NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Music Recommendation Systems using Dempster Shafer Theory

Aikaterini E. Roussaki

ATHENS

OCTOBER 2020

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα συστάσεων μουσικής με χρήση της θεωρίας Dempster Shafer

Αικατερίνη Ε. Ρουσσάκη

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2020

BSc THESIS

Music Recommendation Systems using Dempster Shafer Theory

Aikaterini E. Roussaki
S.N.: 1115201400174

SUPERVISOR: Izambo Karali, Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα συστάσεων μουσικής με χρήση της θεωρίας Dempster Shafer

Αικατερίνη Ε. Ρουσσάκη
Α.Μ.: 1115201400174

ΕΠΙΒΛΕΠΟΥΣΑ: Ιζαμπώ Καράλη, Επίκουρη Καθηγήτρια

ABSTRACT

In this thesis, we deal with the subject of Handling Uncertainty by using Dempster-Shafer Theory of Evidence. The purpose of this project is to use the subject of handling uncertainty as a recommendation technique in a Music Recommendation System (MRS). We use the Dempster's rule of combination and more specifically a Monte Carlo algorithm which is an approximation algorithm to compute the rule. Both the Music Recommendation model and the Dempster's rule of combination are implemented with the use of the programming language Python and it's libraries.

SUBJECT AREA: Artificial Intelligence
KEYWORDS: uncertainty, dempster-shafer, approximation algorithms,montecarlo, recommendation systems, music recommendation, Python, music-metadata

ΠΕΡΙΛΗΨΗ

Στην πτυχιακή εργασία αυτό που μας απασχολεί είναι το θέμα του Χειρισμού της Αβεβαιότητας με την χρήση της θεωρίας Dempster Shafer Theory of Evidence. Ο σκοπός αυτής της μελέτης είναι να χρησιμοποιήσουμε το θέμα του Χειρισμού της Αβεβαιότητας σαν τεχνική σύστασης σε ένα Σύστημα Συστάσεων Μουσικής. Θα χρησιμοποιήσουμε τον κανόνα συνδυασμού του Dempster και πιο συγκεκριμένα θα τον υλοποιήσουμε με έναν προσεγγιστικό αλγόριθμο, Monte-Carlo. Η υλοποίηση του μοντέλου για το Σύστημα Συστάσεως Μουσικής καθώς και η υλοποίηση του προσεγγιστικου αλγορίθμου έχει γίνει με την χρήση τής γλώσσας Python όπως και με την βοήθεια βιβλιοθηκών της.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αβεβαιότητα, dempster-shafer, προσεγγιστικοί αλγόριθμοι, monte-carlo, συστήματα συστάσεων, συστημα συστάσεως μουσικής, Python, μεταδεδομένα μουσικής

AKNOWLEDGMENTS

The first person that i would like to thank is my supervisor, Mrs. Izambo Karali. I am truly thankful, for her support and encouragement not only in academic but also in personal level. Also i need to thank her for the time that she spend to help me in every step of this thesis by numberless conversations and emails.

I would like to thank my friend Alexandros Kaltsounidis without whom i never would have thought to approach Mrs. Izambo Karali and write about this thesis. Another important person that i thank is my friend Mercouris Papamichail for all the hours that he had to put up with me throughout all this time.

Last but not least i give my warmest thanks to my family for the love and support that kepted me going throughout all my academic years.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

With the invention of the World Wide Web every person can stay connected with just the use of a computer and an internet connection. Also a new way to buy or sell products on online services or over the Internet which is called e-commerce has began [13]. Thus, companies should be able to learn their customers needs and preferences in order to become more profitable and grow. This need has lead to create and use the Recommendation Systems(RS) almost everywhere. Nowadays they have become very popular as they are used by major companies such as Amazon [14] [7], YouTube[7], Twitter[15], Netflix [7], LastFm [7], etc., for music recommendations to even everyday product recommendations.

In order for the RS to make a recommendation it will need information about the user preferences[16]. Information about users can be collected explicitly by asking the user for a rating on a certain item or implicitly by keeping information about the users, for example keeping their purchase history. With this information it is easy to predict for each user products that they will like.

Because of the unprecedented scale of readily available content, the user can easily become overwhelmed [16]. Thus this creates a problem which we will show with an example about music recommendation systems.

In the offline world we would walk inside a shop and ask one of the employees about a CD with pop music that was released in 2014 and publisher was Heaven music. Lets say that there are more than 30+ options that meet these criteria, so the employee will ask you more questions, like the name of the primary artist, and will present you less than 10 song options, that will be tailored to your preferences. So there always can be a reasonable limit in the recommendations that the employee will present. On the other hand in an online request the first recommendations would be filters and the options will be again 30+. Thus the number of the song options depends only from the filters that user will select each time.

For this problem the RS gives a solution. By collecting information about the user preferences we can recommend products that have a higher probability of being preferred. In the above example for the online request, we could analyze the music preferences of a user, and then show first the songs that have a higher probability of being preferred.

In this thesis we will research the Music Recommendation Systems (MRS). The MRS are very useful as the digital music distribution has led to an ubiquitous availability of music. Due to the lack of explicitly available user feedback data, music recommendation techniques tend to rely more upon content descriptions of items. Thus it is more advisable to use content-based recommendation techniques for MRS. A content-based MRS could use the song metadata, which are characteristic information about a song such as the song title, artist name, release year, etc. or even apply signal processing and analysis in order to find similarity between songs [16].

As mentioned before in the online music store example there is a gap between a physical and an online store's recommendations. Thus in this thesis we will face this problem with a

Music RS that recommends a homogeneous group of users with the most suitable songs for their tastes. More specifically, we will filter a list with songs from an online collection of audio features and metadata called Million Song Dataset (MSD) [17] and a list with user song selections from a social music website called Thisismyjam [18] in order to find a set of songs that have a higher probability of being preferred.

The model [6] that we will use for the recommendations is going to have a different detail which is that the recommendation will be on the item features and not on the items. This might be considered strange but is very helpful as it will reduce the problem dimensionality and can meet user preferences even when they are not made explicit.

As said above the RS is needed to generate a short list of suitable items to a specific user among a huge number of potential items. The problem is the uncertainty that is created by the the information that the users provides with their preferences as they commonly are uncertain, imprecise or incomplete [31]. Thus this will be a problem with the Music RS that is introduced in this thesis.

A solution in this problem is to use the the Dempster Shafer Theory of Evidence (DS). The DS is a general framework for reasoning with uncertainty and allows us to combine evidence from different independed sources, like the features of each song, and arrive at a degree of belief. Thus in order to find the best recommendations after the preferences are induced by each feature we will use the DS.

For the implementation of the DS Theory in this model we will use an approximation algorithm and more specifically a Monte Carlo algorithm instead of the more traditional method. We made this choice because of the complexity of the Dempste's rule of combination, as is a problem known to be #P-complete. Thus with the Monte Carlo algorithm we will use an optimized algorithm which implements the Dempster's rule of combination.

2. DEMPSTER SHAFER THEORY

2.1 Overview

Dempster-Shafer theory (DS) [1], [2], or also known as theory of belief functions or evidence theory is a general framework, developed by Arthur P. Dempster and Glenn Shafer, for reasoning with uncertainty. In this theory instead of considering a precise value p for the probabilistic measure associated with an event, a pair of lower and upper probabilities are used to include a set of probabilities to quantify uncertainties. This framework allows for the probability of the hypothesis to be represented as intervals with the help of the mass function or basic probability assignment, bounded by two values, belief and plausibility.

2.2 Definitions

Given a question of interest the frame of discernment denoted by Θ or else Universe U, is a finite set of possible answers to the question.

The set of all the subsets of Θ is 2^{Θ} and can include the empty set $\emptyset$ $2^{\Theta} = \{A | A \subset \Theta\}$. A subset A represents the a statement that the truth lies in A [4], [10].

2.2.1 Basic Definitions

Definition 2.2.1. *(Mass Function) The mass function or else called basic probability assignment (bpa), as mentioned above is a mathematical entity for assigning degrees of belief to a set of hypothesis.*

A function m: $2^{\Theta} \rightarrow [0, 1]$

$$m(\emptyset) = 0 \text{ and } \sum_{A \in 2^{\Theta}} m(A) = 1$$

From the mass assignments, the upper and lower bounds of a probability interval can be bounded by two measures called belief (or support) and plausibility. We will analyze them more in detail below.

The interval upper bound will be the plausibility of the subset A and the lower bound will be the belief of the subset A.

$$Bel(A) \leq P(A) \leq Pl(A) \tag{2.1}$$

or else it can be interpreted as

$$P(A) \in [Bel(A), Pl(A)] \tag{2.2}$$

Definition 2.2.2. *(Focal Elements) Focal element of a belief function over Θ is a subset A of Θ for which it has to apply $m(A) > 0$ [10]. The union of all the focal elements of a belief function is called its core [3].*

Definition 2.2.3. *(Belief Function) Belief function is a real function over the subsets Bel $2^{\Theta} \rightarrow [0,1]$. The belief or support for a set A, expresses the total amount of belief committed to A and is defined as the sum of the masses of all subsets of set A.*

$$Bel(A) = \sum_{B \subseteq A} m(B) \quad \forall A \subseteq \Theta \tag{2.3}$$

The following can be concluded for the belief function: $Bel(\Theta) = 1 \quad Bel(\emptyset) = 0$

Definition 2.2.4. *(Plausibility Function) The plausibility for a set A $\forall A \subseteq \Theta$, expresses the total amount of belief not committed to the negation of A.*

$$Pl(A) = Bel(\Theta) - Bel(\overline{A}) \tag{2.4}$$

From the definition of the belief function we have concluded that $Bel(\Theta) = 1$ *which, will help transform the plausibility definition into the following.*

$$Pl(A) = 1 - Bel(\overline{A}) \tag{2.5}$$

We can conclude the following from the definition: $Pl(\Theta) = 1 \quad Pl(\emptyset) = 0$

Proposition 1. *The plausibility of a set A can be computed from a mass function, as the sum of the masses of all the sets that intersect with the set A [10].*

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \quad \forall A \subseteq \Theta \tag{2.6}$$

Proof. We will use the definition of the plausibility function in order to prove the above

$$Pl(A) = Bel(\Theta) - Bel(\overline{A}) \quad \forall A \subseteq \Theta$$

From the definition of the belief function we know that

$$Bel(\Theta) = \sum_{B \subseteq \Theta} m(B) \text{ and } Bel(A) = \sum_{B \subseteq A} m(B) \quad \forall A \subseteq \Theta$$

So the plausibility definition becomes as follows

$$Pl(A) = \sum_{B \subseteq \Theta} m(B) - \sum_{B \subseteq A} m(B) \quad \forall A \subseteq \Theta$$

From the above definition we derive that the plausibility of the subset A equals with the sum of the masses of all subsets of the Θ (frame of discernment) minus the sum of the masses of all subsets of A. From the subtraction derives that the plausibility of the subset A equals to the sum of the masses of all the subsets of Θ that intersect with A.

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B)$$

□

Example 1.

In order to understand the definitions and usages of the mass, belief and plausibility functions we will give a simple but helpful example that uses all the above.

Assume that there is a professional seamstress which has a YouTube channel that shows sewing projects. In order to make a new sewing video she asks her fans about what style do they want the new cloth design to be. Thus she uses a special poll program using the following choices casual (c), vintage (v), sportswear(s). The frame of discernment Θ of this example is $\Theta = \{c, v, s\}$. A fan is able to choose one or more styles of his/her choice. The results from the poll are be the following:

Firstly the program gather all the choices and finds the masses of them
$m(\{c\}) = 0.1$
$m(\{v\}) = 0.75$
$m(\{s\}) = 0.05$
$m(\{c, v\}) = 0.1$

The belief and plausibility of each choice and their combination will be computed by the program using the definitions of the belief and the plausibility function by using 2.2.3, 2.2.4.

Table 2.1: Belief and plausibility of only the focal elements

Style	Belief	Plausibility
Casual	0.1 ($m(\{c\})$)	0.2 ($1 - m(\{v\}) - m(\{s\})$)
Vintage	0.75 ($m(\{v\})$)	0.85 ($1 - m(\{c\}) - m(\{s\})$)
Sportswear	0.05 ($m(\{s\})$)	0.05 ($1 - m(\{v\}) - m(\{c\}) - m(\{c, v\})$)
Casual,Vintage	0.95 ($m(\{c, v\}) + m(\{c\}) + m(\{v\})$)	0.95 ($1 - m(\{s\})$)

The second way to compute the plausibility is by using the proposition 1.

Table 2.2: Belief and plausibility of only the focal elements

Style	Belief	Plausibility
Casual	0.1 ($m(\{c\})$)	0.2 ($m(\{c\}) + m(\{c, v\})$)
Vintage	0.75 ($m(\{v\})$)	0.85 ($m(\{v\}) + m(\{c, v\})$)
Sportswear	0.05 ($m(\{s\})$)	0.05 ($m(\{s\})$)
Casual,Vintage	0.95 ($m(\{c, v\}) + m(\{c\}) + m(\{v\})$)	0.95 ($m(\{c, v\}) + m(\{v\}) + m(\{c\})$)

A diagrammatic representation(or here called lattice) of 2^Θ for the styles example is given in Fig. 1. Please note that a set of size n has 2 n subsets. The empty set, $\emptyset$, is one of these subsets.

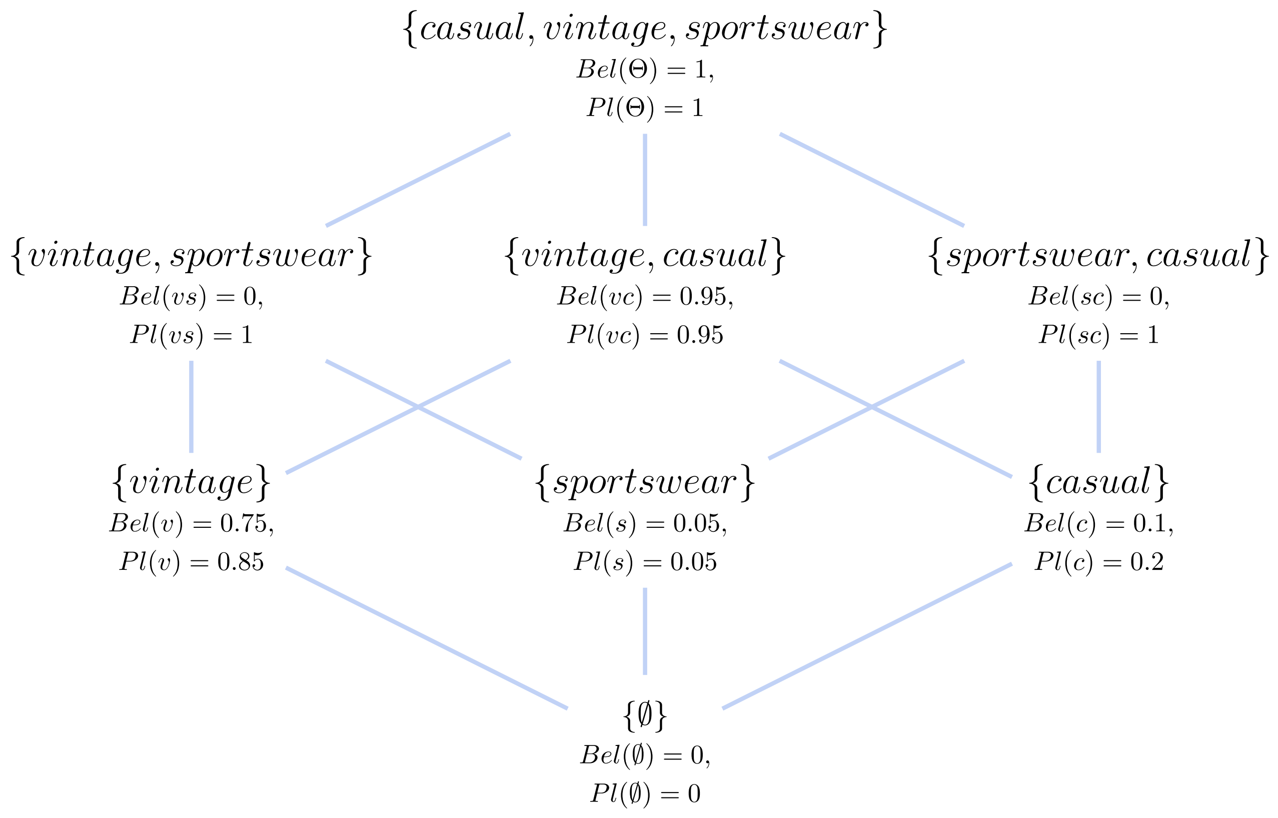

Figure 2.1: Lattice of the styles

Table 2.3: Belief and plausibility of all the subsets

Style	**Belief**	**Plausibility**
Casual	0.1	0.2
Vintage	0.75	0.85
Sportswear	0.05	0.05
Casual,Vintage	0.1	0.95
Sportswear,Vintage	0	1
Casual,Sportswear	0	1
Sportswear,Vintage,Casual	0	1

If we examine closely the results of the program we can see that the style that is mostly likely to be liked from the fans of the channel is the vintage. This is because the interval of the possibility of the vintage style, $0.75 \leq P(Vintage) \leq 0.85$*, has a small range and the plausibility* $Pl(Vintage)$ *is close to* 1. △

2.2.2 Dempster's Rule Of Combination

Dempster's rule of combination was first introduced by Dempster in 1967 and then reinterpreted by Shafer [3]. It allows combining distinct, normalized belief functions that are defined over the same frame of discernment Θ.

Let m_1 and m_2 be two mass functions defined over the same frame of discernment Θ. Using Dempster's rule of combination we compute a new mass function $m_{1,2}$ that integrates the evidence of both m_1 and m_2 [16].
For the Dempster's rule of combination we use the symbol ⊕. Thus the combination of two mass functions m_1 and m_2 will be $m_{1,2} = m_1 \ \oplus \ m_2$.

$$m_{1,2}(A) = \frac{\sum\limits_{C \cap B = A} m_1(C) \cdot m_2(B)}{1 - \sum\limits_{C \cap B = \emptyset} m_1(C) \cdot m_2(B)}$$

$$m_{1,2}(A) = \frac{1}{1 - \sum\limits_{C \cap B=\emptyset} m_1(C) \cdot m_2(B)} \cdot \sum_{C \cap B=A} m_1(C) \cdot m_2(B)$$

$$m_{1,2}(A) = \frac{1}{1-k} \cdot \sum_{C \cap B = A} m_1(C) \cdot m_2(B), \quad k = \sum_{C \cap B = \emptyset} m_1(C) \cdot m_2(B), \quad \forall A \subseteq \Theta, A \neq \emptyset \tag{2.7}$$

For the above definition we need to know that

$$k = \sum_{C \cap B = \emptyset} m_1(C) \cdot m_2(B) \tag{2.8}$$

k is a measure of the amount of conflict between the two mass sets. $K = 1 - k$ is a normalization constant that can serve as a measure of the extend of the conflict between two mass functions. K has the effect of completely ignoring conflict and attributing any mass associated with conflict to the null set $\emptyset$.

The following can be concluded for the Dempster's rule of combination: $m_{1,2}(\emptyset) = 0$

In order to understand the combination rule we need to recall some basic algebraic notions that apply for the above rule [11].

Dempster combination rule properties

- Commutative property $m_1 \oplus m_2 = m_2 \oplus m_1$
- Associative property $(m_1 \oplus m_2) \oplus m_2 = m_1 \oplus (m_2 \oplus m_3)$
- Neutral element $m_1 \oplus m_2 = m_1,$ where $m_2 : 2^{\Theta} \to [0,1],$ and $m_2(\{\Theta\}) = 1$

Example 2.

With the following example we will understand the practical use of the Dempster's rule of combination. In this example we will use the Dempster's rule of combination mathematical formula 2.7 with the use of tables. Will use the tables because it will be easier to understand the practical use of the combination rule.

Suppose that we are patients and we have made an appointment with a doctor about our current condition. The condition that we might have can be the following pneumonia (p), flu (f), allergy(a), common cold (c). The frame of discernment Θ is $\Theta = \{a, p, f, c\}$.

In order for the doctor to make a valid examination, we will be asked a series of questions about our condition. He asks as if we are experiencing sneezing or in general nasal *discharge. Indeed for the past couple days we have nasal discharge. So there is a mass* $function\,1$ *that:*

$m_1(\{a, f, c\}) = 0.5$
$m_1(\{\Theta\}) = 0.5$

The doctor asks if we had a loss for our appetite in the past few days and we answer him/her that indeed we had. So he/she tells us that the condition might be flu or pneumonia, with:

$m_2(\{p, f, c\}) = 0.7$
$m_2(\{\Theta\}) = 0.3$

The table consists of the first row that has the focal points of the first mass function m_1*, the first column has the focal points of the second mass function* m_2 *and in each of the other cells we have calculated the intersection of the focal points.*

Table 2.4: Dempster rule of combination at the mass function m_1 and m_2

m_2 / m_1	$\{a, f, c\}$	Θ
$\{p, f, c\}$	$\{f, c\}$ / 0.35	$\{p, f, c\}$ / 0.35
Θ	$\{a, f, c\}$ / 0.15	$\{\Theta\}$ / 0.15

From the use of the Dempster rule of combination used in the table above the new mass function now will be the $m_{1,2} = m_1 \oplus m_2$*. The mass function* $m_{1,2}$ *did not have any conflict as we can see from the table and from the definition of the K 2.8*

$m_{1,2}(\{a, f, c\}) = 0.15$
$m_{1,2}(\{f, c\}) = 0.35$
$m_{1,2}(\{p, f, c\}) = 0.35$
$m_{1,2}(\{\Theta\}) = 0.15$

Lastly the doctor asks us to take a chest X-ray. The X-ray shows that hew have many small inflammations on the right lung. Thus the doctor is almost sure that we suffer from pneumonia.

$m_3(\{p\}) = 0.85$
$m_3(\{\Theta\}) = 0.15$

Table 2.5: Dempster rule of combination at the mass function m_3 and $m_{1,2}$

m_3 \ $m_{1,2}$	$\{a, f, c\}$	$\{f, c\}$	$\{p, f, c\}$	Θ
$\{p\}$	$\{\emptyset\}$ 0.1275	$\{\emptyset\}$ 0.2975	$\{p\}$ 0.2975	$\{p\}$ 0.102
Θ	$\{a, f, c\}$ 0.0225	$\{f, c\}$ 0.0252	$\{p, f, c\}$ 0.0525	$\{\Theta\}$ 0.1275

The new mass function will be the $m_{1,2,3} = m_3 \oplus m_{1,2}$. There are conflicts as see can see from the table above there are cells that contain the $\{\emptyset\}$. Thus the k of the definition (2.8) is:

$$k = \sum_{A \cap B = \emptyset} m_3(A) \cdot m_{1,2}(B) = 0.1275 + 0.2975 = 0.425$$

$$m_{1,2,3}(\{p\}) = \frac{1}{1-0.425} \cdot (0.2975 + 0.1275) \approx 0.7391$$

$$m_{1,2,3}(\{a, f, c\}) = \frac{1}{1-0.425} \cdot 0.0225 \approx 0.0391$$

$$m_{1,2,3}(\{p, f, c\}) = \frac{1}{1-0.425} \cdot 0.0525 \approx 0.0913$$

$$m_{1,2,3}(\{\Theta\}) = \frac{1}{1-0.425} \cdot 0.1275 \approx 0.2217$$

Now we need to find the interval of the probability that we indeed have pneumonia $P(\{p\})$*. Again from the definition of the belief function and the plausibility function we find the plausibility* $Pl(\{p\}) = 0.8171$ *and the belief* $Bel(\{p\}) = 0.6945$*.*

Thus the probability for us to suffer from pneumonia is $0.6945 \leq P(\{p\}) \leq 0.8171$*.*

$\triangle$

2.2.3 Dempster rule of combination conflicts

As pointed out by Zadeh (1979) normalization can lead to serious problems in the case of what has come to be known as the Dempster rule of combination. The Dempster rule of combination can not be applied if there is not at least one parent relation which is conflict free. Thus it is not permitted to combine distinct probability distributions which means that there must be at least one combination of focal points that intersects. With other words the rule of combination can not be used if there is not even one combination of focal points that intersects which translates as $k = \sum_{A \cap B = \emptyset} m_1(A) \cdot m_2(B) = 1$. If indeed $k = 1$ from the definition of the rule of combination we would have a division with zero which is an undefined expression [12].

With an example we will show how the counter effective results that the Dempster rule of combination can generate when there is a high degree of conflict.

Example 3.

Assume that you are a doctor and a patient comes to the clinic that we work and complains that he has a persistent headache for the past few weeks and that the headache will not stop with mild analgesics.

After a thorough examination of the patient, blood tests and a CT scan, we believe that the patient either has a brain tumor (t) with bpa 0.99 or meningitis (m) with bpa 0.01.

Because both of the severity of both illnesses we ask the opinion of a second doctor that works for the clinic. The second doctor believes that our patient has a migraine headache (h) with probability 0.99 or meningitis with probability 0.01.

We apply the rule of combination in order to combine the two mass functions.

Our examination

$m_1(\{t\}) = 0.99$

$m_1(\{m\}) = 0.01$

Examination of the second doctor

$m_2(\{h\}) = 0.99$

$m_2(\{m\}) = 0.01$

Table 2.6: Dempster rule of combination at the mass function m_1 and m_2

m_1 \ m_2	$\{h\}$	$\{m\}$
$\{t\}$	$\{\emptyset\}$ 0.9801	$\{\emptyset\}$ 0.0099
$\{m\}$	$\{\emptyset\}$ 0.0099	$\{m\}$ 0.0001

The new mass function after the rule of combination is $m_{1,2} = m_1 \oplus m_2$

$$m_{1,2}(\{m\}) = \frac{1}{1 - 0.9999} \cdot 0.0001 = 1$$

As we can see the meningitis is diagnosed with 100 percent of confidence, but this result goes against the common sense since both of the doctors agree there is a little chance that the patient has meningitis.

△

3. APPROXIMATION ALGORITHMS AND DEMPSTER-SHAFER THEORY OF EVIDENCE

3.1 Overview

The Dempster's rule of combination as shown above is a formula for combining evidential information provided by different sources. A major drawback of the Dempster's rule is it's computational complexity. The problem was pointed out by Barnett in his first paper introducing the Dempster-Shafer theory [23]. The computational complexity for applying this rule is in the worst case will be exponential [23],[22],[20]. More specifically the Orphonen [22] has shown that the combination of two mass functions or basic probability assignments using Dempster's rule is #P-complete.

In order to overcome this problem various strategies and methods have been suggested. Some of them are, the exploitation of independence, deterministic algorithms and the approximation algorithms [20],[21]. In this chapter we will study about the approximation algorithms as an optimization for the Dempster rule of combination. Approximation algorithms are efficient algorithms that find approximate solutions to optimization problems with provable guarantees on the distance of the returned solution.

Two categories of approximation algorithms used in order to optimize the Dempster's rule of combination are the Monte Carlo algorithms and algorithms that reduce the number of the focal elements in belief function [21], which we will call input size reduction algorithms. We known that the number of the computations is directly related to the number of focal elements that each mass function has.Thus algorithms that aim to reduce that number or redistributing the corresponding numerical values can help reduce the time needed for computing Dempster's rule of combination. The former category, Monte Carlo algorithms, calculate belief directly. More specifically they approximate belief up to arbitrary accuracy. This calculation has very low complexity [25].

In this chapter we will show epigrammatically some of the most known approximation algorithms for optimizing the Dempster's rule of combination, from both Monte Carlo category as well as the ones that aim at reducing the number of the focal elements. The outline of the overview presented in this section is based on [9]. Then we will focus more on the implementation of the DS theory with a Monte Carlo algorithm [20] as we will use in the Music RS of this thesis.

3.2 Input size reduction algorithms

3.2.1 Bayesian Approximation

The Bayesian Approximation [26] reduces a given mass function m to a discrete probability distribution (a discrete probability distribution is a probability distribution that can take on a countable number of values [27]). This means that only singleton subsets of the frame of discernment Θ are allowed to be focal elements of the Bayesian mass function $\underline{m}$.

The Bayesian approximation is defined as:

$$\underline{m}(A) = \begin{cases} \frac{\sum_{B:A\subseteq B} m(B)}{\sum_{C:C\subseteq\Theta} m(C)\cdot|C|}, & |A| = 1, \\ 0, & otherwise. \end{cases}$$

3.2.2 k-l-x Method

The k-l-x method [28] combines the highest valued focal elements of the original mass function m into the approximation m_{klx}, and assigns zero to all the other subsets of the frame of discernment Θ. The selection of the focal elements for the k-l-x method must be subjected to the following rules:

- The approximation m_{klx} can contain at most l focal elements.
- The minimum number of focal elements must be k.
- The accumulated mass of all focal elements of m that are not included in m_{klx} is restricted to be at most x, where $x \sim [0, 1]$.

3.2.3 Summarization Method

The Summarization method [21] is very similar to the k-l-x method that we have shown. In this method we won't change the best-valued focal elements of the mass function. The numerical values of the remaining focal elements are gathered and then assigned to the union of the corresponding subsets of the frame of discernment Θ. The Summarization method is defined as:

$$m_s(A) = \begin{cases} m(A), & A \in M \\ \sum_{A'\subseteq A,\, A'\notin M} m(A'), & A = A_0 \\ 0, & otherwise. \end{cases}$$

- The number of the focal elements that are contained in a given mass function m are denoted by k.
- The set of the $k-1$ subsets of Θ with the highest values in m is denoted by M.

- The A_0 is defined as $A_0 = \bigcup\limits_{\substack{A' \notin M \\ m(A') > 0}} A'$.

3.3 Monte Carlo Algorithms

Monte Carlo algorithms are another category of approximation algorithms that can be used to optimize the Dempster rule of combination. They make a number of random choices on the basis of some probability distribution. The results of a Monte Carlo algorithm might be incorrect in a limited number of cases and correct within a certain probability. The Monte Carlo algorithms for finding combined belief are much faster as they are almost linear in the size of the data.

Nic Wilson has described in details the Monte Carlo algorithms in his papers [24], [25], [30]. More specifically the belief is estimated using a large number L of trials of a random algorithm. Each trial of the algorithm gives an estimate of belief, which is either 0 or 1. The average of the estimates of all the trials converges to the correct value of the estimates of the L gets larger. It is not feasible to calculate the belief of all the subjects of the frame of discernment Θ when the frame is large. Thus it is assumed that for a small number of important sets $A \subseteq \Theta$, we are interested in calculating the belief $Bel(A)$. The algorithms involve random sampling, thus they can be expressed in terms of source triplets. A source over Θ is a triple (Ω, P, Γ), Ω is a fine set, P is a strictly positive probability distribution over Ω and Γ is a function from Ω to $2^{\Theta} - \{\emptyset\}$. A source triple is associated with a mass function m and a belief function Bel, $m(A) = \sum\limits_{\omega : \Gamma(\omega)} P(\omega)$ and $Bel(A) = \sum\limits_{\omega : \Gamma(\omega) \subseteq A} P(\omega)$. Dempster's rule of combination for source triples is a mapping sending a finite set of source triples $\{(\Omega_i, P_i, \Gamma_i), for i = 1, \cdots, k\}$ to a triple (Ω, P_{DS}, Γ). Let $\overline{\Omega} = \Omega_1 \cdot \cdots \cdot \Omega_k$ and for $\omega \in \overline{\Omega}$, $\omega = (\omega(1), \cdots, \omega(k))$. For $\omega \in \overline{\Omega}$define the function $\Gamma' : \overline{\Omega} \to 2^{\Theta}$ by $\Gamma'(\omega) = \bigcap\limits_{i=1}^{k} \Gamma_i(\omega(i))$ and the probability function $P'(\omega) = \Pi_{i=1}^{k} P_i(\omega(i))$. Let Ω be the set $\{\omega \in \overline{\Omega} : \Gamma'(\omega) \neq \emptyset\}$, let Γ be Γ' restricted to Ω and let probability function P_{DS} on Ω to be P' conditioned on Ω so that for ω $P_{DS}(\omega) = P'(\omega)/P'(\Omega)$. The factor $1/P'(\Omega)$ is a measure of the conflict between the evidences. The combined belief for $A \subseteq \Theta$ is $Bel(A) = P_{DS}(\{\omega \in \Omega : \Gamma(\omega) \subseteq A\})$, which we abbreviate to $P_{DS}(\Gamma(\omega) \subseteq A)$.

3.3.1 Naive Monte-Carlo algorithm

A simple algorithm of Monte-Carlo is based on the papers of the Nic Wilson [30], [24]. For $A \subseteq \Theta,\ Bel(A) = P_{DS}(\Gamma(\omega) \subseteq A)$, in order to calculate the $Bel(A)$ we can repeat a large number of trials of a Monte-Carlo Algorithm. For each trial, we pick ω with chance $P_{DS}(\omega)$ and the trial succeeds if $\Gamma(\omega) \subseteq A$ else the trial fails. The $Bel(A)$ is estimated by the proportions of the trials that succeed.

The simplest algorithm is to pick ω with chance $P_{DS}(\omega)$ by repeatedly picking $\omega \in \overline{\Omega}$ with chance P'_{ω} until we get an ω in Ω. In order to find the ω with chance P'_{ω} we will do the following $for\ each\ i = 1, \cdots, k$, we pick $\omega_i \in \Omega_i$ with chance $P_i(\omega_i)$ and let $\omega = (\omega_1, \cdots, \omega_k)$.

This algorithm is very efficient for problems where the evidenced are not very conflicting [30]. The time that the algorithms takes to achieve a certain accuracy is $|\Theta|k/P'(\Omega)$. The reason that this algorithm is efficient is that the number of the trials needed is not dependent on the size of the problem.

3.3.2 Markov Chain Algorithm

A Markov Chain is a stochastic model describing a sequence of possible events in which the probability of each depends only on the state attained in the previous event. We will form a Markov Chain, so that the result of each trial is dependent only on the result of the previous trial. The Markov Chain algorithms require a condition on Ω to work which we will call ”contentedness” [30]. The contentedness corresponds to the Markov Chain being irreducible. This means that from every state of the chain we can get to any state.

3.4 DS implementation with Monte Carlo Algorithm

The approximation algorithm that we are going to use for the implementation of the Dempster rule of combination is created by the Thomas Reineking for his dissertation ”Belief Functions: Theory and Algorithms”, [9]. He have created a Python library that performs calculations in the Dempster-Shafer Theory of Evidence [20]. The algorithm that is used belongs to the Monte Carlo algorithms and is an approximation algorithm. Here we will show this algorithm and describe all of its steps in detail.

Below we present the Dempster rule of combination between two mass functions m_1 and m_2 implemented with the Monte Carlo. We can change the sample count and if we need to have importance sampling in order to have the best results for our model.

```
m1.combine_conjunctive(m2, sample_count=1000, importance_sampling=True))
```

The function combine_conjunctive conjunctively combines the mass function with another mass function and returns the combination as a new mass function. Both mass functions are assumed to be defined over the same frame of discernment Θ. If normalization is set to true then mass function that results from this combination will be normalized. The importance sampling is what determines the approximation method. If it is set to true then all the empty intersections will be avoided. This leads to a lower approximation errors but is slower. Thus it must be used if there is significant evidential conflict between the mass functions. If the importance sampling is false the method will independently generates samples from both mass functions and computes their intersections.

```
    def combine_conjunctive(self, mass_function, normalization=True,
                            sample_count=None, importance_sampling=False):

        return self._combine(mass_function, rule=lambda s1, s2: s1 & s2,
                             normalization=normalization,
                             sample_count=sample_count,
                             importance_sampling=importance_sampling)
```

The _combine is a helping method that helps to complete the conjunctive combination. It raises an error if the given mass function is not valid. Also it makes the proper combination that depends from the parameters normalization, sample_count, and importance_sampling. It returns the combination of the two functions that are given.

```
    def _combine(self, mass_function, rule, normalization,
                 sample_count, importance_sampling):

        combined = self
        if isinstance(mass_function, MassFunction):
            mass_function = [mass_function]
        for m in mass_function:
            if not isinstance(m, MassFunction):
                raise TypeError("expected type MassFunction but got %s; make sure
                                to use keyword arguments for anything other than
                                mass functions" % type(m))
            if sample_count == None:
                combined = combined._combine_deterministic(m, rule)
            else:
                if importance_sampling and normalization:
                    combined = combined._combine_importance_sampling(m,
  sample_count)
                else:
                    combined = combined._combine_direct_sampling(m, rule,
                                                                 sample_count)
        if normalization:
            return combined.normalize()
        else:
            return combined
```

The method _combine_importance_sampling returns the combination if the given mass functions using importance sampling.

```
    def _combine_importance_sampling(self, mass_function, sample_count):
        combined = MassFunction()
        for (s1, n) in self.sample(sample_count, as_dict=True).items():
            weight = mass_function.pl(s1)
            for s2 in mass_function.condition(s1).sample(n):
                combined[s2] += weight
        return combined
```

The method _combine_deterministic returns the deterministic combination of the two given mass functions.

```
    def _combine_deterministic(self, mass_function, rule):
        combined = MassFunction()
        for (h1, v1) in self.items():
            for (h2, v2) in mass_function.items():
                combined[rule(h1, h2)] += v1 * v2
        return combined
```

The method _combine_direct_sampling returns the combination of the two given mass functions by using a direct sampling.

```
    def _combine_direct_sampling(self, mass_function, rule, sample_count):
        combined = MassFunction()
        samples1 = self.sample(sample_count)
        samples2 = mass_function.sample(sample_count)
        for i in range(sample_count):
            combined[rule(samples1[i], samples2[i])] += 1.0 / sample_count
        return combined
```

Below we can see the implementation of the plausibility and the belief of the DS theory. The function bel computes the belief of the given hypothesis or the entire belief function if the parameter hypothesis is set to None. The method bel. computes the plausibility of the hypothesis or the entire plausibility function. Both functions return their results in a python frozenset in order to be hashable.

```
    def bel(self, hypothesis=None):
        if hypothesis is None:
            return {h:self.bel(h) for h in powerset(self.core())}
        else:
            hypothesis = MassFunction._convert(hypothesis)
            if not hypothesis:
                return 0.0
            else:
```

```
            return fsum([v for (h, v) in self.items() if h and
                         hypothesis.issuperset(h)])



    def pl(self, hypothesis=None):
        if hypothesis is None:
            return {h:self.pl(h) for h in powerset(self.core())}
        else:
            hypothesis = MassFunction._convert(hypothesis)
            if not hypothesis:
                return 0.0
            else:
                return fsum([v for (h, v) in self.items() if hypothesis & h])
```

4. RECOMMENDATION SYSTEMS

4.1 Overview

A recommendation system (RS) is a subclass of information filtering systems that provides suggestions for items that might interest a particular user. The suggestions that an RS provides involve various derision making processes like what song to listen to, what clothes to buy or even what books to read. The item is a term that indicates what the RS recommends to users [16].

Suppose that it is a Saturday night and we look for a movie to watch, so we go to a DVD store in order to rent a movie. We want to watch a thriller, thus we find that there are 60+ movie titles with that genre in the store. Because the number of the choices is overwhelmingly large for us to decide, we ask the employee to give us some advice in which movie we should rent. The employee will ask as if we want the movie to be directed from the 2015 and after, instead we answer that we would like to watch a vintage movie. The number of the suggested movies are 5, which is a suitable number of choices in order for us to decide. Instead when we want to rent a movie from an online DVD store we will be able to add filters in order to find the movies of our selection. Thus the filter that we will use is going to be ”genre: thriller” so the online movie store again will provide us with 60+ movie titles. This time we won’t have an employee to ask us helpful questions in order to minimize our choices, which consists a big disadvantage.

Thus We use the RSs because the users don’t have the time or the experience that is need in order to fully evaluate all the potential overwhelming number of items that might be interesting for them. There might be personalized recommendations and non-personalized. An example of the first is the online store known as Amazon where the user has a personal account that uses a recommendation system in order to give fully personalized item suggestions for them. An example of the non-personalized recommendations is again at the Amazon website where we can see the top products of the day which will be the same for all the users.

4.2 Recommendation Techniques

In order for an RS to provide the recommendations of products or services, which will be tailored to the user’s preferences and constraints needs to collect information from users regarding their preferences. The user data that will be used from the RS can be explicitly expressed, as ratings for products or indirectly by interpreting the actions of the user like collecting demographic attributes [16].

Thus each recommendation technique may vary because of the type of the data. For example in an music recommendation system (MRS) we have the music metadata of a song, the audio data but we have a lack of explicitly available user feedback data so we would choose a contented based technique, that would find similar songs to those that a user might have listened in the past. Another example is the Demographic technique where we won’t need from the user to make a rating for an item, because it recommends items via the demographic data of the user. Below we will present five of the the mostly used recommendation techniques as Burke Robin [16], [18] distinguishes. In the thesis, recommendations will be carried out by reasoning under uncertainty. Towards this, Dempster-Shafer Theory will be used.

4.2.1 Content Based Systems

Content Based RS, recommend items based on the similarity, of the items that the user has liked in the past. The similarity between two or more items is based on the features that characterize them.

The classic content based recommendation (keyword-based systems) [16] tries to match the user profile attributes against the item attributes. Here the attributes will be simple keywords. For example, suppose that we have a Netflix account and we mostly like to watch crime thrillers, the suggestions that Netflix will provides us with will be crime thrillers because the similarity will be the genre of the movies.

Another content based technique will be the semantic indexing technique, where we will use concepts instead of keywords. Top-down and Bottom-up are the main group of semantic indexing [16]. The ”Top-down technique rely on the integration of external knowledge such as ontologies, encyclopedic knowledge” [16] and the ”Bottom-up uses a semantic representation based on the hypothesis that the meaning of words depends on their usage in large corpora of textual documents” [16].

4.2.2 Collaborating Filtering Systems

This approach recommends to the active user the items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users. This is the reason why we refer to collaborative filtering as *”people-to-people”* correlation. Collaborative filtering is considered to be the most popular and widely implemented technique in RS [7]. Data collection under this approach includes both explicit data collection, like asking a user to rate an item, and implicit data collection, like keeping records on how often and for how long a user views an item.

4.2.3 Demographic System

Demographic Recommender systems generate recommendations based on the user demographic attributes. It categorizes the users based on their attributes and recommends the items by utilizing their demographic data [7], [8].

4.2.4 Knowledge-based System

Knowledge-based systems recommend items based on specific domain knowledge about how certain item features meet users needs and preferences and how the item is useful for the user [7]. Knowledge based systems usually tend to work better than others at the beginning of their deployment, if they are equipped with learning components [16].

4.2.5 Hybrid recommender System

A hybrid recommender system can combine all or some of the above techniques. Thus it can use the advantages of the combined techniques and try to fix the disadvantages that arise from them.

4.3 Handling uncertainty inside a Recommendation System

All the above recommendation techniques that have been shown above aim to to enhance the accuracy and the performance of the recommendations. Nevertheless all these approaches are unable to to deal with the uncertainty that arises at different faces of the recommendation process. This is important because the uncertainty may seriously affect the RS performance.

The uncertainty arises from the randomness present in the data itself, like measure errors in the data collection. The information about the user preferences is commonly uncertain, imprecise or even incomplete [31]. Thus handling the uncertainty became an active research area and its integration in real world applications becomes a important challenge [32].

In this thesis we will use the Dempster Shafer Theory of evidence that we have already seen in the Chapter 2 in order to handle the uncertainty. This theory is one of the most known frameworks to model uncertain imprecise and incomplete information. Also is a powerful tool that combines evidence from different sources. This will be proved to be very helpful for our recommendation model. Below we will show how we can use the DS theory as a recommendation technique for the recommendation model that we will implement in this thesis.

4.4 Dempster Shafer in a Recommendation System

DS theory can be used internally in a RS as a recommendation technique in order to provide users with item suggestions. This recommendation technique could be characterized as a non-personalized recommendation as it provides the suggestion in a group of users and not to a specific user. The paper *”Discovering user preferences using Dempster-Shafer theory”* [6] suggests an effective recommendation model that uses as a technique method to use the DS theory in order to bridge the online-offline gap in recommending items. We will explain the problem of the online-offline gap in recommending items with an example.

More specifically in this technique it is proposed to move from items to item features in order to reduce the problem dimensionality and to infer the user preferences even though they are not made explicit. The preferences which are induced by each feature are considered as an independent source of information, and will be combined by a rule. With the help of the DS theory we will be able to handle the uncertainty and still be able to provide the users with item recommendations.

4.4.1 Music Recommendation System Model

In thesis we will use a similar model to the one proposed in the paper *”Discovering user preferences using Dempster-Shafer theory”* [6]. In the paper it is suggested that we use the item features instead of the item itself for the recommendation in order to reduce the problem dimensionality and to infer user preferences even though they are not made explicit. The DS theory is used and more specifically the Dempster’s combination rule is used to combine the preferences that are induced by each item feature, in order to provide item suggestions for a homogeneous group of users. In our case we will use the Dempster’s combination rule implemented by a Python library called ”py_dempster_shafer” [20]. In the library the Dempster’s combination rule that we will use is implemented with a Monte Carlo algorithm. The items in our case will be songs and their features will be some of the music metadata that we have analyzed above and the user data collection that will be used from this RS will be a list of *Like* votes from each user to the songs they prefer.

4.4.1.1 Formal Definitions

The model definitions are the following:

- the characteristics set of a song are $C = \{c_{i_1}, \ldots, cik_i\}$.
- the domain of each feature f $F_i = \{f_1, \ldots, f_p\}$ being f_i the number of different values a feature can take.
- the item set is $I = \{i_1, \ldots, i_m\}$ each I_j is defined by a vector $(f_{1j}, \ldots, f_{pj})$ with $c_{i,j} \in 2^{Ci}$.
- the user set is $U = \{u_1, \ldots, u_n\}$.

4.4.1.2 Model Constraints

The model has the following constrains:

- The users belong to the same market segment, so they are homogeneous in terms of profiling.
- Each item is described by some characteristics which can be single or multiple values.
- Items can be grouped and user preferences can be expressed on item groups.
- Values assigned to groups of items are obtained by the union of characteristic values assigned to each item.

4.4.1.3 Basic probability assignments (bpas) for feature sets

The suggestions of this MRS are made by taking into account the characteristics of the items. The task of the MRS is to search for a subset of items with maximal likelihood to be preferred by the users. As it was previously detailed the users assumed to be homogeneous this means that they belong to the same market segment. Thus their features are assumed to be similar. We will define this assumption as follows.

$$P(U_i \to I_k) = \sum_{G_h} P(U_i \to I_k | G_h \to I_k,\, U_i : G_h)\ P(G_h \to I_k | U_i : G_h) P(U_i : G_h)$$

For the above mathematical expression we need to define the following:

- The expression $U_i \to I_k$ means that the user U_i likes the item I_k.
- The $G_h \to I_k$ means that the subset G_k of users like the item I_k.
- Lastly the expression $U_i : G_h$ meaning is that the user U_i is homogeneous in terms of profiling yo the group G_h.

From all the above definitions we can conclude the following:

- The expression $P(G_h \to I_k | U_i : G_h) = P(G_h \to I_k)$ is obvious.
- Given that the preferences of a group are translated to all the users then we cam write that $P(Ui \to I_k | G_p \to I_k,\, U_i : G_p) = 1$.
- If all the users in a group are homogeneous G_p then $P(U_i : G_p) = 1$.

The search for the MRS suggestions can be limited to the voted expressed by the users to the group G_p.

$$P(U_i \to I_k) = P(G_p \to I_k)$$

Definition 4.4.1. *(Feature-basic probability assignment) Let C_i be a feature with domain $F_i = \{f_{i1}, \cdot, f_{ik_i}\}$ The basic probability assignment $m(K)$, $\forall K \in 2^{F_i}$ is defined as follows:*

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

The $U_K \subseteq U$ are the users who selected items with features c_{ij} are part of K, $\bigcup\limits_{j \in I(U_K)} c_{i,j} = K$, where $I(U_K)$ is the collection of items chosen by U_K.

Example 4.

A simple example will explain the definition of the feature-basic probability assignment 4.4.1. Let F_i be a set with song features, $F_i = \{rock, latin, indie, classic\}$, the number of users is three, #$(U) = 3$. The users that like rock and indie songs are two, #$(U_K) =$ #$U_1, U_2 = 2$. Thus, $m(\{rock, indie\}) \approx 0.667$. △

From the definition of the feature probability assignment we conclude that the number of property sets with basic probability assignment is at most the number of the users. The music genres are approximately 42 without the subcategories that they may have, thus the computation of the basic probability assignment should require exploring $2^{42} \approx 4,398 \times 10^{12}$ sets. Thus we can understand how much we can reduce the problem dimensionality.

Definition 4.4.2. *(Item projection) Given a feature C_i with domain F_i, we can write that for all $\Psi = \{I_1, \cdots, I_s\} \in 2^I$,*

$$pc_i(\Psi) = K_j \subseteq 2^{F_i},$$

where $F_{i,q} \in K_j$ $F_{i,q} \subseteq c_{i,m}$ for which $I_m = (c_{1,m}, \cdots, c_{p,m}) \forall m \in 1, \cdots, s and q \in 1, \cdots, k_i$.

Definition 4.4.3. *(Item basic probability assignment) Let $\Psi \in 2^I$ be an item set and $pc_i(\Psi) = K_j$ its projected feature set with regard to C_i. Then the basic probability assignment of $pc_i(\Phi)$ is*

$$m(\Psi) = m_i(K_j), \text{ being } pc_i(\Psi) = K_j$$

Definition 4.4.4. *(Dempster Rule of Combination) Let $\Psi \in 2^I$ be an item set and the features $\{C_1, \cdots, C_k\}$. Then the $m(\Psi)$ is defined as follows:*

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

where $Z = \sum\limits_{A_1 \cap \cdots \cap A_k = \emptyset} m_1(pc_1(A1) \cdots m_k(pc_k(A_k))$

Definition 4.4.5. *(Desirability of a set) In order to find the best item set suggestions for the users we will use the following mathematical expression.*

$$Des(X) = \sigma Pl(X) + (1 - \sigma) Bel(X) \qquad \forall \sigma \in [0, 1], \forall X \in 2^\Theta$$

The Des defines the desirability of the set and the σ defines the conservation degree where $\sigma = 0$ is full conservative.

5. MUSIC RECOMMENDATION SYSTEM

We are going to take a closer look into the music recommendation systems. More specifically we will use the Dempster-Shafer theory in order to provide the users with song recommendations.

5.1 Music Metadata

Music metadata is the information included in audio files used to identify audio content. It includes information like artist, genre, title, album name, track number etc. The more detailed the metadata the more easy is to make the music recommendations to users. Without the metadata the song would be files with no content. For major streaming platforms like Apple Music, Pandora, Spotify etc it is necessary to have strict metadata conventions.

A list with some of the music metadata is [5]:

- Track Title
- Genre
- Primary Artist/Band Name
- Featured Artists
- Composer
- Publisher
- Producers
- Explicit Content
- Lyrics Language
- Lyrics publisher
- Composition Owner
- Year of Composition
- Master Recording Owner
- Year of Recording
- Release Language

5.1.1 Example of application

In the example below we will show in detail how the Music Recommendation Model works as well as how we will find the best songs to suggest to the users.

Example 5.

Given the table 5.6 below we can see the songs that every user likes and the features of each song. With the data that the table 5.1 provides us with, we can begin with the steps of the model.

Firstly we need to find the basic probability assignment for each feature of the songs. The features are genre, artist name and song title. Thus we will make 3 different tables that associate the bpa with each feature. The table 5.2 shows the bpa of the genre feature, the table 5.3 shows the bpa of the artist name and finally the table 5.4 shows the bpa of *the song title.*

Table 5.1: Song metadata and user song likes

Song	Title	Genre	Artist Name	U_1	U_2	U_3	U_4
1	Back to Black	Soul, Pop	Amy Winehouse		✓		
2	Bohemian Rhapsody	Rock	Queen	✓		✓	
3	Don’t play that song	Soul	Aretha Franklin		✓		
4	Burning For You	Rock	Blue Öyster Cult	✓			
5	Fly me to the moon	Jazz	Frank Sinatra		✓	✓	✓
6	Dream On	Rock, Pop	Aerosmith				
7	(Don’t Fear) The Reaper	Rock	Blue Öyster Cult	✓			

Table 5.2: Basic probability assignment associated with the song genre

Genre	Users	bpa
Jazz	U_4	0.25
Rock	U_1	0.25
Rock, Jazz	U_3	0.25
Soul, Pop, Jazz	U_2	0.25

Table 5.3: Basic probability assignment associated with the song artist name

Artist Name	Users	bpa
Amy Winehouse, Frank Sinatra, Aretha Franklin	U_2	0.25
Blue $\ddot{O}$yster Cult, Queen	U_1	0.25
Frank Sinatra, Queen	U_3	0.25
Frank Sinatra	U_4	0.25

Table 5.4: Basic probability assignment associated with the song title

Title	Users	bpa
Fly me to the moon	U_4	0.25
Back to Black, Don’t play that song, Fly me to the moon	U_2	0.25
Bohemian Rhapsody, Burning For You	U_1	0.25
Fly me to the moon, Bohemian Rhapsody	U_3	0.25

We have computed the bpas that are associated with the features (genre, title, artist name) and we have kept those that have bpa greater than zero. Now the next step is to find the association between the items and the bpas. In order to achive that we will use the tables with the feature-bpa that we have already made and we will associate each feature with the items that contain it.

Table 5.5: Basic probability assignment associated with the item

Genre	**Songs**	**bpa**
Jazz	5	0.25
Rock	$2,4,6,7$	0.25
Rock, Jazz	$2,4,5,6,7$	0.25
Soul, Pop, Jazz	$1,2,3,4,5,6,7$	0.25
Artist Name		
Amy Winehouse, Frank Sinatra, Aretha Franklin	$1,3,4$	0.25
Blue $\ddot{O}$yster Cult, Queen	$2,4,7$	0.25
Frank Sinatra, Queen	$2,5$	0.25
Frank Sinatra	5	0.25
Title		
Fly me to the moon	5	0.25
Back to Black, Don’t play that song, Fly me to the moon	$1,3$	0.25
Bohemian Rhapsody, Burning For You	$2,4$	0.25
Fly me to the moon, Bohemian Rhapsody	$2,5$	0.25

In the table 5.5 we have made the association between the items-features-bpa. Again we will keep only those whose bpa is greater than zero. Now that we have made the above association we will use the Dempster rule of combination in order to fuse two feature tables at a time. Here we will compute the combination between the genre and the artist name. The table 5.6 depicts the combination between the feature Genre and Artist Name.

$m_{genre}(\{5\}) = 0.25$
$m_{genre}(\{2,4,6,7\}) = 0.25$
$m_{genre}(\{2,4,5,6,7\}) = 0.25$
$m_{genre}(\{1,2,3,4,5,6,7\}) = 0.25$

$m_{artist}(\{1,3,4\}) = 0.25$
$m_{artist}(\{2,3,7\}) = 0.25$
$m_{artist}(\{2,5\}) = 0.25$
$m_{artist}(\{5\}) = 0.25$

Table 5.6: Dempster rule of combination at the items associated with genre and the items associated with the artist name

Genre \ Artist	{1,3,4}	{2,4,7}	{2,5}	{5}
{5}	$\emptyset$ 0.0625	$\emptyset$ 0.0625	$\{5\}$ 0.0625	$\{5\}$ 0.0625
{2,4,6,7}	$\{4\}$ 0.0625	$\{4,7\}$ 0.0625	$\{2\}$ 0.0625	$\emptyset$ 0.0625
{2,4,5,6,7}	$\{4\}$ 0.0625	$\{2,4,7\}$ 0.0625	$\{2,5\}$ 0.0625	$\{5\}$ 0.0625
{1,2,3,4,5,6,7}	$\{1,3,4\}$ 0.0625	$\{2,4,7\}$ 0.0625	$\{2,5\}$ 0.0625	$\{5\}$ 0.0625

As we have shown in the Chapter 2 we can compute the new mass functions.

$m_{g,a}(\{5\}) = 0.3076$

$m_{g,a}(\{2\}) = 0.0769$

$m_{g,a}(\{4\}) = 0.1538$

$m_{g,a}(\{2,5\}) = 0.1538$

$m_{g,a}(\{1,3,4\}) = 0.0769$

$m_{g,a}(\{4,7\}) = 0.0769$

$m_{g,a}(\{2,4,7\}) = 0.1538$

Table 5.7: Dempster rule of combination at the items associated with genre and the items associated with the artist name

Title Genre,Artist	{5}	{1,3}	{2,4}	{2,5}
{5}	$\{5\}$ 0.0769	$\emptyset$ 0.0769	$\emptyset$ 0.0769	$\{5\}$ 0.0769
{2}	$\emptyset$ 0.0192	$\emptyset$ 0.0192	$\{2\}$ 0.0192	$\{2\}$ 0.0192
{4}	$\emptyset$ 0.0384	$\emptyset$ 0.0384	$\{4\}$ 0.0384	$\emptyset$ 0.0384
{2,5}	$\{5\}$ 0.0384	$\emptyset$ 0.0384	$\{2\}$ 0.0384	$\{2,5\}$ 0.0384
{4,7}	$\emptyset$ 0.0192	$\emptyset$ 0.0192	$\{4\}$ 0.0192	$\emptyset$ 0.0192
{1,3,4}	$\emptyset$ 0.0192	$\{1,2\}$ 0.0192	$\{4\}$ 0.0192	$\emptyset$ 0.0192
{2,4,7}	$\emptyset$ 0.0384	$\emptyset$ 0.0384	$\{2,4\}$ 0.0384	$\emptyset$ 0.0384

Table 5.8: The new masses after the Dempster rule of combination

Mass function	Belief	Plausibility
$m_{g,a,t}(\{5\}) = 0.4338$	$Bel(\{5\}) = 0.4338$	$Pl(\{5\}) = 0.5204$
$m_{g,a,t}(\{2\}) = 0.1733$	$Bel(\{2\}) = 0.1733$	$Pl(\{2\}) = 0.3898$
$m_{g,a,t}(\{4\}) = 0.1733$	$Bel(\{4\}) = 0.1733$	$Pl(\{4\}) = 0.2599$
$m_{g,a,t}(\{2,5\}) = 0.0866$	$Bel(\{2,5\}) = 0.6937$	$Pl(\{2,5\}) = 0.8236$
$m_{g,a,t}(\{1,2\}) = 0.0433$	$Bel(\{1,2\}) = 0.2166$	$Pl(\{1,2\}) = 0.3898$
$m_{g,a,t}(\{2,4\}) = 0.0866$	$Bel(\{2,4\}) = 0.4332$	$Pl(\{2,4\}) = 0.5631$

With the Dempster rule of combination between the three features we have computed the new masses. Now we can find the belief and the plausibility of each probability as we can see in the table 5.8. The final step is to select the preferred items for the suggestions for the users. This will happen by using the desirability definition 4.4.5 with $\sigma = 0.6$ $Des(X) = 0.6Pl(X) + 0.4 \cdot Bel(X)$.

Table 5.9: The desirability score for each song set

Songs	Desirability Score
$\{5\}$	$Des(\{5\}) = 0.4854$
$\{2\}$	$Des(\{2\}) = 0.3031$
$\{4\}$	$Des(\{4\}) = 0.2253$
$\{2,5\}$	$Des(\{2,5\}) = 0.7715$
$\{1,2\}$	$Des(\{1,2\}) = 0.3204$
$\{2,4\}$	$Des(\{2,4\}) = 0.5111$

Suppose that we need all the item sets that their desirability is greater than 0.5 *then the songs that the MRS is going to suggest will be the* $\{2:$ *Bohemian Rhapsody*, $4:$ *Burning for You*, $5:$ *Fly me to the moon*$\}$.

△

5.1.2 Implementation of the model

In this chapter we present the implementation of the model in that we made 4.4.1 and which tools have been used. We will describe the input data, how we manipulate it in order to be able to use it for the MRS model purposes. Furthermore we will describe in detail the application code.

5.1.2.1 Tools

The Music Recommendation System model that we have described in this chapter 5 has been implemented with the Python programming language. Other tools that have been used are the SQlite in order to handle the input data, the pandas which is a python library that we use in order to manipulate csv files and the python library py_dempster_shafer that is implemented by Thomas Reineking [20] and we will use the implementation of the Dempster rule of combination with the use of Monte Carlo algorithm.

5.1.2.2 Input data

The input data that we are going to use are provided by the MillionSongDataset (MSD) [17] and the Thisismyjam [18]. From the first we have been provided with one SQLite database which is called **track_metadata.db**. This database contains the songs and their metadata. The second provides us with two files which contain the data about the users and their song choices. These files are the **likes.tsv** which contains the user and their jam choices and the **jam_to_msd.csv** which associates the jams with the songs.

Below in the table 5.10 we can see the form of the track_metadata that we need. The original form contains some extra metadata but they will not be useful for the model. Thus we will not use them and instead we will form a table with only the necessary features.

Table 5.10: An example of the track_metadata table

track_id	title	release	artist_name	duration	year
TR···	Silent Night	Monster Ballads X-Mas	Faster Pussy cat	252.05506	2003

Below we show the input data that the Thisismyjam has provided. The table 5.11 depicts the relationship between a user and a jam. As said before the jam is used to connect the user with the MillionSongDataset songs. Thus we could describe the two following tables as intermediate tables between the user and the song metadata.

The likes.tsv is a file which is separated with tabs thus the tsv ending of the file. Each record that contains is a user and a jam_id. Thus if a user has liked many songs there will be many records with the same user and a different jam_id. This constitutes a problem because we need only one record that associates a user with many jam_ids. Before we solve this problem we will firstly associate the user with the msd_id. So we construct a new table called user_msd 5.13.

Table 5.11: The likes.tsv file

user_id	jam_id
u0	1211
u0	1212
u1	1212

Table 5.12: The jam_to_msd.csv file

jam_id	msd_id
1211	a67
1212	a57

Table 5.13: The table user_msd

user_id	msd_id
u0	a67
u0	a57
u1	a57

Now we will solve the problem that we described above. From the table 5.13 we will construct a new table that connects a user with all their msd_ids. The msd_ids will be separated with the character ”|”. An example of this table is depicted below.

Table 5.14: The likes table

user_id	msd_id
u0	a67|a57
u1	a57

In the figure 5.1 all the tables that have been shown in this chapter are depicted in a diagram. As the diagram depicts, the data that we will use in the model application, are stored in from the files song.csv and likes.csv. If we combine the likes file and the songs file we would have as a result a table like the 5.1.

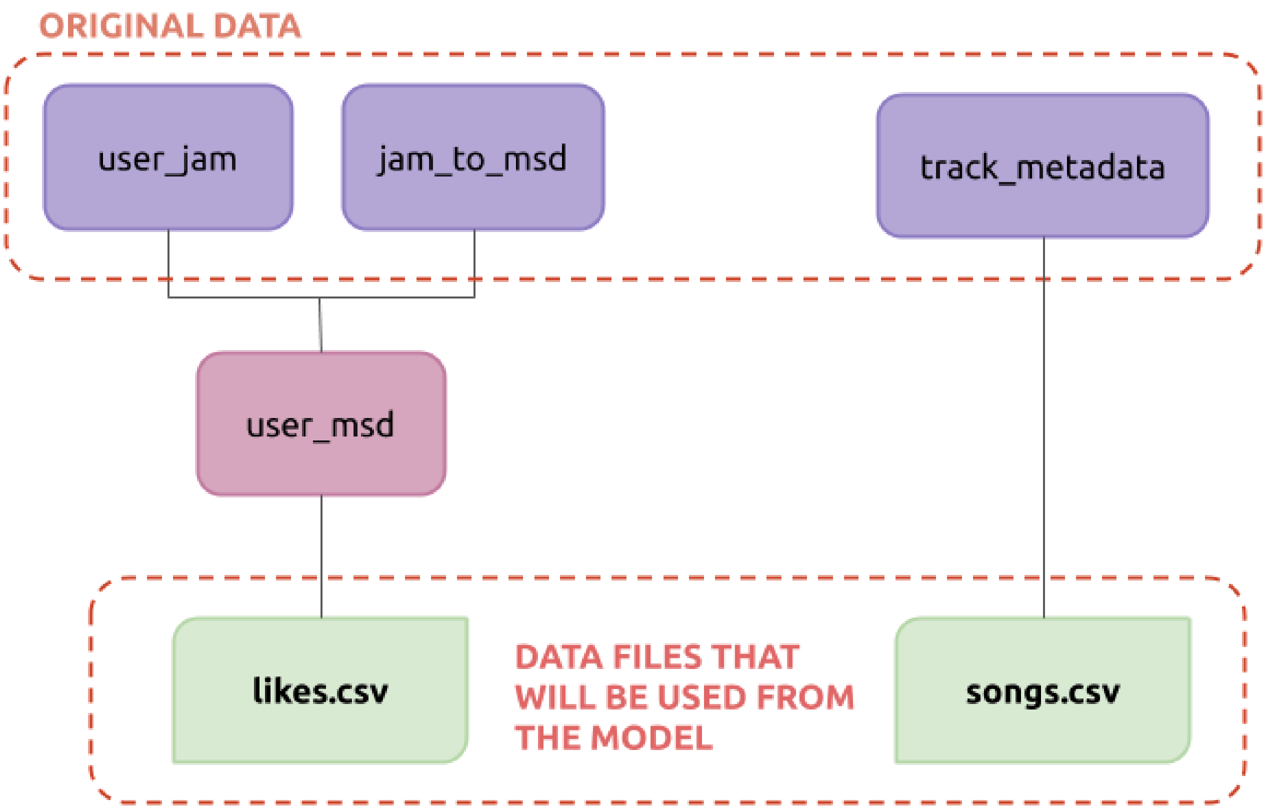

Figure 5.1: Input data diagram

5.1.2.3 Model application

After generating the two input csv files songs and likes, we can continue with the application of the model. The steps that are implemented in the Python code are the same as are shown in the section 5.1.1. Here we will show how the model application works.

The first step of the application is to read the two files songs.csv, likes.csv and store their data in two tables. After storing the files we will use them in order to make a dictionary that contains the item sets that the users prefer along with their bpas. After constructing this dictionary we need to connect the items with their features. Thus we will construct new dictionaries, one for each song characteristic, that will contain the features with their bpa.

Next we will find the mass function for each characteristic, by using the corresponding dictionary. The next step is to use the implementation of Dempster's rule of combination with the Monte Carlo algorithm and fuse all the mass functions until we are left with one final mass function. We will do the same for the Dempster's implementation.

The last step of the application is to select the most desired songs that the recommendation system will suggest to the users along with their scores and the time needed for their finding. This will be implemented by a simple yet effective method. For this method we will choose a desirability threshold and then search the song set. If their desirability score is greater than the threshold they will be suggested. This step will be done for both implementations (Monte Carlo and Exact computation) in order to be able to compare their results.

The application steps are shown in the figure below 5.2.

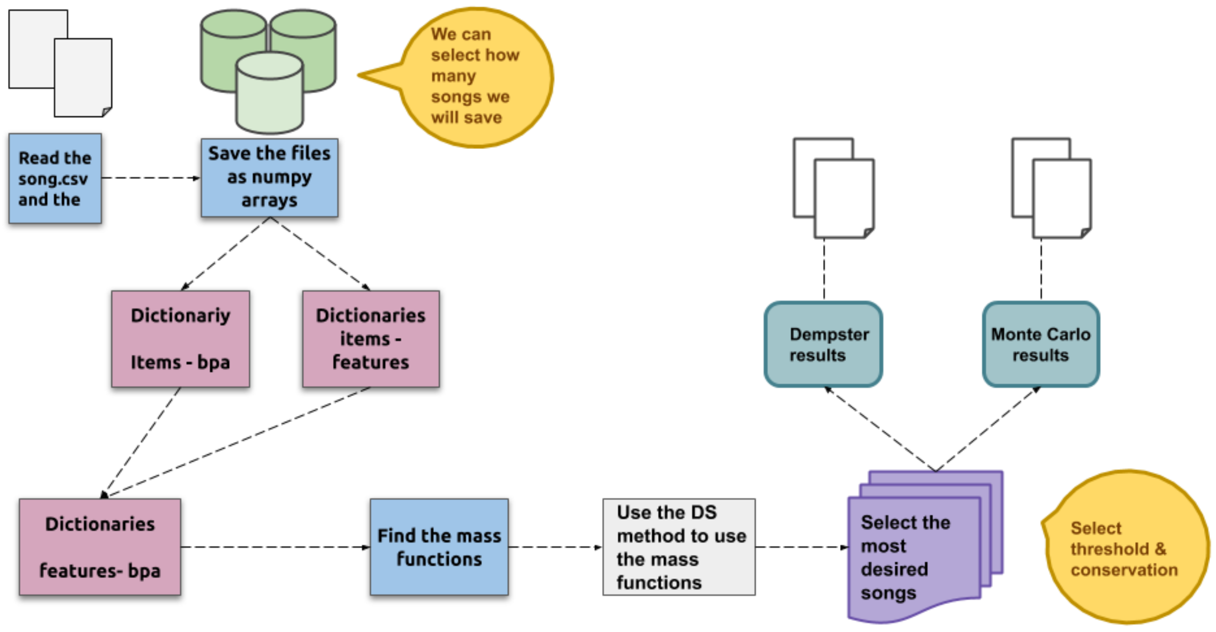
We can select how many songs we will save
Read the song.csv and the
Save the files as numpy arrays
Dictionariy Items - bpa
Dictionaries items - features
Dictionaries features- bpa
Find the mass functions
Use the DS method to use the mass functions
Select the most desired songs
Select threshold & conservation
Dempster results
Monte Carlo results

Figure 5.2: Application of the model

5.1.3 Problems with the implementation of the model

In the implementation of the model the most serious problem that we faced was the space complexity of the model. As we mentioned before this model constructs python dictionaries in order to construct the mass functions that both Dempster's rule of combinations implementations will use. Thus as the song number is growing the need for memory grows alongside. In order to avoid this problem we tried to use as less space for the dictionaries.

This means that if the data of a dictionary were no longer useful we would immediately delete the dictionary or even change some of the dictionaries data with new instead of using two dictionaries in the same time.

Another problem that we faced was that for a large song number we faced an underflow in the desirability calculation. Thus we used mathematical calculation of the numpy library of the Python in order to avoid the desirability results with underflow.

5.1.4 Tests and Results

We will test the two implementations of the Dempster's rule of combination. We will test for a certain number of records their desirability scores along with the conservation degree as well as the time results again along with the conservation degree.

For the results in the tests below we will assume that the desirability threshold is 0.5.

Table 5.15: Results of the Dempster's and the Monte Carlo implementations with 100 song records

Results of 100 Records						
	Exact method			Monte Carlo		
conservation degree	Best Score	Number of Songs	Av. Score	Best Score	Number of Songs	Av. Score
0.3	1.0	2	0.9999	1.0	1	1.0
0.5	1.0	2	0.9999	1.0	1	1.0
0.7	1.0	2	0.9999	1.0	1	1.0

Table 5.16: Results of the Dempster's and the Monte Carlo implementations with 500 song records

Results of 500 Records						
	Exact method			Monte Carlo		
conservation degree	Best Score	Number of Songs	Av. Score	Best Score	Number of Songs	Av. Score
0.3	0.9992	12	0.9979	0.9975	1	0.9975
0.5	0.9992	12	0.9979	0.9975	1	0.9975
0.7	0.9992	12	0.9979	0.9975	1	0.9975

Table 5.17: Results of the Dempster's and the Monte Carlo implementations with 1000 song records

Results of 1000 Records						
	Exact method			Monte Carlo		
conservation degree	Best Score	Number of Songs	Av. Score	Best Score	Number of Songs	Av. Score
0.3	0.9999	45	0.9989	0.9981	1	0.9981
0.5	0.9999	45	0.9989	0.9981	1	0.9981
0.7	0.9999	45	0.9989	0.9981	1	0.9981

Table 5.18: Results of the Dempster's and the Monte Carlo implementations with 5000 song records

Results of 5000 Records						
	Exact method			Monte Carlo		
conservation degree	Best Score	Number of Songs	Av. Score	Best Score	Number of Songs	Av. Score
0.3	0.9997	2189	0.9283	0.8967	1	0.8967
0.5	0.9997	2189	0.9284	0.8967	1	0.8967
0.7	0.9997	2189	0.9284	0.8967	1	0.8967

From the above results we can conclude that the Monte Carlo algorithm implementation scores are lower than the Exact method. Also we can see that the latter wields more song options than the Monte Carlo. But in the table 5.18 the number of the most desired songs of the Exact's method is very large for the RS to recommend to the users. We could select only a number of the best songs but as we see in the 5.18 the song number is greater than 2.000.

We also need to examine the time that both methods needed to compute those results. From the figure below we can see that the Monte Carlo needs lesser time to find the best results than the Dempster's method as the song number is growing.

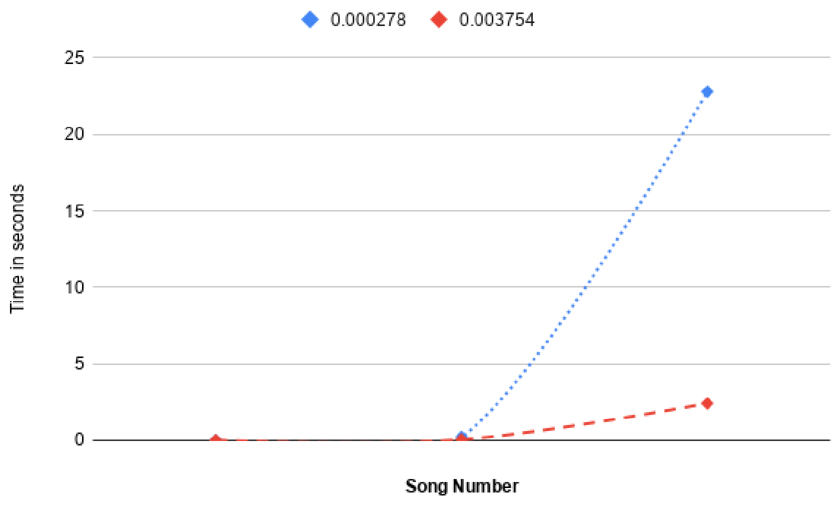

Figure 5.3: Time chart

Below we can see how much time is needed for each algorithm to find the best results if we use different conservation degrees.

Table 5.19: Time and conservation degree for 100 songs

Time needed for 100 Records		
	Exact method	Monte Carlo
conservation degree	Time (seconds)	Time (seconds)
0.3	0.0001684	0.0035307
0.5	0.0002501	0.0039434
0.7	0.0002784	0.0037543

Table 5.20: Time and conservation degree for 500 songs

Time needed for 500 Records		
	Exact method	Monte Carlo
conservation degree	Time (seconds)	Time (seconds)
0.3	0.0120897	0.0123480
0.5	0.0086407	0.0173909
0.7	0.0105557	0.0146330

Table 5.21: Time and conservation degree for 1000 songs

Time needed for 1000 Records		
	Exact method	Monte Carlo
conservation degree	Time (seconds)	Time (seconds)
0.3	0.0778903	0.0501570
0.5	0.0776214	0.0774978
0.7	0.2225786	0.0553936

Table 5.22: Time and conservation degree for 5000 songs

Time needed for 5000 Records		
	Exact method	Monte Carlo
conservation degree	Time (seconds)	Time (seconds)
0.3	22.897744	2.3440137
0.5	23.083891	2.8748354
0.7	22.796713	2.4226841

6. CONCLUSION AND FUTURE WORK

By now it is clear how helpful a Recommendation System can be in the day to day applications, as the users can find the best suited items for them. A main problem that the RS need to solve, in order to make the best recommendations, is the uncertainty. The uncertainty arises from the data that is collected from the users. In the Music RS that we have implemented in this thesis we have used the Dempster Shafer Theory of evidence as it is one of the most expressive frameworks for reasoning with uncertainty.

In our Music RS model we make recommendations by using the items features and not the items. Thus the DS Theory does not only helps us handle the uncertainty but is also very helpful as it can combine evidence from different sources, like the features of each song and arrive at a degree of belief.

The Music RS uses data from real world applications such as the Million Song Dataset [16]. This is one of the reasons that we have chosen to use an optimized algorithm called Monte Carlo algorithm [20] in order to implement the DS Theory. We have compared this Monte Carlo algorithm and a simple implementation of the Dempster rule of combination in order to see the desirability of the results and the time needed in order to produce those results.

It is a question of future work to further investigate the capabilities of the approximation algorithms that implement the DS Theory. More examinations based on the desirability of the results, the time and space complexity could be carried out with the same data. Thus those data could help us to fully understand which algorithm is best suited for this type of RS model.

ABBREVIATIONS - ACRONYMS

DS	Dempster-Shafer
bpa	basic probability assignment
MSD	Million Song Dataset
MRS	Music Recommendation System

ANNEX I

The following code blocks include some of the most important functions of this thesis Music RS Model. This Model is implemented with the Python programming language.

The function find_feature_set makes a set with the users and their song choices by using the user likes data and the song data.

```
def find_feature_set(user_likes, song_data):
  feature_set = set()

  for row_number in user_likes:
    [multi_feature] = song_data[int(row_number) - 1]
    # remove the spaces from the features strings
    multi_feature = multi_feature.replace(" ", "")
    for feature in split_string(multi_feature, "|"):
      # we won't take in consideration the None in the features
      if feature == 'None':
        continue
      feature_set.add(feature)

  return feature_set
```

The function find_feature_set_bpa_dictionary returns a dictionary with the feature sets and their bpa.

```
def find_feature_set_bpa_dictionary(likes_data,
        feature_data, user_number):

  feature_dict = dict()

  for row in likes_data:
    user_likes = split_string(row[1], "|")
    feature_set = frozenset(find_feature_set(user_likes, feature_data))
    if feature_set in feature_dict:
      feature_dict[feature_set] += 1
  else:
    feature_dict[feature_set] = 1

  for key in feature_dict:
    feature_dict[key] /= user_number
  return feature_dict
```

The function feature_item_dictionary returns a dictionary with feature as keys and the items that is related as values.

```
def feature_item_dictionary(features_array):
  feature_item_dict = dict()

  row_num = 0

  for [row] in features_array:
    if row is None:
      continue
    row_num += 1
    features = split_string(row, "|")
    for feature in features:
      feature = feature.replace(" ", "")
      if feature == "None":
        continue
      if feature in feature_item_dict:
        feature_item_dict[feature].add(row_num)
      else:
        feature_item_dict[feature] = {row_num}

  return feature_item_dict
```

The function feature_item_dictionary takes as arguments two dictionaries. The first one contains features sets and the items that contain them and the second contains the feature sets and their bpa.

```
def from_feature_to_item_dictionary(feature_user_dict,
                                    feature_item_dict):

  dictionary = dict()

  for feature_set in list(feature_user_dict):
    item_set = set()
    for feature in feature_set:
      if feature == "None":
        continue
      items = feature_item_dict[feature]
      item_set = item_set.union(items)
    dictionary[frozenset(item_set)] = feature_user_dict[feature_set]

  return dictionary
```

The function return_all_item_set_bpa_dicts returns an array with all the item set and bpa dictionaries.

```
def return_all_item_set_bpa_dicts(likes_data, song_data):
      user_number = likes_data.shape[0]
      col_song_data = song_data.T.shape[0]

      return_array = []

      for feature_col in range(1, col_song_data):
        feature_user_dict = find_feature_set_bpa_dictionary(
          likes_data, song_data[:, [feature_col]], user_number
        )

      feature_item_dict = feature_item_dictionary(song_data[:, [feature_col]])

      item_bpa_dict = from_feature_to_item_dictionary(
        feature_user_dict, feature_item_dict
      )

      return_array.append(item_bpa_dict)

      return return_array
```

The function mass_functions_from_dictionaries returns the mass functions from the dictionaries that contain the item sets and their bpa's.

```
    def mass_functions_from_dictionaries(dictionary_array):
        mass_function_array = []

        for dictionary in dictionary_array:
          mass_function_array.append(MassFunction(dictionary))

        return mass_function_array
```

The function ds_mc_combination_rule takes as input the mass function array and returns the mass function after using the Dempster Shafer combination rule via using a monte carlo algorithm.

```
  def ds_mc_combination_rule(mass_function_array):
    monte_carlo = mass_function_array[0] & mass_function_array[1]

    for mass_function in mass_function_array[2:]:
      monte_carlo = monte_carlo.combine_conjunctive(
        mass_function, sample_count=1000, importance_sampling=True
      )

    return monte_carlo
```

The function find_desirable_sets searches all the item sets and returns the best item suggestions.

```
  def find_desirable_sets(mass_functions, threshold):
    item_set = set()

    for key in mass_functions:
      if DempsterShafer.set_desirability(mass_functions, key, threshold):
        item_set = item_set.union(key)

    return item_set
```

The function set_desirability returns true if the desirability of an item set is greater than a threshold that we have selected.

```
def set_desirability(mass_function, item_set,
                threshold=0.5, conservation_degree=0.5):

  plausibility = mass_function.pl(item_set)
  belief = mass_function.bel(item_set)

  desirability = ( conservation_degree * plausibility +
              (1 - conservation_degree) * belief)

  if desirability > threshold:
    return True
  else:
    return False
```

REFERENCES

[1] G. Shafer, *”Dempster-Shafer theory”*, http://www.glennshafer.com/assets/downloads/articles/article48.pdf, [Accessed July 7 2020]

[2] A.P. Dempster , *”Upper and Lower Probabilities Induced by a Multivalued Mapping”*, The Annals of Mathematical Statistics, vol. 38, no. 2, pp. 325-339, 1967

[3] G. Shafer, *”A Mathematical Theory of Evidence”*, Princeton University Press, 1976

[4] T. Reineking, *”Belief Functions: Theory and Algorithms”*, doctoral dissertation, University of Bremen, February 2014

[5] LANDR, *”Metadata for Musicians: What It Is and Why It’s Vital”*, https://blog.landr.com/music-metadata/, [Accessed July 7 2020]

[6] L. Troiano, L.J. Rodríquez-Muñiz, I. Díaz, *”Discovering user preferences using Dempster-Shafer theory”*, Fuzzy Sets and Systems, vol. 278, pp. 98-117, 12 June 2015

[7] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor *”Recommender Systems Handbook”*, Springer, pp. 1-29, May 2010

[8] M. Sridevi, R.Rajeswara Rao, *”DECORS: A Simple and Efficient Demographic Collaborative Recommender System for Movie Recommendation”*, Advances in Computational Sciences and Technology, vol. 10, no. 7, pp. 1969-1979

[9] A. Kaltsounidis, *”Dempster-Shafer Theory Computation using Constraint Programming”*, BSc thesis, October 2019

[10] R.R. Yager, L.Liu, *”Classic Works of the Dempster-Shafer Theory of Belief Functions”*, Studies in Fuzziness and Soft Computing, Springer, vol. 219, pp. 1-5, 293

[11] M. Daniel, B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh, *”Algebraic structures related to Dempster-Shafer theory”*, International CoÇınlarnference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Advances in Intelligent Computing — IPMU ’94, pp 51-61, 1994

[12] L.A. Zadeh, *”A Simple View of the Dempster-Shafer Theory of Evidence and its Implication for the Rule of Combination”*, AI Magazine, vol. 7, no. 2, 1986

[13] Y. Tian, C. Stewart, *”History of E-Commerce”*, pp. 1-2

[14] G. Linden, B. Smith, J. York, *”Amazon.com recommendations: item-to-item collaborative filtering”*, IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003

[15] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, R. Zadeh *”WTF: The Who to Follow Service at Twitter”*, Twitter Inc

[16] F. Ricci, L. Rokach, B. Shapira *”Recommender Systems Handbook”*, pp. 37-76, 453-492

[17] *”Million Song Dataset”*, http://millionsongdataset.com/, [Accessed July 7 2020]

[18] *”This is my jam”*, https://www.thisismyjam.com/, [Accessed July 7 2020]

[19] R. Burke, *”Hybrid web recommender systems”*, The Adaptive Web, pp. 377–408, 2007

[20] T. Reineking, *"py_dempster_shafer (pds)"*, https://pypi.org/project/py_dempster_shafer/#description, https://github.com/reineking/pyds, [Accessed July 7 2020]

[21] M. Bauer, *"Approximation Algorithms and Decision Making in the Dempster-Shafer Theory of Evidence–An Empirical Study"*, International Journal of Approximate Reasoning, vol. 17, pp. 217-237, August–October 1997

[22] P. Orponen, *"Dempster's Rule of Combination is #P-complete"*, Artificial Intelligence, vol. 44, pp. 245-253, July 1990

[23] J.A. Barret, *"Computattional methods for a mathematical theory of evidence"*, Proceedings IJCAI-81, pp. 868-875, Vancouver BC, 1981

[24] N. Wilson, S.ín Moral, *"Fast Markov Chain Algorithms for Calculating Dempster-Shafer Belief"*

[25] N. Wilson, *"A Monte-Carlo Algorithm for Dempster-Shafer Belief"*, Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, 1991

[26] F. Voorbraak, *"A computationally efficient approximation of Dempster-Shafer theory"*, Internat. J. Man-Machine Stud., vol. 30, pp. 525-536, 1989

[27] E. Çınlar, *"Probability and Stochastics"*, pp. 51, 2011

[28] B. Tessem, *"Approximations for efficient computation in the theory of evidence"*, Artificial Intelligence, vol. 61, pp. 315-329, 1993

[29] P. Smets, *"Belief functions versus probability functions"*, Uncertainty and Intelligent Systems, pp. 17-24, July 1988

[30] N. Wilson, *"Algorithms for Dempster-Shafer Theory"*, October 27, 1999

[31] V. DoanNguyen, V. NamHuynh, *"Two-probabilities focused combination in recommender systems"*, 11 April, 1999

[32] R. Abdelkhalek, *"Handling Uncertainty in Recommender Systems under the Belief Function Theory"*, LADOREC, Institute Supérieur de Gestion de Tunis