elements m_1, m_2, and the multiplication of their masses.

So:

$$K = \sum_{A \cap B} m_1(A)m_2(B) = m_1(\{draw\})m_2(\{victory\}) + m_1(\{draw, defeat\})m_2(\{victory\})$$
$$+ m_1(\{victory, defeat\})m_2(\{draw\}) = 0.03 + 0.02 + 0.12 = 0.17$$

and the weight of conflict:

$$Con(m_1, m_2) = -log(1 - \sum_{B \cap C = \emptyset} m_1(B)m_2(C)) = -log(1 - 0.17) = 0.08$$

we can also calculate:

$$m_{1,2}(\{victory\}) = 1/(1 - 0.17)(0.01 + 0.03 + 0.04 + 0.04 + 0.02) = 0.169$$

$$m_{1,2}(\{draw\}) = 1/(1 - 0.17)(0.09 + 0.06 + 0.06) = 0.253$$

$$m_{1,2}(\{victory, draw\}) = 1/(1 - 0.17)(0.12) = 0.145$$

$$m_{1,2}(\{defeat\}) = 1/(1 - 0.17)(0.08) = 0.096$$

$$m_{1,2}(\{draw, defeat\}) = 1/(1 - 0.17)(0.04) = 0.048$$

$$m_{1,2}(\{victory, defeat\}) = 1/(1 - 0.17)(0.16) = 0.193$$

$$m_{1,2}(\Theta) = 1/(1 - 0.17)(0.08) = 0.096$$

2.3 Dempster rule of combination conflicts

The main criticism of the DRC lies in the counterintuitive results produced in cases where the system is governed by conflict. In particular, Zadeh analyzed some cases where the DRC seems to have a problem [25]. Suppose two meteorologists give the following data for the same day, $m_1(\{sunny\}) = 0.99$, $m_1(\{snow\}) = 0.01$ and $m_2(\{cloudy\}) = 0.99$, $m_2(\{snow\}) = 0.01$. We understand that meteorologists disagree, but it will probably rain or the weather will be cloudy. Although they agree that it is almost impossible to snow, the DRC results in $m_{1,2}(\{snow\}) = 1$. In case that meteorologists did not have a common focal element, then K = 1 and DRC could not be calculated due to zero division. These cases, of course, can be avoided by using a limit on K for high conflict detection.

2.4 Dempster rule of combination complexity

Despite its conflicts, Dempster's rule of combination remains a powerful tool for combining evidence from different sources. By detecting conflicts, DRC could be a reliable asset in statistical analysis problems, however, one reason it is not so widely used lies in its complexity.

Given a frame of discernment Θ, the powerset contains $2^{|\Theta|-1}$ focal points consequently, the DRC must control up to $2^{2|\Theta|}$ intersections. Thus, if the number of mass functions $m_1, \ldots, m_n$, the calculation of $m_{1,\ldots,n} = \oplus_{i=1}^{n} m_i$ in the worst-case scenario the complexity is $O(2^{2|\Theta|})$

According to Pekka Orponen in [12], DRC belongs to the #P-complete complexity class. We say that a function f mapping strings over an alphabet Σ to integers belongs to the #P class if there is a nondeterministic polynomial-time Turing machine M that at each input $x \in \Sigma^*$ has exactly $f(x)$ accepting computational paths. The function f is #P-complete if any other function in #P can be computed by some deterministic polynomial-time Turing machine that is allowed to access values of f at unit cost. The generic #P-complete function is:

> #SAT, the problem of computing number of satisfying truth assignments for a boolean formula in conjuctive normal form.

The class #P can be viewed as the counting version of NP, and for each NP decision problem, there is a corresponding #P counting function that measures the number of accepted solutions of the NP decision problem. Therefore, the calculation time of #P-complete problems is proportional to the NP problems and if $P \neq NP$ holds, the #P-complete problems cannot be solved in polynomial time.

2.5 Approximation algorithms for DRC

In problems where the number of mass functions or focal points is large, DRC computation time may be prohibitive. In order to overcome this difficulty, approximation algorithms have been suggested that reduce the DRC complexity to the detriment of its accuracy. An approximate algorithm tries to approach the actual result while reducing the computation time. There are two main categories of these algorithms: the reduced input algorithms and the Monte Carlo algorithms.

2.5.1 Input-size reducing algorithms

The input-size reducing algorithms as the name implies are efficient algorithms that try to reduce the input size, which in DST theory is the number of focal elements of each mass function. Listed below are the most well-known input size algorithms for DRC.

2.5.1.1 The Bayesian Approximation

The Bayesian approximation [22], reduces a given mass function m by allowing only singleton subsets of the frame of discernment Θ to be focal elements of approximated version $\underline{m}$ of m:

$$\underline{m}(A) = \begin{cases} \dfrac{\sum\limits_{B: A \subseteq B} m(B)}{\sum\limits_{C: C \subseteq \Theta} m(C) \cdot |C|}, & |A| = 1, \\ \\ 0 & otherwise \end{cases} \tag{2.8}$$

In this way, the initial mass function m is converted in a probability distribution with discrete values. The number of values cannot surpass the number of the elements of the frame of discernment $|\Theta|$. Given the mass function m has n focal elements, the computation time of Bayesian approximation is $O(n|\Theta|)$.

This method holds the following properties:

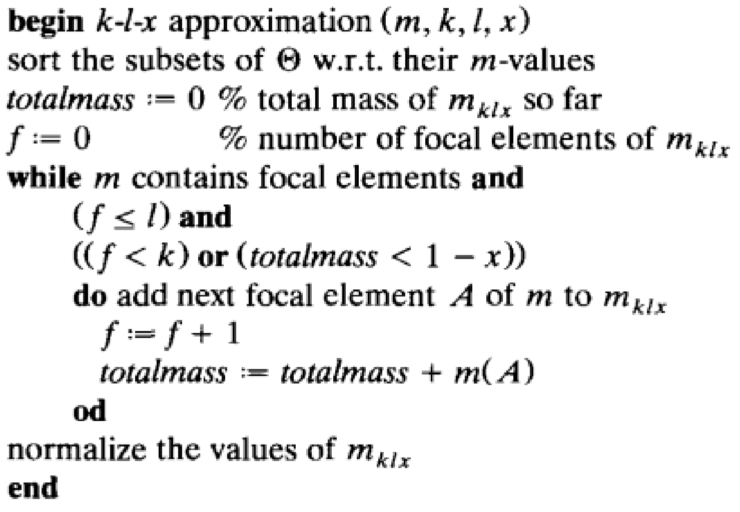
- The order of combination and approximation does not affect the final result, $\underline{m_1 \oplus m_2} = \underline{m_1} \oplus \underline{m_2}$.
- If a mass function has only singletons for focal elements (Bayesian mass function), then $\underline{m_1 \oplus m_2} = m_1 \oplus m_2$.

2.5.1.2 The k-l-x Method

The k-l-x approximation keeps only the highest values of the original mass function m for a discriminant frame Θ. The new mass function m_{klx} contains the focal elements with the highest mass according to the following constraints:

- m_{klx} cannot contain more than l focal elements
- m_{klx} should have at least k focal elements
- the accumulated mass of all remaining focal elements can be at most x, where $x \in [0, 1]$.

The algorithm of k-l-x approximation as presented in [2] by Mathias Bauer is given in Figure 2.1.



```
begin k-l-x approximation $(m, k, l, x)$
sort the subsets of $\Theta$ w.r.t. their $m$-values
$totalmass := 0$ % total mass of $m_{klx}$ so far
$f := 0$             % number of focal elements of $m_{klx}$
while $m$ contains focal elements and
      $(f \leq l)$ and
      $((f < k)$ or $(totalmass < 1 - x))$
      do add next focal element $A$ of $m$ to $m_{klx}$
         $f := f + 1$
         $totalmass := totalmass + m(A)$
      od
normalize the values of $m_{klx}$
end
```

Figure 2.1: k-l-x algorithm

For a mass function m with n focal elements, the complexity of the k-l-x algorithm is $O(n \log n)$

2.5.1.3 Summarization Method

The summarization method (as reviewed in [2]) is similar to the k-l-x approximation with the main difference that the accumulated mass of the remaining focal elements is assigned to the union of the corresponding subsets of Θ.

The new mass function m_s contains k focal elements, the set of k-1 focal elements with the highest mass values in m is denoted by M, and the summarization approximation m_s is defined as:

$$m_S(A) = \begin{cases} m(A), & A \in M, \\ \sum\limits_{A' \subseteq A, A' \notin M} m(A'), & A = A_0 \\ 0 & otherwise \end{cases} \tag{2.9}$$

where

$$A_0 = \bigcup_{\substack{A' \notin M \\ m(A')>0}} A'$$

For a mass function m with n focal elements, the summarization algorithm is computed in time $O(n)$.

2.5.1.4 The D1 Approximation

The D1 approximation [2] follows a similar idea of the k-l-x and the summarization method of maintaining the focal element with higher mass values. In the D1 approximation, M^+ denotes the k-1 elements with the highest mass values for a given mass function m, and M^- the remaining focal elements of m.

$$M^+ = \{A_1, \ldots, A_{k-1} \subseteq \Theta | \, \forall A \notin M^+ : m(A_i) \geq m(A), i = 1, \ldots, k-1\}$$

$$M^- = \{A \subseteq \Theta | \, m(A) > 0, A \notin M^+\}$$

The main difference from the above methods is that the numerical values of the focal elements of M^- are allocated among the elements of M^+. For a focal element $A \in M^-$ of m, m(A) is shared uniformly among the M_A, where M_A is the superset of A in M^+. In case M_A is empty, m(A) is distributed among M'_A where:

$$M'_A = \{B \in M^+ | \, |B| \geq |A|, B \cap A \neq \emptyset\}$$

For a mass function m with n focal elements, the complexity of the D1 Approximation is $O(n)$

2.5.2 Monte-Carlo Algorithms

Monte Carlo methods are computational algorithms that attempt to obtain numerical quantities through repeated random sampling using the Law of Large Numbers and other methods of statistical inference. The idea of Monte Carlo algorithms is to repeat an experiment in order to approach the solution to a deterministic problem.

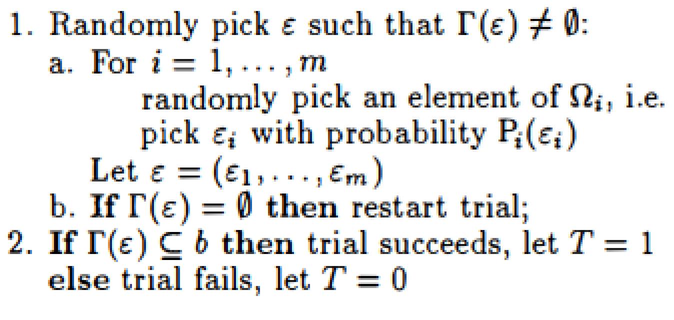
Monte Carlo algorithms are widely used to approximate the result of Dempster's rule of combination for two or more mass functions over a frame of discernment Θ. The Monte Carlo approximation algorithms perform L trials, each trial gives an estimation of belief for a focal point $A \in \Theta$ that is either 0 or 1. The final belief of A is the average result of the L trials. As L increases, the approach converges to the real solution.

Nick Wilson in [24] analyzes Monte Carlo approximation algorithms by Dempster's framework presented in Dempster's lower probabilities induced by a multivalued mapping [6]. A triple (Ω, P, Γ) where Ω is a finite set, P is a probability distribution on Ω and Γ is a function from Ω to 2^Θ where for all $\omega \in \Omega$, $\Gamma(\omega) \neq \emptyset$ and $P(\omega) \neq 0$, defines a source triple over Θ. Any mass and belief function can be expressed by a source triple as $m(A) = \sum\limits_{\omega|\Gamma(\omega)=A} P(\omega)$ and $bel(A) = \sum\limits_{\omega|\Gamma(\omega)\subseteq A} P(\omega)$ respectively. The combination of independent evidence is defined by Dempster's rule as a mapping sending a finite sequence of source triples $(\Omega_i, P_i, \Gamma_i) : i = 1, \ldots, k$, to a triple (Ω, P_{DS}, Γ). Let $\overline{\Omega} = \Omega_1 \times \cdots \times \Omega_k$, for $\omega \in \overline{\Omega}$ the ith component is $\omega(i)$ so that $\omega = (\omega(1), \ldots, \omega(k))$. The function $\Gamma' : \overline{\Omega} \rightarrow 2^\Theta$ and the probability function P' are defined on $\overline{\Omega}$ by $\Gamma'(\omega) = \bigcap\limits_{i=1}^{k} \Gamma_i(\omega(i))$ and $P'(\omega) = \prod\limits_{i=1}^{k} P_i(\omega(i))$ respectively. Let Ω be the set $\{\omega \in \overline{\Omega} : \Gamma'(\omega) \neq \emptyset\}$, Γ be Γ' restricted to Ω and P_{DS} restricted to Ω so $P_{DS} = P'(\omega)/P'(\Omega)$ for $\omega \in \Omega$. The $1/P(\omega)$ represents the measure of the conflict. The combined measure of belief for $A \subseteq \Theta$ is defined by $Bel(A) = P_{DS}(\{\omega \in \Omega : \Gamma(\omega) \subseteq A\})$, also cited as $P_{DS}(\Gamma(\omega) \subseteq A)$.

2.5.2.1 Naive Monte-Carlo algorithm

Naive Monte Carlo algorithm is a simple algorithm that tries to estimate the combined belief for $A \subseteq \Theta$ that defined previously as $Bel(A) = P_{DS}(\Gamma(\omega) \subseteq A)$. The algorithm according to the distribution P_{DS} picks independent ω_l to form a sequence $\omega_1, \ldots, \omega_L$, to perform trials for each ω_l in that sequence. If $\Gamma(\omega) \subseteq A$ then the trial succeeds and the value 1 is assigned else the trial fails with value 0. The $Bel(A)$ is estimated by the proportion of the successful trials.

For a large number of trials the algorithm converges in $Bel(A)$, the steps of the trial are presented in [23] as follows:



```
1. Randomly pick $\varepsilon$ such that $\Gamma(\varepsilon) \neq \emptyset$:
   a. For $i = 1, \ldots, m$
            randomly pick an element of $\Omega_i$, i.e.
            pick $\varepsilon_i$ with probability $P_i(\varepsilon_i)$
      Let $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_m)$
   b. If $\Gamma(\varepsilon) = \emptyset$ then restart trial;
2. If $\Gamma(\varepsilon) \subseteq b$ then trial succeeds, let $T = 1$
   else trial fails, let $T = 0$
```

Figure 2.2: Naive Monte Carlo algorithm

The Naive Monte Carlo algorithm is accurate considering that the measure of conflict remains low.

2.5.2.2 Markov Chain Algorithm

The Markov Chain algorithm follows a similar idea with Naive Monte Carlo algorithms to make a large number of trials that are successful (if $\Gamma(\omega) \subseteq A$) in order to estimate the P_{DS}. The main difference from the Naive Monte Carlo is that the trials are not independent. To avoid picking $\omega \in \overline{\Omega}$ that do not belong to Ω Markov Chain algorithm generates a sequence $\omega^0, \omega^1, \ldots, \omega_L$ where ω^i depends on ω^{i-1}. For the algorithm to be successful and converge to P_{DS}, a condition called ”connectedness” must apply. This condition states that from one element ω_i of the sequence $\omega^0, \omega^1, \ldots, \omega_L$, we can get any element of Ω.

2.6 Software tool for implementing DST

DST software tools help a lot in developing applications that use Dempster-Shafer theory. Using a ready-made tool prevents errors in a new implementation and makes development faster and easier. Some useful functions for a software tool are the computation of the belief and plausibility function given a mass function, and the combination of two or more mass functions with or without approximation algorithms.

In Chapter 4 we implemented our model in the Python programming language and we used the py_dempster_shafer library [14] in order to perform Dempster-Shafer calculations. Specifically, the py_dempster_shafer library supports the following functions (and operators):

MassFunction(data)	Creates mass function, data can be a dictionary or a list of tuples.
MassFunction.bel(self)	Returns the belief of given mass function.
MassFunction.pl(self)	Returns the plausibility of given mass function.
MassFunction.frame(self)	Returns the frame of discernment of the given mass function.
MassFunction & MassFunction	Computes Dempster rule of combination of two mass functions (exact method).
MassFunction.combine_conjunctive(self, sample_count, importance_sampling)	Computes Dempster rule of combination of two mass functions with Monte-Carlo method
MassFunction.conflict(self)	Returns the weight of conflict between two mass functions.
MassFunction.pignistic(self)	Returns pignistic transformation of the given mass function.

Some examples of the library are:

Creating mass function:

```
>> m1 = MassFunction({'ab':0.4, 'b':0.3,  'cd':0.3})
>> print(m1)
{{'a', 'b'}:0.4; {'b'}:0.3; {'c', 'd'}:0.3}
```

Calculation of belief and plausibility functions:

```
>> print('m1.bel({a, b}) =', m1.bel({'a', 'b'}))
m1.bel({a, b}) = 0.7

>> print('m1.pl({a, b}) =', m1.pl({'a', 'b'}))
m1.pl({a, b}) = 0.7
```

Returns the frame of discernment and focal sets:

```
>> print('frame of discernment of m1 =', m1.frame())
frame of discernment of m1 = frozenset({'a', 'c', 'b', 'd'})

>> print('focal sets of m1 =', m1.focal())
focal sets of m1 = {frozenset({'c', 'd'}), frozenset({'a', 'b'}), frozenset({'
    b'})}
```

DRC calculation:

```
>> print(DRC for m1 and m2 =', m1 & m2)
DRC for m1 and m2 = {{'a'}:0.40816326530612246; {'b'}:0.28571428571428575; {'c
    ', 'd'}:0.18367346938775551; {'c'}:0.12244897959183673}

>> print('DRC with Monte-Carlo for m1 and m2 =', m1.combine_conjunctive(m2,
    sample_count = 1000))
DRC with Monte-Carlo for m1 and m2 = {{'a'}:0.4267241379310345; {'b'
    }:0.2844827586206896; {'c', 'd'}:0.18318965517241378; {'c'
    }:0.10560344827586207}
```

Conflict calculation and pignistic transformation:

```
>> print('weight of conflict between m1 and m2 =', m1.conflict(m2))
weight of conflict between m1 and m2 = 0.7133498878774648

>> print('pignistic transformation of m1 =', m1.pignistic())
pignistic transformation of m1 = {{'b'}:0.5; {'a'}:0.2; {'c'}:0.15; {'d'
    }:0.15}
```

3. RECOMMENDATION SYSTEMS

Recommendation Systems (RS) [9] are web applications that aim to offer users products that they probably like. We can consider RS as sellers who aim to offer their customers the products they may want to buy.

3.1 Long Tail Problem

A physical store, such as a bookstore, a video club, or a clothing store, has limited space available for its products, so it selects certain products that considers popular or more likely to sell. In contrast, an online store could have a larger variety of products in order to satisfy more customers with different needs. The distinction between online and offline stores in the number of products they have is called the long tail phenomenon [9] and is shown in Figure 3.1 where the vertical axis represents the popularity of a product and the horizontal axis the products sorted according to their popularity. The products on the left of the vertical bar are the products that could be in a physical store while an online store could host all the products of the line. The huge quantity of products that could be in an online store makes it impossible for a user to see all the products and forces the store to present specific suggestions to each user.

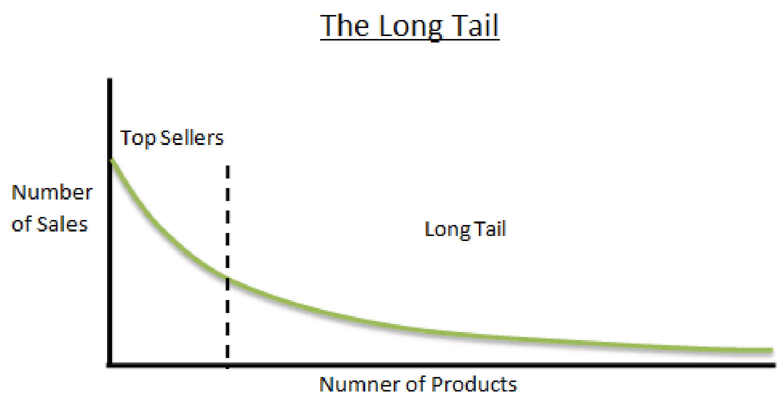


Figure 3.1: Long Tail

3.2 Types of Recommendation Systems

There are two main types of Recommendation Systems [9]:

- Content-based recommenders, the suggestion they make to users relate to the features of products that have shown interest in the past.
- Collaborative-Filtering recommenders, make recommendations to users based on the choices of other users with similar preferences.

Other types of recommenders [21], are Demographic recommenders, which categorize users according to some characteristics such as age, sex assuming that users with common categories have similar tastes, and Hybrid methods that combine some of the aforementioned types. In the following, we will focus on Content-based and Collaborative-Filtering recommenders.

3.2.1 Content-based recommenders

Content-based recommenders emphasize on product features and try to match those features to users' habits and preferences. A profile is created for each product that describes it according to its features, the product will be recommended to the user if the users' preferences fit the products' profile.

To implement a content-based recommendation system, the information and user ratings for each product need to be available, in order to create object profiles and recommend to users categories that may interest them.

For instance, in a movie recommendation system, the profile of each song can consist of the following characteristics: type of music, the singers, the band, the producers, the composer. Each item in an online store would be described by these, so when a user likes a type of music or a singer, a good Content-based recommender system will trace songs that fulfill one or more.

3.2.2 Collaborative-Filtering recommenders

Collaborative-Filtering recommenders focus on the similarities of users' preferences. If some users have common preferences, regarding the products they have used or evaluated, it is reasonable to conclude that they have a common opinion on products that some of them have tried. So for each user, we find a set of similar ones and suggest products that the user has not tried but the others have a good opinion of them.

For a Collaborative-Filtering recommender, we need the users' opinion on the products, which is usually a matrix with the users' ratings on the products. Usually, user ratings are represented by a utility matrix where columns refer to products and rows to users. Table 3.1 shows the ratings for the books Don Quixote, The Great Gatsby, Moby Dick, Pride and Prejudice, Frankenstein, and the ratings of users A, B, C, D from 1 to 5. We should bear in mind that users have not rated most products, so the utility matrix will be empty in many places.
From the utility matrix, we can calculate the distance of two users and find their similarity.

Table 3.1: Utility Matrix

	Don Quixote	Pride and Prejudice	Moby Dick	The Great Gatsby	Frankenstein
A	5		2	3	
B		5			4
C		4		2	3
D	2		5		

The shorter the distance between two users in their ratings, the more similar they are.

Some classic measures are Jaccard Distance and Cosine Distance.

3.2.2.1 Jaccard Distance

Jaccard similarity is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(U_i, U_j) = \frac{|I_i \cap I_j|}{|I_i \cup I_j|} \tag{3.1}$$

The Jaccard distance, measures dissimilarity:

$$D_J(U_i, U_j) = 1 - J(U_i, U_j) \tag{3.2}$$

This method is not very efficient in cases where the utility matrix indicates a preference or not, e.g. ratings, but can be applied when the utility matrix has a quantitative value, such as the sales of a product.

To calculate the Jaccard similarity of users from Table 3, we have to count the number of books that both users have rated and divided by the total number of books that have been rated by the two users:

$J(U_A, U_B) = \frac{0}{5} = 0$, users A, B do not have any common evaluation, depending on the application the result 0 may be the desired result.

$J(U_B, U_C) = \frac{2}{2} = 1$, users have rated the same books

$J(U_A, U_D) = \frac{2}{3}$, although users A, D have a completely different view, Jaccard similarity fails to identify the disagreement.

We can see that Jaccard similarity is not suitable for this example because it can not represent the difference in ratings. If instead of ratings the utility matrix represented the number of times that users had read or bought the book Jaccard similarity would be ideal.

3.2.2.2 Cosine Distance

The cosine similarity is the inner product space between two non-zero vectors. In a RS we can calculate cosiine similarity as:

$$Sc(U_i, U_j) = cos(\theta) = \frac{I_i I_j}{\|I_i\| \, \|I_j\|} = \frac{\sum_{i=1}^{N} I_{i,k} I_{j,k}}{\sqrt{\sum_{i=1}^{n} I_{i,k}^2} \sqrt{\sum_{i=1}^{n} I_{j,k}^2}} \tag{3.3}$$

where U_i, U_j are two users, I_i I_j are the rows of utility matrix and $I_{i,k}$ $I_{j,k}$ are the ratings of the item k of users U_i, U_j respectively .
Cosine distance is defined as:

$$Dc(U_i, U_j) = 1 - Sc(U_i, U_j) \tag{3.4}$$

Because vectors should not have empty cells, we can complete non-existent ratings with zero or average product scores.

In our example for table 3.1 we will consider the non-existent ratings to be zero:

$Sc(U_A, U_B) = \frac{0}{\sqrt{38}\sqrt{41}} = 0$, users A, B have not rated any common book.

$$Sc(U_B, U_C) = \frac{4 \times 5 + 5 \times 2}{\sqrt{5^2 + 4^2}\sqrt{4^2 + 2^2 + 3^2}} = 0.99$$

$$Sc(U_A, U_D) = \frac{2 \times 5 + 5 \times 2}{\sqrt{5^2 + 2^2 + 3^2}\sqrt{2^2 + 5^2}} = 0.6$$

Although the difference in similarity B, C, and A, D is now noticeable, the indication 0.6 in the case of A, D does not reflect disagreement. To express the difference more intensely, we can round or normalize the ratings. One way to round the ratings is to match the positive scores of 3,4,5 to 1 and the negative 1.2 to 0 so:

$$Sc(U_B, U_C) = \frac{1 + 1}{\sqrt{2}\sqrt{3}} = 0.81$$

$$Sc(U_A, U_D) = \frac{0 \times 1 + 1 \times 0}{\sqrt{1}\sqrt{1}} = 0$$

3.2.3 Differences of Content-based and Collaborative-Filtering algorithms

Content-based recommenders relate more to information about objects, while Collaborative-Filtering recommenders base their results on users' ratings and their differences. In general, both algorithms can be applied, but usually Content-based ones are preferred when there is plenty of information about the products or their characteristics determine the users' choices. While Collaborative filtering recommenders are preferred if there is a lack of important information about the product, or the opinion of users has a greater impact.

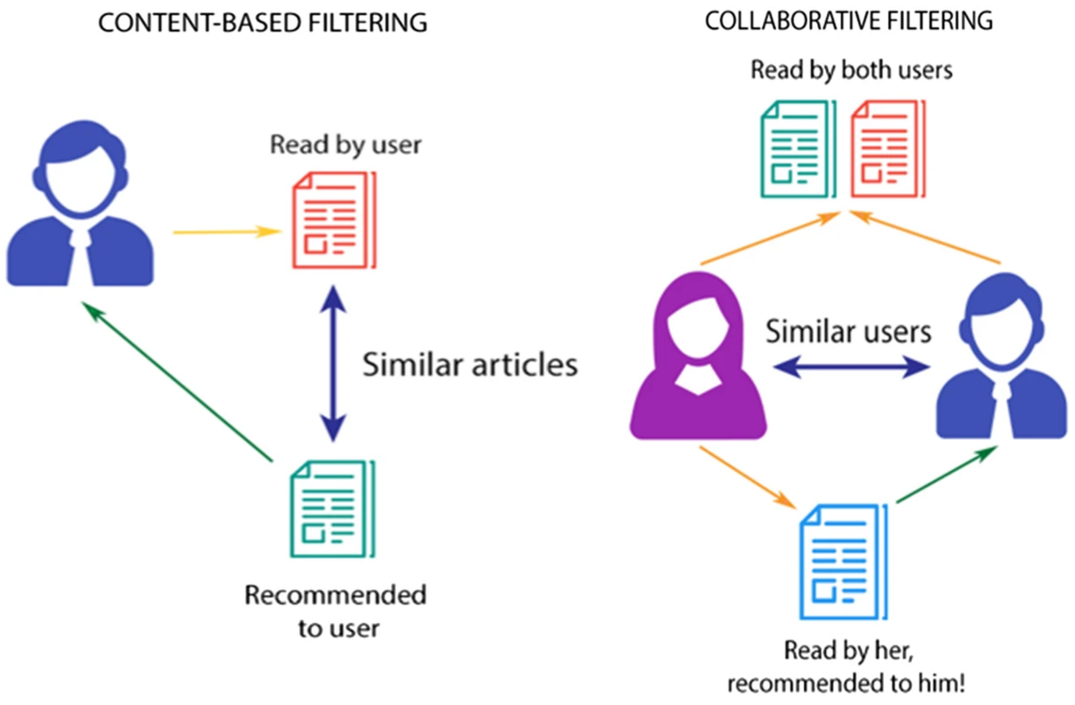


Figure 3.2: Content-based vs Collaborative-Filtering RS

3.3 Performance measures

It is very important to evaluate a recommendation system to improve the potential problems of a method and then compare its effectiveness with different approaches. From beginning to the end of development of an RS, accuracy plays an important role; the most well-known evaluation metrics are Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Precision, Recall, F1-score.

3.3.1 Calculating Error

Let R(u, i), P(u, i) be the actual and predicted scores respectively of the user u for the object i. We can measure the mean absolute error as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|P(u,i) - R(u,i)| \tag{3.5}$$

MAE is a linear score which means that all the individual differences are weighted equally in the average.
Root Mean Square Error (RMSE) measures the average magnitude of the error:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1} N(P(u,i) - R(u,i))^2} \tag{3.6}$$

3.3.2 Normalize Error

It is important in the implementation to take into account the range of possible solutions. In the case where RS predicts a movie rating of 0-5, the deviation of 1 from the actual value for a product is much more significant than the same deviation if the range was 0-10. So to avoid this, we can normalize the error by dividing it by the range of possible scores:

$$NormalizedError = \frac{Error}{\theta_{max} - \theta_{min}} \tag{3.7}$$

3.3.3 Precision and Recall

Usually, in an RS we are not so interested in predicting the precise rating that users will give to a product, but the purpose is to suggest suitable products for them. Error calculation with indicators such as MAE and RMSE helps to implement a good RS that does not make arbitrary predictions, but it may be insufficient to evaluate the final result. Suppose that an RS has ratings 0-5 for each product; we can classify the ratings from 4-5 as positive and the rest as negative. To evaluate an RS [20], we can define:

True Positive (TP): - an interesting item is recommended to the user.

True Negative (TN): an uninteresting item is not recommended to the user.

False Negative (FN): an interesting item is not recommended to the user.

False Positive (FP): an uninteresting item is recommended to the user.

Table 3.2: Results Classifications

	Recommended	Not recommended
Actually good	TP	FN
Actually bad	FP	TN

A good RS ought to suggest interesting products to users, so the measure TP has to be high and the measure FP needs to be low. An indicator of good versus bad recommendations is:

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$

Another indication of the success of an RS is called recall (or sensitivity) and expresses the overall success in distinguishing ”actually good” products:

$$Recall = \frac{TP}{TP + FN} \tag{3.9}$$

A measure that combines precision and recall is called an F1-measure or F1-score:

$$F_1 = 2 \cdot \frac{Precision \cdot recall}{Precision + recall} \tag{3.10}$$

3.4 Uncertainty in Recommendation Systems

The traditional Recommendation systems with the growth of e-commerce have received a great deal of attention and are now very effective. However, a problem with these algorithms lies in the limited number of reviews. Specifically, product reviews using scoring introduce a significant degree of ambiguity and imperfection into the system. In this section, we will study approaches that introduce uncertainty handling through Dempster-Shafer’s theory for dealing with the review’s incompleteness.

3.4.1 Dempster Shafer theory in Content-based approach

Dempster-Shafer theory is a very useful tool for combining different item characteristics in order to make recommendations to users according to their preferences. In [21] authors use DST in order to discover user preferences by estimating the probability interval of an item by its features. The preferences of the users expressed in the past give an insight into each user’s favorite product features.

Let $U = \{U_1, U_2, \ldots, U_n\}$ be the set of the users and $I = \{I_1, I_2, \ldots, I_m\}$ the set of the items. Let $C = \{C_1, \ldots, C_p\}$ denote all the concepts with domain all the features $\Phi_i = \{\phi_{i,1}, \ldots, \phi_{i,k_i}\}$. Each item I_j is defined by a vector $(c_{1,j}, \ldots, c_{p,j})$ where $c_{i,j} \in 2^{\Phi_i}$. In this model is assumed that the number of all features ($\Phi = \bigcup_{i=1}^{p} \Phi_i$) is much smaller than the number of items $\#(\Phi) << \#(I)$. For each $K \in 2^{\Phi_i}$ the mass function is defined as:

$$m_i(K) = \frac{\#U_K}{\#U} \tag{3.11}$$

where $U_K \subseteq U$ beeing the users who selected items with features $c_{i,j} \in K$.

For example, consider the users $U = \{U_1, U_2, U_3, U_4\}$ of a cinema with movies $I = \{I_1, \ldots, I_m\}$. Let the feature concepts be $C = \{Genre, Director, Actors\}$ and let $Genre$ have the domain $\Phi = \{Action, Thriller, Animation, Western\}$. If only one user watches movies with genres $K = \{Action, Western\}$ then $\#U_K = 1$ and $\#U = 4$ so $m(\{Action, Western\}) = 0.25$.

Given a set of items $\Psi = \{I_1, \ldots, I_s\}$ and the set of their common features denoted as $pc_i(\Psi) = K_j \in 2^{\Phi_i}$ the Dempster rule of combination can be applied in order to obtain the mass functions of other item sets as follows:

$$m(\Psi) = \begin{cases} \frac{1}{1-Z} \sum\limits_{A_1 \cap \cdots \cap A_K = \Psi} m_1(pc_1(A_1)) \ldots m_k(pc_k(A_k)) & if A_1 \cap \cdots \cap A_K = \Psi \\ 0 & if A_1 \cap \cdots \cap A_K = \emptyset \end{cases} \tag{3.12}$$

where $Z = \sum\limits_{A_1 \cap \cdots \cap A_K = \emptyset} m_1(pc_1(A_1)) \ldots m_k(pc_k(A_k))$.

Furthermore, the proposed model can calculate the belief and plausibility of a set of items taking into account one or more feature concepts.

3.4.2 Dempster Shafer theory in Collaborative Filtering approach

In [13], [11] the authors propose an automated collaborative filtering model (ACF) that can handle subjective or unreliable ratings and generate hard or soft predictions. By creating soft predictions, their model referred to as CoFiDS (Collaborative Filtering based on Dempster-Shafer belief-theoretic framework) propagates uncertainties from user ratings to the final result.

Given a dataset with the users' hard ratings the algorithm uses DST to express uncertainty as follows. The frame of discernment equals to the set of possible rating $\Theta = \{\theta_1, \theta_2, \ldots, \theta_L\}$ and the hard rating of user U_i of item I_k is transformed into the corresponding soft rating as:

$$m_{i,k}(A) = \begin{cases} \alpha_{ik}(1-\sigma_{ik}), & A = \theta_l \\ \alpha_{ik}\sigma_{ik}, & A = B \\ 1-\alpha_{ik}, & A = \Theta \\ 0, & otherwise \end{cases} \quad \text{with } B = \begin{cases} (\theta_1, \theta_2), & if \quad l = 1 \\ (\theta_{L-1}, \theta_L), & if \quad l = L \\ (\theta_{l-1}, \theta_l, \theta_{l+1}), & otherwise \end{cases} \tag{3.13}$$

where $\alpha_{ik} \in [0,1]$ is the trust factor that reflects the user's true perception and $\sigma_{ik} \in [0,1]$ is the dispersion factor that adjusts the rating span.

In the first step, CoFiDS uses contextual information to predict unrated entries based on the assumption that users that are in the same group have similar preferences. Let $C = \{C_1, \ldots, C_p\}$ denote all the concepts and each concept C_i to have a $Group_i$ (similar to feature domain from chapter 2). The group preference $m_k^{Group_j} : 2^\Theta \to [0,1]$. for an item I_k for a $Group_j$ is defined as:

$$m_k^{(Group_j)} = \bigoplus_{\substack{i:U_i \in Group_j; \\ I_k \in R_i}} m_{ik} \tag{3.14}$$

where R_i are the rating items of user U_i. Respectively the concept preference of m_{ik} is computed by:

$$m_{ik}^{(Concept)} = \bigoplus_{j:Group_j \in f_C(U_i)} m_{ik}^{(Group_j)} \tag{3.15}$$

Given the concept preferences of all items CoFiDS predicts all not existing ratings of user U_i on an item I_k:

$$r_{ik} = m_{ik}^{(Context)} = \bigoplus_{AllConcepts} m_{ik}^{(Concept)} \tag{3.16}$$

CoFiDS uses Chan-Darwiche (CD) distance measure [4] in order to measure the distance of two mass functions and the total dissimilarity of two users:

$$CD(Bp_i, Bp_j) = ln \max_{\theta \in \Theta} \frac{Bp_j(\theta)}{Bp_i(\theta)} - ln \min_{\theta \in \Theta} \frac{Bp_j(\theta)}{Bp_i(\theta)} \tag{3.17}$$

where Bp is the pignistic probability. So the distance between two users U_i, U_j is calculated from the sum of their mass functions $D(U_i, U_j) = \sum_{k=1}^{N} CD(Bp_{ik}, Bp_{jk})$. Moreover instead of calculating the cross product $\theta = \theta_{l_1} \times \cdots \times \theta_{l_N} \equiv \prod_{k=1}^{N} \theta_{l_k} \in \Theta$ of frame of discernment one may use $\theta \in \{\theta_1, \theta_2, \ldots, \theta_L\}$.

Proof provided in [13]:

$$\begin{aligned} & ln \max_{\theta \in \Theta} \frac{Bp_j(\theta)}{Bp_i(\theta)} \\ & = ln \max_{l=\overline{1,L}} \frac{\prod_{k=1}^{N} Bp_j(\theta_{l_k})}{\prod_{k=1}^{N} Bp_i(\theta_{l_k})} = \max_{l=\overline{1,L}} ln \prod_{k=1}^{N} \frac{Bp_j(\theta_{l_k})}{Bp_i(\theta_{l_k})} \\ & = \max_{l=\overline{1,L}} \sum_{k=1}^{N} ln \frac{Bp_j(\theta_{l_k})}{Bp_i(\theta_{l_k})} = \sum_{k=1}^{N} ln \max_{l=\overline{1,L}} \frac{Bp_j(\theta_{l_k})}{Bp_i(\theta_{l_k})} \end{aligned}$$

Supposing that user U_i has not rated item I_k CoFiDS estimates the predicted rating r_{ik} with DRC of the neighbor's ratings. The neighborhood for a user U_i is formed by the K nearest neighbor (KNN) and the minimum similarity thresholding (MST):

$$Nbhd_{ik} = \{U_j \in U : I_k \in R_j, s_{ij} \geq \max_{\forall U_l \notin Nbhd_{ik}} \{t, s_{il}\}\} \tag{3.18}$$

where s_{ij} is the similarity of users U_i, U_j and the size of the neighborhood does not exceed K (ie $|Nbhd_{ik}| \leq K$).

3.4.3 Data classification using the Dempster-Shafer theory

Classifiers are types of algorithms in the field of Statistics and Artificial Intelligence that try to identify complex data into several finite classes. Dempster's rule of combination is considered a classification methodology that combines different sets of evidence from independent sources. Recommendation systems are statistical classification problems in the sense that they attempt to classify items as ”interesting” or ”not interesting” for the users. Usually, DRC is used to combine results from different classifiers, however, in [5] authors propose three models that DRC is an independent classification system.

The first model in [5] is proposed for the WBCD standard benchmark dataset of the UCI Machine Learning Repository [10], which classifies items as malignant or benign. Each item contains nine integer attributes in the range of 1 to 10, with the assumption that lower values represent normal data the mass functions are modeled using sigmoid functions as follows:

$$m(normal) = (1 + e^{(v-t)})^{-1}$$
$$m(abnormal) = 1 - m(normal)$$

where the frame of discernment is $\Theta = \{normal, abnormal\}$, t is the median threshold of each attribute obtained from the training set and v is the value for the test data for that attribute. Using DRC one may combine the different attributes to reach one conclusion.

The second experiment [5] is the Iris Plant dataset [10] in which the frame of discernment is $\Theta = \{Setosa, Versicolour, Virginica\}$, with every data item containing four numeric attributes. From the training set is retrieved the maximum and the minimum values for each attribute. From these values, each attribute of the validation set is assigned the classes in Θ that the value is between the corresponding minimum and maximum and the mass functions are denoted as:

$$m(C) = 0.9, m(\Theta) = 0.1$$

if it is assigned in one class then $C = \{class\ x\}$ else if there are two possible classes then $C = \{class\ x \cup class\ y\}$ else if it belongs to all three possible classes $m(\Theta) = 1$.

The third experiment [5] is for the Duke Outage Dataset (DOD) with the frame of discernment $\Theta = \{tree, animal, lightning, others\}$ and each data item containing six non-numerical attributes. From the training set, the likelihood measure of fault i given a event j is defined as:

$$L_{ij} = \frac{N_{ij}}{N_j}$$

where N_{ij} is the number of outages caused by fault i under event j and N_j the total number of outages under event j. Given the likelihood measure L_{ij}, the mass function of each attribute is computed as:

$$\begin{aligned} & m(\Theta) = 1, if L_{ij} = L_I \\ & m(i) = \frac{(L_{ij} - L_i)}{1 - L_i}, m(\Theta) = 1 - m(i), && if \quad L_{ij} > L_i \\ & m(\neg i) = \frac{(L_{ij} - L_i)}{L_i}, m(\Theta) = 1 - m(\neg i), && if \quad L_{ij} < L_i \end{aligned}$$

Given the above mass function for each attribute, the model can combine them using DRC.

Overall, all three experiments performed well and achieved high classification accuracy. We can therefore conclude that DS theory provides the framework that is effective if it is carefully designed with the right attributes.

4. A RECOMMENDATION SYSTEM BASED ON DEMPSTER SHAFER THEORY

In this chapter we present the movie recommendation system that we implemented, using the Dempster-Shafer theory with a collaborative filtering approach. The purpose of our model is to design a simple collaborative filtering algorithm using DST in order to handle imperfections or lack of information in user ratings and express them in the final results. This purpose is achieved by converting the real ratings in mass functions that cover a range of scores and using DRC in order to combine similar users' views to reach a prediction that is also expressed in a mass function. This work presents a model that takes user ratings, converts them to mass functions and pignistic probabilities, calculates user-user distances to form neighborhoods and finally predicts ratings for unrated movies (Figure 4.1). We also make a reference to the decision-making process.

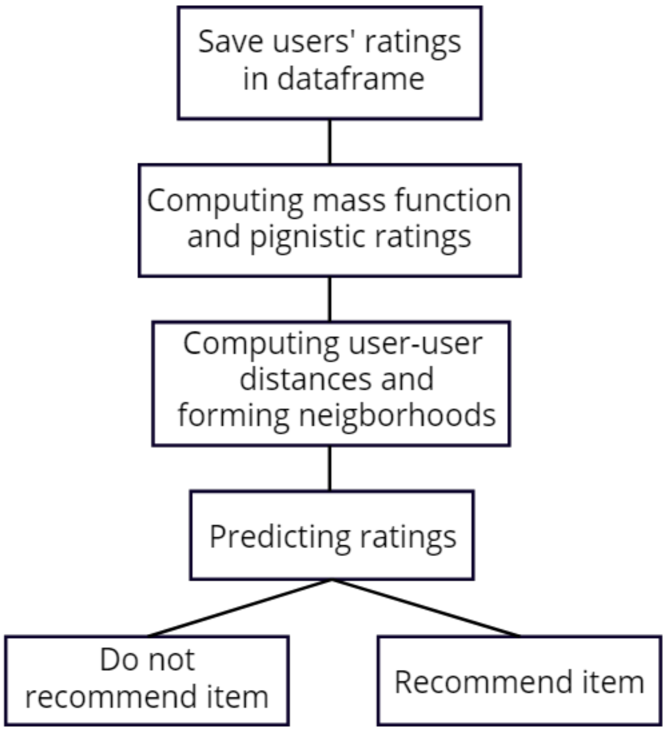

Save users' ratings in dataframe
Computing mass function and pignistic ratings
Computing user-user distances and forming neigborhoods
Predicting ratings
Do not recommend item
Recommend item

Figure 4.1: General steps of the model

The overall architecture of the implementation is presented in Figure 4.2.

4.1 The model

Our model is based on [13], [11], which are presented in subsection 3.4.2, with the difference that we did not follow the proposed approach to use contextual information to rate unrated data before computing user's similarity and generating predictions. We implemented a collaborative filtering RS which consists of three main steps:

a) computing users similarity,

b) forming neighborhoods,

c) make predictions based on similar users.

At first, we convert user's ratings into mass functions. Given the mass functions, we calculate the similarity of each user pair and for each of them, we form a neighborhood with the most similar users. Finally, we predict unrated movies for each user according to their neighbors. In the next sections, we analyze these steps in our proposed method.

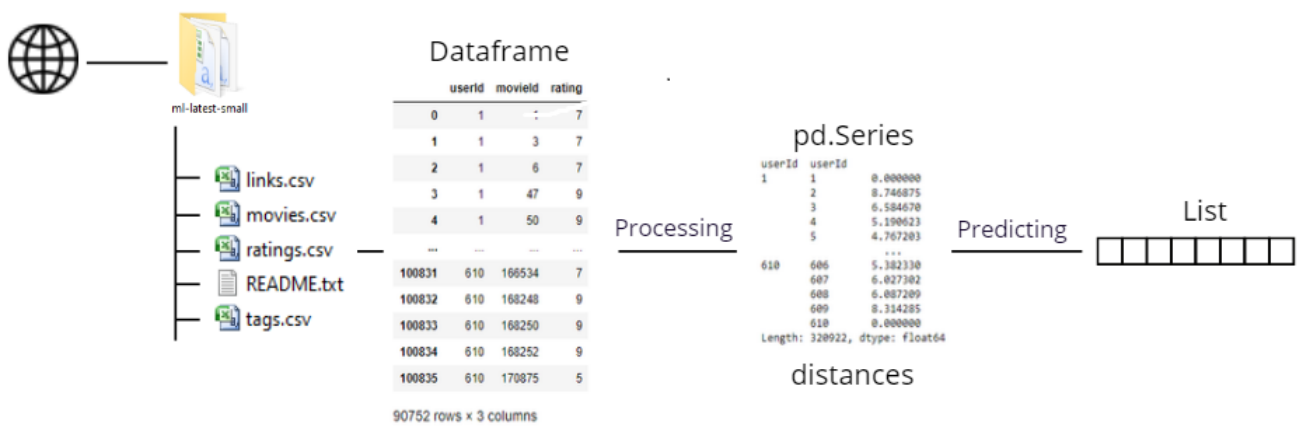
4.1.1 Input Data

For the implementation, we used two versions of the MovieLens dataset [1], [7]: the Small version with 100,000 ratings to 9,000 movies by 600 users and the Full version of 27,000,000 ratings to 58,000 movies by 280,000 users (due to space complexity, only 1,000,000 ratings were used). We used the ratings.csv file (Figure 4.2) that contains hard ratings of the users in the set {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}; for the needs of the implementation, ratings were scaled from 0 to 9, ie $new\_rating = 2 * rating - 1$. Then we converted each hard rating to soft ratings accordingly with the Dempster-Shafer modeling function:

$$m_{i,k}(A) = \begin{cases} 0.7, & A = \theta_l \\ 0.2, & A = B \\ 0.1, & A = \Theta \\ 0, & otherwise \end{cases} \quad \text{with } B = \begin{cases} (\theta_1, \theta_2), & if \quad l = 1 \\ (\theta_{L-1}, \theta_L), & if \quad l = L \\ (\theta_{l-1}, \theta_l, \theta_{l+1}), & otherwise \end{cases} \tag{4.1}$$

where in our implementation the frame of discernment is $\Theta = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $L = 10$.

Suppose a movie k by user i is rated 7 out of 9 then $m_{i,k}(\{7\}) = 0.7$, $m_{i,k}(\{6, 7, 8\}) = 0.2$ and $m_{i,k}(\Theta) = 0.1$ also denoted as $m_{i,k} = \{\{7\} : 0.7, \{6, 7, 8\} : 0.2, \Theta : 0.1\}$. Respectively if the movie was rated 9 out of 9 then $m_{i,k} = \{\{9\} : 0.7, \{8.9\} : 0.2, \Theta : 0.1\}$.



	userId	movieId	rating
0	1	:	7
1	1	3	7
2	1	6	7
3	1	47	9
4	1	50	9
...	...	...	...
100831	610	166534	7
100832	610	168248	9
100833	610	168250	9
100834	610	168252	9
100835	610	170875	5

90752 rows × 3 columns

```
userId  userId
1       1         0.000000
        2         8.746875
        3         6.584670
        4         5.190623
        5         4.767203
                    ...
610     606       5.382330
        607       6.027302
        608       6.087209
        609       8.314285
        610       0.000000
Length: 320922, dtype: float64
```

Figure 4.2: Input

4.1.2 Computing users similarity

We measure users dissimilarity with the distance first introduced in [4] and further developed by [13]. We define the distance between two users as:

$$D(U_i, U_j) = \sum_{k=1}^{N} \left(\ln \max_{\theta \in \Theta} \frac{Bp_{j,k}(\theta)}{Bp_{i,k}(\theta)} - \ln \min_{\theta \in \Theta} \frac{Bp_{j,k}(\theta)}{Bp_{i,k}(\theta)} \right) \tag{4.2}$$

where $Bp_{i,k}$ is the pignistic probability of user U_i in item k.

In our model, we ignore the movies that have not been rated either from U_i or U_j, so $N = |I_i \cap I_j|$ and because it can be different from each pair of users, we calculate the average distance of each movie:

$$D(U_i, U_j) = \frac{1}{N} \sum_{k=1}^{N} \left(\ln \max_{\theta \in \Theta} \frac{Bp_{j,k}(\theta)}{Bp_{i,k}(\theta)} - \ln \min_{\theta \in \Theta} \frac{Bp_{j,k}(\theta)}{Bp_{i,k}(\theta)} \right) \tag{4.3}$$

4.1.3 Forming Neighborhoods

In our model for selecting neighbors, we used the algorithm K-nearest neighbor (KNN), which selects for a user u_i the K users who are more similar with u_i. Since we do not calculate the similarity, we select the users who have the shortest distance from the user u_i.

4.1.4 Predicting ratings

To predict whether a user will like an item or not, we look for ratings from neighbors who have rated the item.
For an item $I_k \in I$ and a user $U_i \in U$, the estimated rating in form mass function $m_{i,k}$ is calculated from DRC of all $U_j \in Nbhd_i$ neighbors that have rated the product I_k:

$$m_{i,k} = \bigoplus_{\substack{j:U_j \in Nbhd_i; \\ I_k \in R_j}} m_{j,k} \tag{4.4}$$

The maximum pignistic probability can be selected as the predicted hard rating.

Neighbors who have not rated the product I_k are not included in the calculation of DRC, so in case no neighbor has rated the product, we can not predict with the above method the rating of user U_i.

4.1.5 Making Recommendations

As stated by [13], [11] there are two main strategies for producing recommendations.

In the first strategy, the choices are made according to the predicted hard rating, which is the singleton with the maximum pignistic ratings.

The second strategy, the maximum belief with overlapping interval strategy (maxBL) [3], makes soft decisions. The soft ratings are the singletons, which their belief are greater than all the plausibility of all others singletons. In case there is no such singleton, the soft ratings with the highest belief, plausibility function are selected.

With either of these strategies, I believe that the extra information provided by belief and plausibility functions can play a crucial role in the final stage of an RS. This is because they are two indicators that guide us on how sure we can be for our choices.

4.1.6 Calculate Error

In Chapter 3, we referred to the measurements of MAE and RMSE, which we can use to calculate the error using the predicted hard rating $r_{i,k}$. However, it is useful to evaluate the overall information expressed in the prediction of the mass function $m_{i,k}$. For this reason, we chose the evaluation method DS-MAE [13] which is defined as:

$$DS\text{-}MAE(\theta_j) = \frac{1}{|D_j|} \sum_{\substack{(i,k)\in D_j;\\ \theta_l \in \Theta}} Bp_{i,k}(\theta_l) \cdot |r_{i,k} - \theta_l| \tag{4.5}$$

where D_j is the testing set and $r_{i,k} \in \Theta$ is the actual rating.

We can also include the result of the mass function to calculate precision and recall:

$$DS\text{-}Precision(\theta_j) = \frac{TP(\theta_j)}{TP(\theta_j) + FP(\theta_j)} \tag{4.6}$$

$$DS\text{-}Recall(\theta_j) = \frac{TP(\theta_j)}{TP(\theta_j) + FN(\theta_j)} \tag{4.7}$$

where:

$$TP(\theta_j) = \sum_{(i,k)\in D_j} Bp_{i,k}(\theta_j) \tag{4.8}$$

$$FP(\theta_j) = \sum_{\substack{(i,k)\in D_l\\ l\neq l}} Bp_{i,k}(\theta_j) \tag{4.9}$$

$$FN(\theta_j) = \sum_{\substack{(i,k)\in D_j\\ l\neq l}} Bp_{i,k}(\theta_l) \tag{4.10}$$

So:

$$DS\text{-}F_1(\theta_j) = 2 \cdot \frac{DS\text{-}Precision(\theta_j) \cdot DS\text{-}Recall(\theta_j)}{DS\text{-}Precision(\theta_j) + DS\text{-}Recall(\theta_j)} \tag{4.11}$$

4.1.7 An example of application

Let’s look at the example for the following data:

For this example we will not scale the ratings.

Table 4.1: example, ratings

Users	Items	Ratings
U1	I1	5
U1	I2	4
U1	I4	2
U2	I1	4
U2	I4	3
U3	I2	3
U3	I3	2
U3	I4	4

Table 4.2: example, computing soft ratings

Users	Items	Ratings	Mass function ratings
U1	I1	5	$\{\{5\}:0.7,\{4,5\}:0.2,\Theta:0.1\}$
U1	I2	4	$\{\{4\}:0.7,\{3,4,5\}:0.2,\Theta:0.1\}$
U1	I4	2	$\{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$
U2	I1	4	$\{\{4\}:0.7,\{3,4,5\}:0.2,\Theta:0.1\}$
U2	I4	3	$\{\{3\}:0.7,\{2,3,4\}:0.2,\Theta:0.1\}$
U3	I2	3	$\{\{3\}:0.7,\{2,3,4\}:0.2,\Theta:0.1\}$
U3	I3	2	$\{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$
U3	I4	4	$\{\{4\}:0.7,\{3,4,5\}:0.2,\Theta:0.1\}$

For a start we convert the hard ratings into soft ratings and create from the soft ratings mass function, Table 4.2
For each rating we calculate the pignistic probabilities of the mass function and we calculate the distances between the users, Table 4.3

Table 4.3: example, users distances

Users	Users	Distances
U1	U2	4.2
U1	U3	5.8
U2	U3	4.4

We choose for each user neighbors who have the shortest distance.
For this example, we assume that all users are neighbors, so we can make the following predictions:

$$U1, I3 : m_{1,3} = m_{3,3} = \{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$$

$$U2, I1 : m_{2,1} = m_{1,1} = \{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$$

$$U2, I2 : m_{2,2} = m_{1,2} \oplus m_{3,2} = \{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$$

$$U2, I3 : m_{2,3} = m_{3,3} = \{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$$

$$U3, I1 : m_{3,1} = m_{1,1} \oplus m_{2,1} = \{\{2\}:0.7,\{1,2,3\}:0.2,\Theta:0.1\}$$

We can recommend users items that $bel(\{4,5\})$ and $pl(\{4,5\})$ is high.

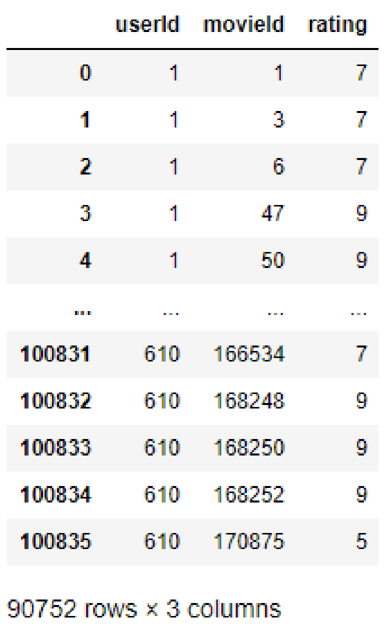
Table 4.4: exmaple, DRC results

Users	Items	$m_{1.2}$	bel({4,5})	pl({4,5})
U1	I3	$\{\{2\}: 0.7, \{1,2,3\}: 0.2, \Theta: 0.1\}$	0.0	0.1
U2	I1	$\{\{5\}: 0.7, \{4,5\}: 0.2, \Theta: 0.1\}$	0.9	1.0
U2	I2	$\{\{4\}: 0.41, \{3\}: 0.41, \{3,4\}: 0.08, \{3,4,5\}: 0.04, \Theta: 0.02\}$	0.41	0.59
U2	I3	$\{\{2\}: 0.7, \{1,2,3\}: 0.2, \Theta: 0.1\}$	0.0	0.1
U3	I1	$\{\{4\}: 0.91, \{3,4,5\}: 0.08, \Theta: 0.01\}$	0.91	1.0

4.2 Implementation

We implement the model in Python 3.6 programming language, for the Dempster-Shafer theory. We rely on the py_dempster_shafer library by Thomas Reineking [14], and we use Pandas library to store and process the data. In this thesis, we developed a method for calculating users distances, which takes advantage of the effectiveness of the Pandas library. Pandas library is based on the Numpy library, which is very efficient in array operations. Moving on, we will analyze step by step the implementation of the model.

Table 4.5: implementation, load ratings



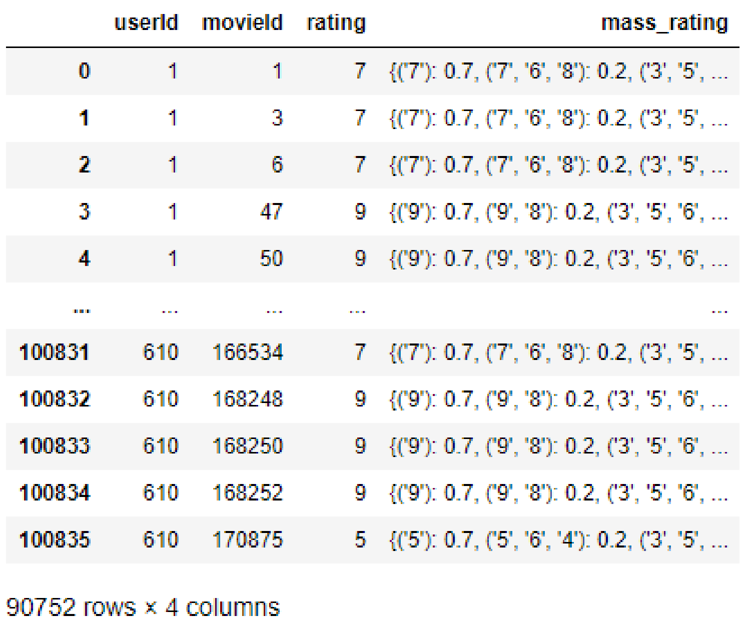
	userId	movieId	rating
0	1	1	7
1	1	3	7
2	1	6	7
3	1	47	9
4	1	50	9
...	...	...	...
100831	610	166534	7
100832	610	168248	9
100833	610	168250	9
100834	610	168252	9
100835	610	170875	5

90752 rows × 3 columns

Initially, the data are loaded in a dataframe, ratings are scaled from 0 to 9 and the timestamp column is dropped, since we do not use it (Table 4.5). With the Pandas method $apply$, we calculate and store in a different column the soft ratings (as a mass function) from the corresponding hard ratings (Table 4.6); respectively from the mass functions, we calculate the pignistic probabilities for each rating (Table 4.7).

To calculate the distances of the users, we create a new dataframe where for columns are the elements of $\Theta = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and it is filled with the pignistic values of the ratings (Table 4.8). We take the rows of the pignistic ratings of the products that each user has rated from the new dataframe and apply a function that calculates the distances of

Table 4.6: implementation,mass ratings

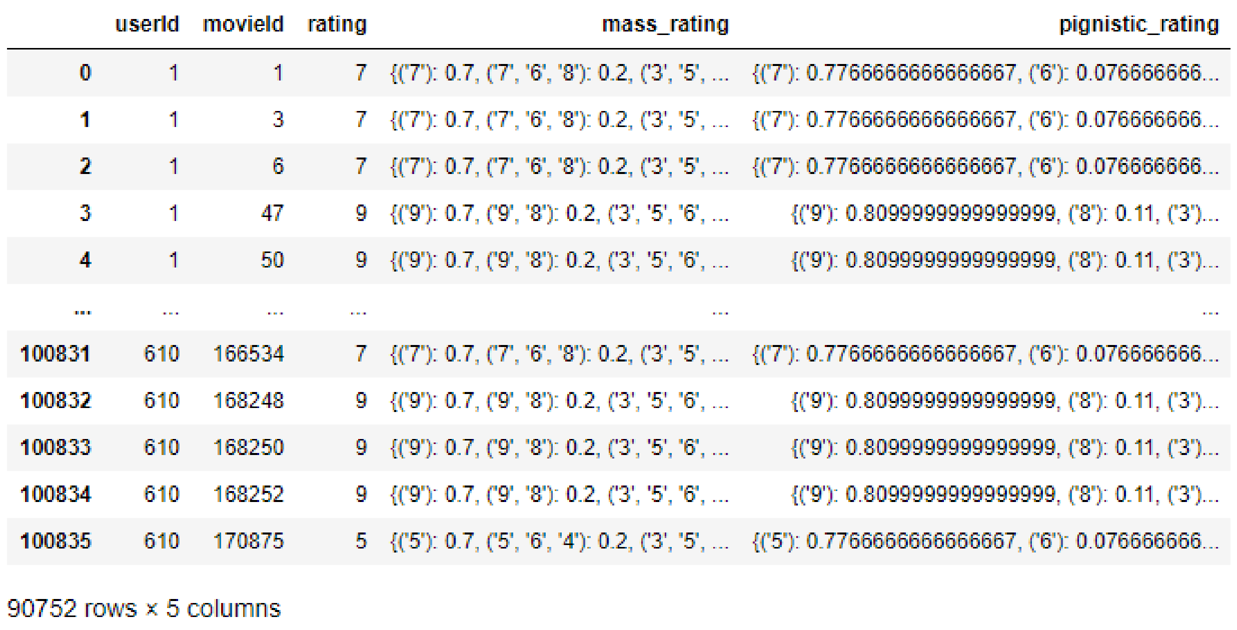


	userId	movieId	rating	mass_rating
0	1	1	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...
1	1	3	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...
2	1	6	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...
3	1	47	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...
4	1	50	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...
...	...	...	...	...
100831	610	166534	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...
100832	610	168248	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...
100833	610	168250	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...
100834	610	168252	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...
100835	610	170875	5	{('5'): 0.7, ('5', '6', '4'): 0.2, ('3', '5', ...

90752 rows × 4 columns

the user from the rest. The function divides the user part with the rest of the dataframe for the movies they have in common, in this way, we have all the divisions need to calculate $\frac{Bp_{j,k}(\theta)}{Bp_{i,k}(\theta)}$ for the distance type 4.3. So the sum of the difference of the logarithms of the minimum and the maximum of the result is calculated. The final result of this process is a pd.Series with the distances of each user (Table 4.9).

Table 4.7: implementation, pignistic ratings

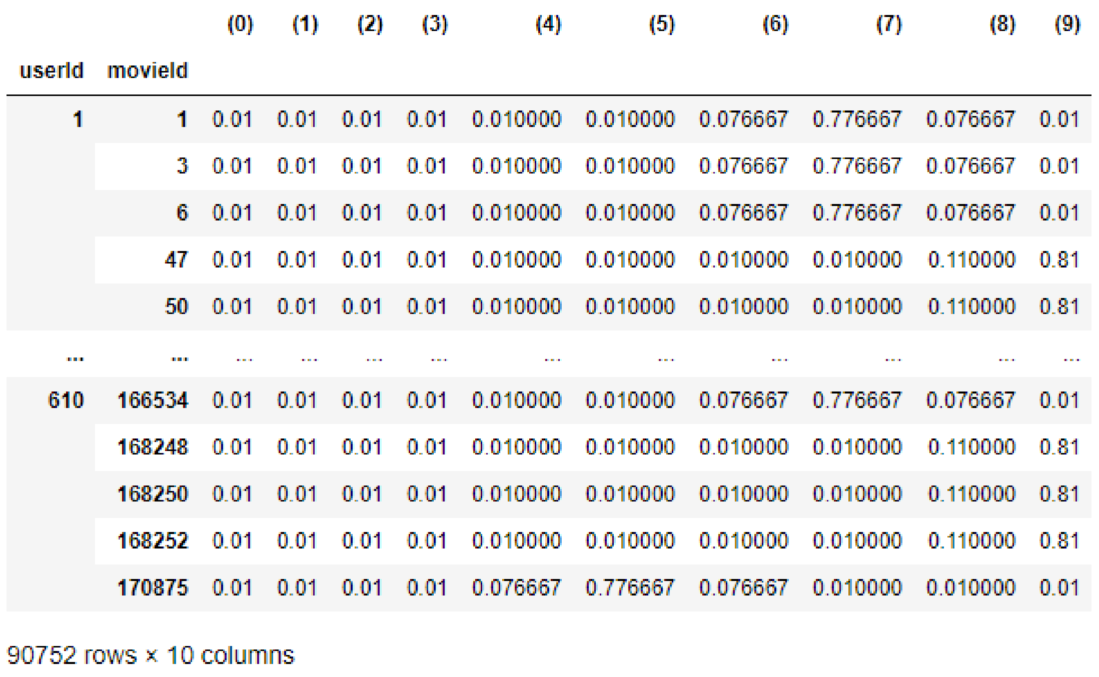


	userId	movieId	rating	mass_rating	pignistic_rating
0	1	1	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...	{('7'): 0.7766666666666667, ('6'): 0.076666666...
1	1	3	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...	{('7'): 0.7766666666666667, ('6'): 0.076666666...
2	1	6	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...	{('7'): 0.7766666666666667, ('6'): 0.076666666...
3	1	47	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...	{('9'): 0.8099999999999999, ('8'): 0.11, ('3')...
4	1	50	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...	{('9'): 0.8099999999999999, ('8'): 0.11, ('3')...
...	...	...	...	...	...
100831	610	166534	7	{('7'): 0.7, ('7', '6', '8'): 0.2, ('3', '5', ...	{('7'): 0.7766666666666667, ('6'): 0.076666666...
100832	610	168248	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...	{('9'): 0.8099999999999999, ('8'): 0.11, ('3')...
100833	610	168250	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...	{('9'): 0.8099999999999999, ('8'): 0.11, ('3')...
100834	610	168252	9	{('9'): 0.7, ('9', '8'): 0.2, ('3', '5', '6', ...	{('9'): 0.8099999999999999, ('8'): 0.11, ('3')...
100835	610	170875	5	{('5'): 0.7, ('5', '6', '4'): 0.2, ('3', '5', ...	{('5'): 0.7766666666666667, ('6'): 0.076666666...

90752 rows × 5 columns

The advantage of this method is that the time is drastically reduced, namely in an initial implementation that was serially looking for the cost for each user finding the common movies with the other users for the file with 100,000 ratings the calculation took 240 seconds while

Table 4.8: implementation, expanded dataframe with pignistic values

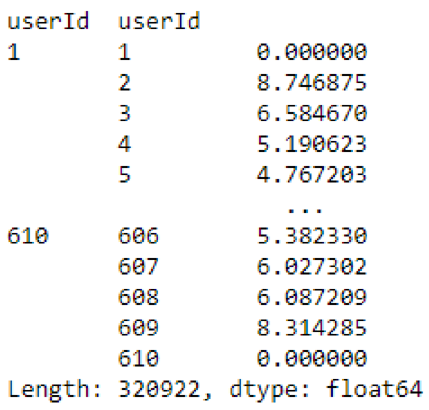


userId	movieId	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
1	1	0.01	0.01	0.01	0.01	0.010000	0.010000	0.076667	0.776667	0.076667	0.01
	3	0.01	0.01	0.01	0.01	0.010000	0.010000	0.076667	0.776667	0.076667	0.01
	6	0.01	0.01	0.01	0.01	0.010000	0.010000	0.076667	0.776667	0.076667	0.01
	47	0.01	0.01	0.01	0.01	0.010000	0.010000	0.010000	0.010000	0.110000	0.81
	50	0.01	0.01	0.01	0.01	0.010000	0.010000	0.010000	0.010000	0.110000	0.81
...	...	...	...	...	...	...	...	...	...	...	...
610	166534	0.01	0.01	0.01	0.01	0.010000	0.010000	0.076667	0.776667	0.076667	0.01
	168248	0.01	0.01	0.01	0.01	0.010000	0.010000	0.010000	0.010000	0.110000	0.81
	168250	0.01	0.01	0.01	0.01	0.010000	0.010000	0.010000	0.010000	0.110000	0.81
	168252	0.01	0.01	0.01	0.01	0.010000	0.010000	0.010000	0.010000	0.110000	0.81
	170875	0.01	0.01	0.01	0.01	0.076667	0.776667	0.076667	0.010000	0.010000	0.01

90752 rows × 10 columns

with the implementation of dataframe in just 0.2 seconds. The huge difference is mainly due to the vectorization of the problem and the Pandas functions for dividing arrays into parameters.

Table 4.9: implementation, users distances



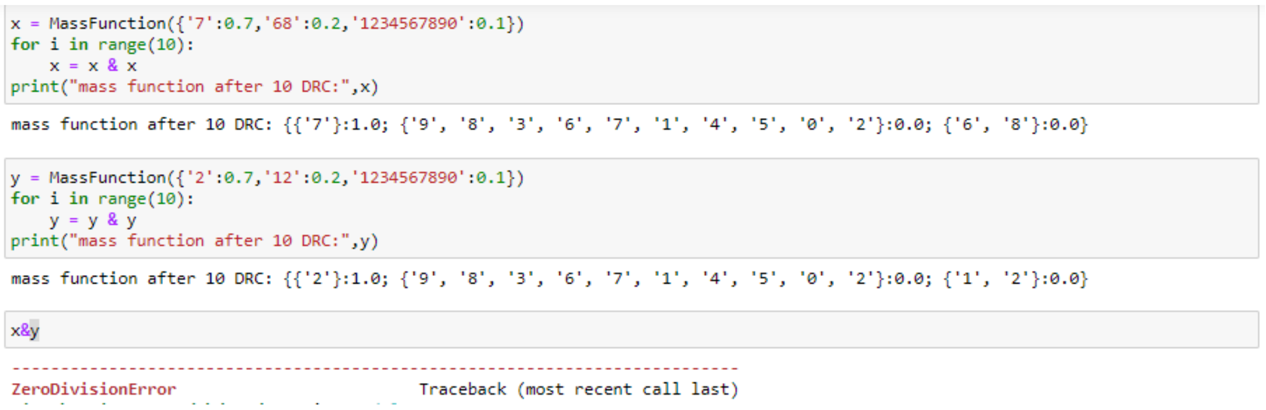
```
userId  userId
1       1          0.000000
        2          8.746875
        3          6.584670
        4          5.190623
        5          4.767203
                     ...
610     606        5.382330
        607        6.027302
        608        6.087209
        609        8.314285
        610        0.000000
Length: 320922, dtype: float64
```

From the Pd.series (Table 4.9), which is the distances for each user pair, we select for each user the K nearest neighbors and we use the exact DRC method to predict the rating of a movie that the user has not rated. We exclude users who have zero distance, because either they have seen the same movies or they have not seen any common movie and therefore we can not judge whether they are similar or not.

4.2.1 Problems in Implementation

The main problem we encountered in the implementation was its space complexity in the large database of 27,000,000 ratings. Only the rating.csv file we had to load is 724 MB and to calculate the mass function and the pignistic ratings the necessary space is increased extremely. To avoid the necessity of more space, we limited the implementation to 1,000,000 ratings.

Another problem also mentioned in [16] lies in the underflow in some DRC calculations. The problem occurs when mass functions with low conflict and a high degree of belief in the same focal element are combined. In such a case, DRC returns very small value at Θ, which due to underflow is zeroed. This can lead to mass functions that do not have a common focal element. The library pyds that implements the DRC, if there is no common focal element the exact DRC method (&) returns an error (Figure 4.3), while the Monte-Carlo algorithm returns the empty set ({}). We encountered this problem when we tried to implement the proposed method in [13], [11] in the first step, predicting unrated data. Our final model is not affected by such significant underflows since all DRCs have a non-zero value in Θ for at least one of the two mass functions. Let's consider an example where twenty users rate a movie k from 0 to 9. So the frame of discernment is $\Theta = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let half of the users h_1 have rated the movie with 7 and the other half h_2 with 2. The mass functions for each rating is $r_{i,k} = \{\{7\} : 0.7, \{6, 7, 8\} : 0.2, \Theta : 0.1\}$ for $i \in h_1$ and $r_{i,k} = \{\{2\} : 0.7, \{1, 2, 3\} : 0.2, \Theta : 0.1\}$ for $i \in h_2$. Due to associative property, we can compute the DRC as follows: $r_k = (\underset{i \in h_1}{\oplus} r_i) \oplus (\underset{i \in h_2}{\oplus} r_i)$. But, because of the underflow $\underset{i \in h_1}{\oplus} r_i = \{\{7\} : 1, \{6, 7, 8\} : 0, \Theta : 0\}$ and $\underset{i \in h_2}{\oplus} r_i = \{\{2\} : 1, \{1, 2, 3\} : 0, \Theta : 0\}$. Thus there is no common focal element and r_k cannot be computed.



```
x = MassFunction({'7':0.7,'68':0.2,'1234567890':0.1})
for i in range(10):
    x = x & x
print("mass function after 10 DRC:",x)
```

```
mass function after 10 DRC: {{'7'}:1.0; {'9', '8', '3', '6', '7', '1', '4', '5', '0', '2'}:0.0; {'6', '8'}:0.0}
```

```
y = MassFunction({'2':0.7,'12':0.2,'1234567890':0.1})
for i in range(10):
    y = y & y
print("mass function after 10 DRC:",y)
```

```
mass function after 10 DRC: {{'2'}:1.0; {'9', '8', '3', '6', '7', '1', '4', '5', '0', '2'}:0.0; {'1', '2'}:0.0}
```

```
x&y
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
```

Figure 4.3: error example

Another intention of this thesis was to compare the DRC approximation algorithm using Constraint Programming, as presented in [8], in a real-life problem such as the work we implemented. Unfortunately, the implementation of the DRC Constraint Programming approach has not been possible because it requires an ECLiPSe prolog environment to run, and although there is a python library that connects ECLiPSe to Python it only seems to work for Python 3.3 which is now obsolete. In an attempt to enable communication of Python and Eclipse possible, we tried to use Python external commands and we managed to execute ECLiPSe source files, but this caused new problems.

4.3 Results

To produce the results, we used the K-fold Cross-Validation method for K = 10, in which we divided our data into K equal parts. For K repetitions, a different part is chosen each time as the test set, i.e. we try to predict the ratings from 0 to 9, while all the other parts consist the training set that we use to make the predictions. The final result is the average score of the K repetitions. Note, the results are scaled from 0 to 5.

Table 4.10 presents the MAE, RMSE, and DS-MAE errors of the Small dataset for a different number of ratings and neighbors.

Table 4.10: Small dataset, error per number of ratings and neighbors

Ratings size	Neighbors size	MAE	RMSE	DSMAE	Number of predictions	Possible predictions	Prediction time (seconds)
50,000	20	0.86	0.81	1.13	1950	5000	8.57
	100	0.78	0.75	1.05	4318		40.45
	500	0.82	0.81	1.10	4718		118.27
	1000	0.82	0.81	1.10	4718		119.28
100,000	20	0.86	0.81	1.13	3402	10000	16.47
	100	0.78	0.75	1.05	7832		63.19
	500	0.77	0.77	1.04	9615		414.76
	1000	0.78	0.78	1.05	9630		445.95
200,000	20	0.86	0.81	1.13	3385	10084	18.66
	100	0.78	0.75	1.05	7895		74.41
	500	0.77	0.76	1.04	9693		485.55
	1000	0.78	0.77	1.05	9708		522.02

Table 4.11 presents the calculation time of mass ratings, pignistic ratings and users' distances of the Small dataset for a different number of ratings.

Table 4.11: Small dataset, time per number of ratings

Number of ratings	Time (seconds) to calculate mass functions	Time (seconds) to calculate pignistic ratings	Time (seconds) to calculate users' distances
50,000	0.42	2.37	4.26
100,000	1.09	4.70	12.29
200,000	1.29	4.56	12.32

Respectively Table 4.12 and Table 22 4.4 shows the errors and the times of the Full data-set.

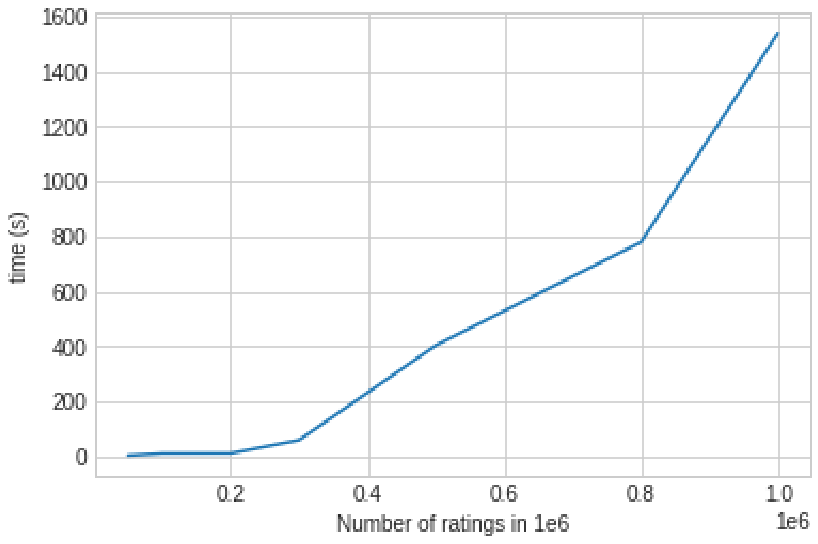
Table 4.12: Full dataset, error per number of ratings and neighbors

Ratings size	Neighbors size	MAE	RMSE	DSMAE	predictions	Possible predictions	Predictions time (seconds)
200,000	20	0.93	0.88	1.22	5299	20,000	36.60
	100	0.87	0.84	1.17	11803		101.60
	500	0.76	0.75	1.06	18520		626.80
	1000	0.78	0.77	1.07	19365		1434.34
500,000	20	0.90	0.86	1.19	13704	50,000	98.05
	100	0.88	0.85	1.18	25935		225.58
	500	0.77	0.76	1.08	42068		1126.10
	1000	0.73	0.72	1.02	46922		2653.47
800,000	20	0.89	0.84	1.17	21797	80,000	221.03
	100	0.89	0.85	1.19	40038		425.77
	500	0.89	0.77	1.09	65023		1809.05
	1000	0.74	0.73	1.04	72333		3736.13
1,000,000	20	0.89	0.84	1.17	27496	100,000	362.07
	100	0.89	0.85	1.19	50220		609.47
	500	0.80	0.78	1.11	80199		2255.27
	1000	0.75	0.74	1.06	89427		4669.89

Table 4.13: Full dataset, time per number of ratings

Number of ratings	Time (seconds) to calculate mass functions	Time (seconds) to calculate pignistic ratings	Time (seconds) to calculate users' distances
200,000	2.46	9.49	63.25
500,000	6.63	22.24	316.43
800,000	11.89	36.11	767.38
1,000,000	16.97	48.53	1279.99

Figure 4.4: Full dataset, time of calculating distances per number of ratings



The results presented in the above tables are indicative of the fact that the model does not make arbitrary predictions. We believe that the predictions produced have the advantage of expressing more information than conventional recommendation systems. Thus the main advantage of our model lies in the mass functions it produces as a result, which can be useful in handling the uncertainty of a real-life recommendations system.

5. CONCLUSIONS AND FUTURE WORK

It is clear that movie Recommendation Systems are an important field of research and will be in the spotlight for a long time. One crucial characteristic that is to be addressed in the future studies is the Uncertainty that the systems are called upon to manage by imprecise user ratings. We deem that Dempster-Shafer's theory provides the proper tools to handle uncertainties emerging from an RS.

In this thesis, we implemented a model using Dempster-Shafer's theory that takes into consideration this uncertainty and expresses it in the final result. In the implementation process, we met and faced many problems regarding the time complexity, the modeling of Dempster-Shafer theory and the accuracy of the final result. We believe that the foremost advantage of our model is its capability to express the predicted ratings using mass functions. Mass functions provide us with a measure of uncertainty about the predicted rating that the system can take into account in deciding whether or not to recommend a product to a user.

Future work will be focused on additional evaluation measures, performance improvements and further evaluation of our model as a real-world application. Specifically, the Precision, Recall and F-score accuracy measures play a vital role in the assessment of classification problems as well as RS. We also consider it important, to further develop methods to examine how a model can use the uncertainty of the predictions to make more reliable suggestions.

As for our model, it is necessary to improve time and space complexity. In our implementation, we have already created optimizations that significantly reduce the time of calculating the users' distances. However, the calculation of suggestions needs optimizations in terms of time complexity. Finally, it would be worthwhile to use approximation algorithms to further study their efficiency in a real-world application.

ABBREVIATIONS - ACRONYMS

DST	Dempster-Shafer theory
bpa	basicprobability assignment
DRC	Dempster rule of combination
RS	Recommendation Systems
MAE	Mean Absolute Error
RMSE	Root Mean Square Error

BIBLIOGRAPHY

[1] Movielens latest datasets, https://grouplens.org/datasets/movielens/latest/, Mar 2021.

[2] Mathias Bauer. Approximation algorithms and decision making in the dempster-shafer theory of evidence — an empirical study. *International Journal of Approximate Reasoning*, 17(2):217–237, 1997. Uncertainty in AI (UAI'96) Conference.

[3] Isabelle Bloch. Some aspects of dempster-shafer evidence theory for classification of multi-modality medical images taking partial volume effect into account. *Pattern Recognition Letters*, 17(8):905–919, 1996.

[4] Hei Chan and Adnan Darwiche. A distance measure for bounding probabilistic belief change. *International Journal of Approximate Reasoning*, 38(2):149–174, 2005.

[5] Qi Chen, Amanda Whitbrook, Uwe Aickelin, and Chris Roadknight. Data classification using the dempster–shafer method. *Journal of Experimental & Theoretical Artificial Intelligence*, 26(4):493–517, 2014.

[6] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Ann. Math. Statist.*, 38(2):325–339, 04 1967.

[7] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[8] Alexandros Kaltsounidis. *Dempster-Shafer Theory Computation using Constraint Programming*. BSc Thesis, Departmnet of Informatics and Telecommunications, NKUA, 2019.

[9] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Recommendation Systems*, page 292–324. Cambridge University Press, 2 edition, 2014.

[10] C.L. Blake D.J. Newman and C.J. Merz. UCI repository of machine learning databases, 1998.

[11] Van-Doan Nguyen and Van-Nam Huynh. Two-probabilities focused combination in recommender systems. *International Journal of Approximate Reasoning*, 80:225 – 238, 2017.

[12] Pekka Orponen. Dempster's rule of combination is #p-complete. *Artificial Intelligence*, 44(1):245–253, 1990.

[13] Kamal Premaratne, Thanuka Wickramarathne, and M. Kubat. Cofids: A belief theoretic approach for automated collaborative filtering, 02 2011.

[14] Thomas Reineking. py_dempster_shafer, https://pypi.org/project/py_dempster_shafer/#description.

[15] Thomas Reineking. *Belief Functions: Theory and Algorithms*. PhD thesis, 03 2014.

[16] Aikaterini Rousaki. *Music Recommendation Systems using Dempster Shafer Theory*. BSc Thesis, Departmnet of Informatics and Telecommunications, NKUA, 2020.

[17] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[18] Philippe Smets. *What is Dempster-Shafer's Model?*, page 5–34. John Wiley & Sons, Inc., USA, 1994.

[19] Philippe Smets. Practical uses of belief functions, 2013.

[20] Márcio Soares and Paula Viana. Tuning metadata for better movie content-based recommendation systems. *Multimedia Tools and Applications*, 74, 04 2014.

[21] Luigi Troiano, Luis J. Rodríguez-Muñiz, and Irene Díaz. Discovering user preferences using dempster–shafer theory. *Fuzzy Sets and Systems*, 278:98 – 117, 2015. Special Issue on uncertainty and imprecision modelling in decision making (EUROFUSE 2013).

[22] Frans Voorbraak. A computationally efficient approximation of dempster-shafer theory. *International Journal of Man-Machine Studies*, 30(5):525–536, 1989.

[23] Nic Wilson. A monte-carlo algorithm for dempster-shafer belief, 2013.

[24] Nic Wilson and Serafin Moral. Fast markov chain algorithms for calculating dempster-shafer belief. In *IN PROCEEDINGS OF THE 12TH EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 672–676. John Wiley & Sons, 1996.

[25] L.A. Zadeh. On the validity of dempster's rule of combination of evidence. Technical Report UCB/ERL M79/24, EECS Department, University of California, Berkeley, Mar 1979.