Ecole Polytechnique Federale de Lausanne

MasterThesis

# Thesis Title

*Author:*
Khalil Hajji

*Supervisor:*
Dr. Fabien Cardinaux

*A thesis submitted in fulfilment of the requirements*
*for the degree of Master of Science in Communication Systems*

*in the*

Sony Technology Center
Communication Systems Faculty

August 2015

EPFL

# *Abstract*

Faculty Name
Communication Systems Faculty

Master of Science in Communication Systems

**Thesis Title**

by Khalil Hajji

We describe *Dirichlet Latent Multimodal Representation* ($DLMR$), a generative probabilistic model for multimodal data. Multimodal data is a dataset that stores elements of different types. $DLMR$ models the heterogeneity of this data by representing each type in a separate distribution and then by combining the different types to extract the common meaning transported by these features. We use $DLMR$ to extract users' behaviors from their smartphone logs. We compare $DLMR$ to multiple other methods (including *Latent Dirichlet Allocation* ($LDA$) and *Linearly Constrained Bayesian Matrix Factorization* ($LCBMF$)) and show that it performs better than these models.

# *Acknowledgements*

I would like to thank my supervisor Dr. Fabien Cardinaux, who supported me along this work. I would like to thank him for his availability, his open mindset and the all good suggestions and ideas he provided to me.
I would like to thank Dr. Stephen Uhlich, who showed interest to my work and was always available to provide hints and critics when needed.
I would like to thank all Sony engineers of the Speech and Sound Group for the good working atmosphere they created and for the great confidence they show to interns.
Finally, I would like to thank Sony for having allowed me to achieve my master thesis in their labs, providing me their resources and their data.

I would like to express my sincere gratitude to my academic supervisor, Prof. Patrick Thiran for his recurrent support and his relevant feedback.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Today, data comes from everywhere and in any kind. It streams for daily life; from computers, credit cards, TVs, cars, sports shoes and watch. The availability of this data is changing the way companies lead their business and are changing the rules of competitiveness. In a publication that zooms in the challenges and the power of big data, Mc Kinsey affirms that "the use and the understanding of the data will become a key basis of competition and growth for individual firms" and they estimate that a retailer exploiting his data to the full has the potential to increase its operating margin by more than 60 percent.

In this work, we are interested in the data collected from a gadget that shares the life of the user; his smartphone. Smartphone data contains the locations a user visits, the activities he does, the notification he receives, the applications he launches and many other information. This kind of data is unique in the sense that it represents a complete snapshot of a user's life. In a world driven by the power of data and the ability to anticipate and understand the needs of users, companies shows a big interest in studying this emerging kind of dataset. In the other hand, the richness and the diversity of this data attracts the curiosity of researchers.

In this context, many work have been done with this kind of logs, many paths have been explored and multiple questions answered. Those researches are discussed later with more details. In this work, we tackle an important question that escaped to the interest of previous researches: Having the smartphone logs of one user, can we find a model that exhibits his particular behaviors and habits? More practically, let's imagine the following example. Let's imagine that Bob has some particular habits: he does running while listening music on Saturdays, he visits his parents on Sundays, he reads news when he is in his office in the mornings, and he puts an alarm clock at 7am during his working days. The question we are answering in this work is the following: Can we

find a model that discovers those particular behaviors by analyzing Bob smartphone's logs? How precise can this model be in doing this task? It is important to note that we are interested in discovering the individual behaviors of a user. Moreover, we choose to build a model that preserves the privacy of the user. For this reason, we aim to find a method that takes as input the logs of a unique user so that it can be applied to a data that never exits his phone.

In the recent few years, key innovations allowed smartphones to drastically evolve from cell-phone devices used for calling to powerful "pocket-computers" devices that can be used as cell-phone, camera, calendar, clock, game consol, web browser and many other roles at the same time. As an important company in the smartphone industry, Sony is one of the actors of the smartphones evolution and is aiming on keeping innovating this sector.
Our work is a part of this continuous research for innovation. Indeed, it aims to allow smartphones building a personal relationship with their owner by adapting to their specific needs and answering heir specific requests. Let's keep the parallel with Bob to understand what does building a personal relationship with a user concretely means. Let's suppose that Bob's smartphone is able to learn the specific habits of Bob. When Bob forgets to put his alarm clock on a working day, his smartphone can remind him to do it. When an unusual traffic congestion appears in Sunday in the rode that Bob use to take to reach his parents place, his smartphone can inform him. Finally, when his smartphone does not have enough power to play music on Saturday morning, Bob's smartphone can remind him to recharge it because he will probably need it for running. Our work is a start in making it possible for a smartphone to adapt to it's owner's behaviors and to react interactively in some contexts. In other words, it is a start is making smartphones behaves smarter.

Scientifically speaking, this problem can be seen as a clustering problem: our goal is to find different clusters of data points where each cluster represents a particular behavior of the user. Coming back to Bob, running while listening to music on Saturdays morning can be represented by a cluster that contains the running activity, the application launch music, the day Saturday and the time frame 8am-12am. Putting an alarm clock at 7 pm during the working days can be represented as a cluster containing all the days of the week, the notification alarm and the time frame 7pm-8am.
Clustering is a widely addressed problem in machine learning and data analysis, and it has been applied to many contexts and topics. It as been used for example in corpus-text modeling, recommender systems and image recognition. Different approaches have been developed to answer those challenges; the probabilistic latent topic modeling and the matrix factorization are examples of these different approaches. A parallel between our methods and these approaches is made multiple times in this thesis and it will be

shown that it is of a strong benefit.
Our problem sits at the interface between an emerging area of research that takes profit of the existence of a new kind of data and an area that constitutes one of the basis of the emergence of the machine learning and data analysis techniques. From a scientifically point fo view, addressing the clustering problem in a new emerging context makes our problem particularly challenging.

The thesis is organized as follows: In chapter 2, we introduce some notations and definitions, state the problem in a mathematical way and go through the researches done in this field.
In chapter 3, we describe in details the Generative Hidden Class Model for Mixed Data Types (GHCM-MDT). It is model that we developed specifically to answer our needs and that shows to perform better than the other existing methods.
In chapter 4, we introduce other known and widely used models that has been shown to perform very well in doing tasks similar to ours. We use those models as baselines to evaluate the performances of GHCM-MDT. To have a complete overview, we both use some models based on the matrix factorization approach using some sophisticated techniques and others based on advanced methods of probabilistic latent topic modeling.
In chapter 5, we detail the metrics used to test the performance of the different models in performing the task needed.
In chapter 6, we present the results obtained with the different models and compare the performances of the different models to GHCM-MDT.
Finally, chapter 7 presents our conclusions.

# Chapter 2

# Preliminaries

## 2.1 Definitions and notations

In this work, we take an interest on datasets containing smartphone log of a use. We refer to those datasets as *data*, *smartphonelogs*, *userlogs* or *dataset*.

The particularity of those logs is that they contain different information sources and types. They contain for example information about the location of the user, the activities he is doing, the applications he uses, the settings he puts in the phone, the notifications he receives, the bluetooth devices he connects to and the external devices that he connects to the phone (headset, powerplug). We refer to these different types as $features$ or *types*.

Smartphone logs contains multiple features ($Location$, $Notification$) where each feature can take multiple values. For example, feature $Location$ can take values $Home$, $Work$, the feature *Activity* can take values as $on\_foot$, $on\_bicycle$, $in\_vehicle$ and *still*. We refer to the values of a feature $frame$ as the *values* of feature $frame$ or the *vocabulary* of feature $frame$.

We note the set $F = \{1, .., J\}$ as the set representing a user logs containing J different features. Here, we refer to features as ids and no longer as names. We nite $f \in F$ as the feature number $f$.

Similarly, we note the set $V_f = \{1, .., I_f\}$ as the set representing the $I_f$ different values that can be taken by $f, f \in F$. We refer to values as ids and no longer as names. Note that $I_f$ represents the vocabulary size of feature $f$. $v \in V_f$ indicates the value number $v$ of the feature $f$.

Thus, the complete values space of the dataset is completely defined by the sets $F$ and $v_1, ..., v_f$. We refer to this *completevaluesspace* as the *language* defined by the dataset. Note that the size of the language is just the sum of the vocabulary sizes of all the features ($\sum_{f=1}^{F} I_f$).

A *datapoint*, also called a *realization* is then represented as the pair $(f, v)$ which means the $v^{th}$ value of the $f^{th}$ feature.

The user logs is nothing than a set of multiple realizations where each data point $(f, v)$ is linked with a time stamp indicating it s time of occurence.

Let the *record* be the set of realizations that occured during a certain time frame. Thus a record containing n realizations $\{(y_1, w_1), ..., (y_N, w_N)\}$ that all occured in the same time frame can be represented as vector $\mathbf{r} = [(y_1, w_1), ..., (y_N, w_N)]$. Here $y \in F$ denotes the feature of the $n^{th}$ realization and $w_n$ the values of the $n^{th}$ realization (i.e the value taken by the feature $y_n$).

Using the record definition, smartphone logs can be represented as a set of records where each record contains the data points that occured during a certain period in the time. A period could be for example 1 hour. In this case, each record represents 1 hour of the period of observation of a user.

Using this representation a smartphone logs can be represented as a set $R = \{\mathbf{r_1}, ..., \mathbf{r_M}\}$, of $M$ records, where each record $\mathbf{r_m}$ has a size $N_m$, for $m \in \{(1, ..., M\}$.

We call $R$ a corpus and when considering about this representation, we may refer to smartphone logs as a *corpus*.

## 2.2 Problem Statement

The goal that drives our work is to extract the behaviors and habits of a user from his smartphone logs data. In fact, all individuals have their own behaviors and habbits that they repeat periodically over the time. For example, an individual generally use to work during the week days, to sleep at home during the night. He might also be playing sport once a week, listening to music while driving or reading news when he is in the bus.

The question is, without previously knowing any behavior of the use, how well can we discover and extract the behaviors by analyzing a user logs. We can note that each of the examples of behaviors given above can be represented by the smatphone logs as a set containing a specific types of realizations. For example, the behavior working on the week days can be represented by the set { $(day : monday)$, $(day :$

$tuesday$),...,($day$ : $friday$),($hour$ : $8am - 6pm$), ($location$ : $work$)}. The behavior listening to music while driving could be represented by the set {($Activity$ : $invehicle$), ($ApplicationLaunch$ : $music$)}.

More generally, most of humans behaviors that can be expressed by smartphone logs that one think of can be represented as a group containing some realizations. For this reason and starting from this observations, the approach that we decide to take in our work is to consider a behavior as a set of data logs realizations.

We define a behavior z as a set of data realizations. By considering smartphone logs as a data showing a subsample of a user s life, then the task of finding the behaviors that are the most representative of his life is equivalent to find the groups of realizations that are the most descriptive of the smartphone dat. The terms *habbits*, *class* or *cluster* are also used to refer to a behavior.

Formulating our problem differently, we aim to find a set of $K$ classes $\{z_1, ..., z_K\}$, where each class is a group of realizations. This means that those classes represents short descriptions of the dataset while preserving the essential statistical relationships of the data. It is this requirement that imposes the classes to represent real user s behaviors.

In section 8, we discuss in more details what does this requirement mean, what does it imply and how it can be measured.

## 2.3 Related work

In this section we expose the different researches that have been done using smartphone logs and that are relevant to our work. Smartphone logs are very rich datasets containing multiple types of informations and with an abundant quantity, thus they can be used to achieve different goals and to tackle a large range of different problems. Etter and al. [**?** ] used smartphone logs to predict the next location to which the user will go. Zhu and al. [**?** ] used them to classify smartphone applications. We detail below two works that are specifically relevant to ours.

Cao and al. [**?** ] addressed in 2010 the following problem: having a dataset of smartphone logs, they tried to discover the context that causes a user to have a certain interaction with the phone. An interaction could be ?*listening to music*?, ?*reading news*?, ?*having a message session*?. A context is a set composed by the features that are not phone interactions. For example a Context is $C$ = {?$Is\ holiday$=$Yes$?, ?$Dayperiod$=$morning$?, ?$Phone\ profile$=$silent$?, ?$speed$=$High$?}. They were for example interested in learning that the context $C$ usually implies the interaction ?*reading*

*news*?.
To that end, they used association rules to learn the contextual information that leads to a phone interaction. The idea is to look through all the contextual features, build contexts and count how many times a context appears with a certain type of phone interaction. As checking all the combinations of the contextual features exponentially explode, thresholds $min_support$ and $min_confidence$ are used to only go through the promising contexts.
We can note that the problem tackled by [**?** ] is similar to our problem in the sense that they tried to learn a user behavior by processing his smartphone logs. However, they were interested in a specific type of behavior, the behavior of a user that leads him to make an interaction with his phone. In our work, we are considering smartphone logs as a sample of a user's life and are interested in discovering the behaviors that drive his life (and not only the ones related to the interaction with his smartphone). To that end we want to stress out that we do not input any prior knowledge or general behavior that humans might have. For example we do not take profit from the fact that most of people have different behaviors during the week days and the week ends (i.e separate the week end from the week day). This is to be able to catch any kind of behavior and in any kind of users. Indeed, if Bob is a farmer, he may want to take a rest of Fridays and not during the week ends.

In extension to this work, Ma et al. [**?** ] tried to detect similarities between users based on their habits (2012). Based on the same dataset than [**?** ] and taking the results of [**?** ] as input (i.e having learned context interaction relations for each user individually), they tried to cluster the users by behavior similarities. This work could be used for context-aware recommendation systems. Context aware recommendation systems take into account the context of the user to give him the right recommendation at the right time.
To extract super- behaviors (which are clusters of behaviors) of users, they use *Linearly Constrained Bayesian Matrix Factorization* ($LCBMF$) model described in [**?** ]. This method is a bayesian matrix factorization technique that enables to impose some prior linear constrains. $LCBMF$ showed good performances in [**?** ], and noting that our task is similar (extracting clusters from smartphone logs), we apply this method to our problem and discuss it in more details in **??**.

# Chapter 3

# Generative Hidden Class Model for Mixed Data Types (GHCM-MDT)

We recall that our task is the following: by observing user logs, we want to discover the behaviors and habits that describe his life.
To that end, we use a usual and common practice when trying to extract some hidden properties (behaviors) from an observable structure (logs): We assume that the logs we are observing are generated by behaviors. Then our task is to find the behaviors that generated the data we are observing. This practice drives the models we are going to discuss in the next sections.

## 3.1 Hidden Class Model for Mixed Data Types (HCM-MDT)

The first model we introduce is the Hidden Class Model for Mixed Data Types (HCM-MDT). As the title imply it, this model is used to model hidden (i.e unobserved) classes for a data that contains mixed types. The smartphone logs is a mixed types dataset in the sense that it contains multiple features and the hidden classes that we want to model are the behaviors.
This section is organized as follows. First, we describe the HCM-MDT model. Second, to better understand the utility and the intuitions that lead us to build the HCM-MDT model, we make a parallel between this model and the Probabilistic Latent Semantic Indexing (pLSI) [**?** ], which is a model that is widely used to model hidden classes in a

corpus of documents. More generally, pLSI can be used to model hidden classes in any kind of dataset that contain a unique type feature (for document corpus dataset, the unique feature is words).

### 3.1.1 HCM-MDT model

Let's consider the corpus representation of the smartphone logs. Smartphone logs are represented by a corpus containing $\boldsymbol{R}$ containing $M$ records where each record $\mathbf{r}$ is a vector representing the realizations that occurred in a given time frame $T$ ($T$ could be 1 hour for example).
The corpus defines a language containing features $F = \{1, ..., J\}$ where each feature $f \in F$ defines a vocabulary$V_f$ of $I_f$ values, $V_f = \{1, ..., I_f\}$. We assume that the corpus is described by $K$ behaviors for some integer $K$. We note these behaviors as $\{z1, ..., z_K\}$.
Let's suppose that we want to generate a record $\mathbf{r}$ of size $N$. Moreover, let's suppose that the record $\mathbf{r}$ we want to generate contains $N_1$ values form the vocabulary of feature 1, $N_2$ values from the vocabulary of feature 2, $N_f$ values from feature $f$ and $N_J$ values from feature $J$ ($\sum_{f=1}^{J} N_f = N$). For now, let's also assume that $\mathbf{r}$ contain al least one value from each feature (i.e $N_f \geq 1, \forall f \in F$).
To generate $\mathbf{r}$, we do the follwing:

1. generate the values coming from the vocabulary of feature 1 by doing the following:
   (a) generate the $1^{st}$ value $w_1$ by doing the following:
      i. choose one behavior $z_k, k \in [1, K]$ with probability $p(z_k|\mathbf{r})$, where $Z$ is a random variable that follows a multinomial distribution conditioned on the record $\mathbf{r}$
      ii. knowing the behavior $z_k, k \in [1, K]$, select a value from the vocabulary of feature 1, $w_1 \in V_1$ with probability $p(V = w_1|Z = z_k, F = 1)$, where $V$ is a random variable that follows a multinomial distribution conditioned on the behavior and the feature.
   (b) repeat the same process to generate the values $w_2, ..., w_{N_1}$
2. repeat the same process to generate the values of the features $2, ..., J$.

This process assumes that the generation of realizations in a same record are independent from each other (and their order does not matter). Moreover, it assumes that a realization is independent from the record it appears in when it is conditioned in the behavior. For now, we note that we assume that the number of values coming from each

feature in the record $\mathbf{r}$ are previously known $N_f \geq 1, \forall f \in F$. This assumption simplify the model and we will see that it does not impact it.

In smartphone logs, there is two different kinds of features. On the one hand, there is features that take values during all the time range of the observation. Examples if these features are "*Location*", "*Activity*" or "*Day*". In fact, a user is always in a location, doing some activity at a certain day. We call those features $permanent features$.
In the other hand, there is features that take values only in certain points in the time range of the observation. Examples are "$Application launches$", "$Notifications received$" or "$Bluetooth paired$". We call those features $temporary features$.
In the generation process described so far, we impose to select at least one value for each feature. This is a good representation for permanent features because it requires each permanent feature to have at least one value in each record. However, it is a bad representation for temporary features. Indeed, a user is not always running an application, receiving a notification or pairing his smartphone with another Bluetooth device. To address this problem, we enrich the language of the corpus $R$ as follows: a value is added to the vocabulary of each temporary feature. This value indicates that the concerned temporary feature is absent. We modify the records $\mathbf{r}_m, m \in \{1, ..., M\}$ pf the corpus $R$ accordingly by adding the realization $(f, v_{non\_present})$ to each record that does not have any realization for the temporary feature $f$. This transformation allows the the generation process described above to represent temporary features. Indeed, the value $v_{non\_present} \in V_f$ is selected with probability $p(V = v_{non\_present}|Z = z_k, F = 1)$ (when the behavior $z_k$ was selected). This models a record that do not contain the temporary feature $f$.

Using this model, the probability of obtaining a sequence of realizations $[(y_1, w_1), ..., (y_N, w_N)]$ belonging to record $\mathbf{r}$ can be derived. It is expressed as the probability of obtaining the sequence $[(y_1, w_1), ..., (y_N, w_N)]$ knowing that the record $\mathbf{r}$ was selected. It is computed as follows:

$$\begin{aligned} p([(y_1, w_1), ..., (y_N, w_N)]|\mathbf{r}) &= \prod_{n=1}^{N} p((y_n, w_n)|\mathbf{r}) \\ &= \prod_{n=1}^{N} \sum_{k=1}^{K} p(w_n|Z = z_k, F = y_n) p(Z = z_k|\mathbf{r}) \end{aligned} \tag{3.1}$$

We can also express $p([(y_1, w_1), ..., (y_N, w_N)], \mathbf{r})$ as:

$$\begin{aligned} = p([(y_1, w_1), ..., (y_N, w_N)], \mathbf{r}) &= p([(y_1, w_1), ..., (y_N, w_N)]|\mathbf{r}) p(\mathbf{r}) \\ &= \frac{1}{M} \prod_{n=1}^{N} \sum_{k=1}^{K} p(w_n|Z = z_k, F = y_n) p(Z = z_k|\mathbf{r}) \end{aligned} \tag{3.2}$$

Where $M$ represents the number of the records in the corpus.
Using Eq. (**??**) and Eq. (**??**), we can express the probability of a corpus $R = \{\mathbf{r}_1, ..., \mathbf{r}_M\}$ as the product of the probabilities of each record $\mathbf{r}_m$:

$$p(R) = p(\{\mathbf{r}_1, ..., \mathbf{r}_M\}) = \prod_{m=1}^{M} p([(y_1, w_1), ..., (y_N, w_N)]|\mathbf{r}) \tag{3.3}$$

Here note that $p(\mathbf{r}_m) = \frac{1}{M}$ disappeared from the equation. Indeed, as the order of selecting the records does not matter, then summing up over all the possible orderings of $\{\mathbf{r}_1, ..., \mathbf{r}_M\}$ cancels the factor $\frac{1}{M}$. We refer to the the probability of having a corpus $R$ as the likelihood $L$ of corpus $R$. The final expression of $L(R)$ can be derived using Eq. (**??**):

$$L(R) = p(R) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} \sum_{k=1}^{K} p(w_n|Z = z_k, F = y_n)p(Z = z_k|\mathbf{r}) \tag{3.4}$$

$HCM\_MDT$ assumes that the observed corpus was generated by the process described. However, the only thing known is the observed data.The distributions the generated the data $p(Z|\mathbf{r}_m), \forall m \in [1, M]$ and $p(V|z_k, f), \forall k \in [1, K]$ are unknown. Our problem becomes then finding these distributions. In that case, a good assumption is to say that good parameters are the parameters that maximize the likelihood $L(R)$ of the corpus we are observing.

When we model user logs according to $HCM\_MDT$, we assume that in each time frame, a user has a mixture of behaviors. When he knows his behaviors, he selects realizations according to these behaviors. This is a model that fits the real life in the sense that in a given range of time, an individual may act following one or more behaviors. Moreover, a behavior is defined by a set of events (i.e realizations) that may occur more or less probably. For example, let's assume that Bob goes to his gym on Saturdays, uses his car only to go either to work or to the gym and loves listening to music from his smartphone when driving. Let's suppose that Bob generates a record in some Saturday morning according to the $HCM\_MDT$ process. From all of his possible behaviors, Bob first selects with high probability the behavior $z_1 =$ "$do\_gym\_on\_Saturday\_Morning$" or the behavior $z_2 =$ "$listening\_to\_music\_in\_car$". Then Bob chooses from the selected behavior a location. Here the most probable location is "$gym$". Indeed location gym is the most probable in the behavior "$dogymonSaturday$". In, the behavior "$listeningtomusicincar$", location "$gym$" is equally probable with location "$work$" as Bob uses his car only to go to "$work$" or to "$gym$". After selecting a location, Bob chooses again a behavior between $z_1$ and $z_2$. From this behavior he chooses one activity which should be with high probability either "$running$" (if he selected $z_2$) or "$in_vehicle$" (if he selected $z_2$). He does

the same process to choose a day (which should be "*Saturday*" with high probability) an application launch ect... Here we see that the record generated by Bob following the generation model of $HCM\_MDT$ describes well Bob's life in some Saturday morning.

This example concludes the model description of $HCM\_MDT$. In the next part, we establish a relation between $HCM\_MDT$ and $pLSI$, which is a widely used algorithm to model hidden classes that describe a dataset. This allows us to expose a nice property of $HCM\_MDT$, that enables him to fit well the problem of modeling smartphone logs.

### 3.1.2 Relationship between HCM-MDT and Probabilistic Latent Semantic Indexing (pLSI)

*Probabilistic Latent Semantic Indexing*($pLSI$) was introduced in 1991 by C. J. Hawthorn[**?** ]. It came as a good alternative to the problem of modeling a corpus of text documents. This model allowed research to make big jump in information retrieval and text document representation. It constituted a significant step forward in modeling text document as a probabilistic model. Noting that smartphone logs can be seen as a corpus of documents where documents are records and words realizations, we briefly introduce the $pLSI$ model applied to a corpus $R$ of smartphone logs. $pLSI$ assumes the following generation process.
To generate a record $\mathbf{r} = [(y_1, w_1), ..., (y_N, w_N)]$, we do the following:

1. choose one behavior $z_k, k \in [1, K]$ with probability $p(z_k|\mathbf{r})$, where $Z$ is a random variable that follows a multinomial distribution conditioned on the record $\mathbf{r}$
2. knowing the behavior $z_k, k \in [1, K]$, select the realization $(y_1, w_1)$ where $y_1 \in F$ and $w_1 \in I_f$ with probability $p(V = (y_1, w_1)|Z = z_k)$, where $V$ is a random variable that follows a multinomial distribution conditioned on the behavior and the feature.
3. repeat the same process to generate the realizations $(y_2, w_2), ..., (y_N, w_N)$.

This generation process shows the similarity between $pLSI$ and $HCM\_MDT$. In both cases, we assume that a record is composed by a mixture of behaviors and each behavior is represented as a mixture of realizations. Moreover, a realizations are independent between each other, and a realization is independent from the record it appears in conditioned in the behavior.

In each hidden class $z_k$, $HCM\_MDT$ imposes that the distribution of the values of feature $f_i$ is completely independent from the distribution of the values of feature $f_j$

,$\forall f_j, f_i \in F, f_j \neq f_i$. In fact for each feature $f$, the the probabilities of its values in a given behavior $z_k$ sum to 1 ($\sum_{v=1}^{I_f} p(v|z_k, f) = 1$).
This requirement is one of the major strengths of $HCM\_MDT$. In fact, each feature represents a different type. Thus, there is no sense that two different types share the same distribution. For example, saying that location "*home*" is selected with probability 0.5 or day "$Monday$" is selected with probability 0.5 does not make sense. Indeed selecting the location and selecting the day are two *concurrent* events and not *competing* ones. This is because they belong to two different types (i.e features). However, it make sense to say that we select location "*home*" (day "*monday*") with probability 0.5 and location "*others*" (day "*others*") with probability 0.5. Indeed those two events are *competing* because they belong to the same type (i.e feature). The fact that each feature has his own distribution over values allows $HCM\_MDT$ to model the concept of *concurrent* and *competing* events, which is an essential aspect for a dataset containing multiple types as the smartphone logs dataset.
Actually, the main difference between $HCM\_MDT$ and $pLSI$ can be found here. In fact, $pLSI$ assumes a unique distribution for all the realizations for each class $z_k$. Thus, all the realizations share the same probability space and the representing the *concurrent* concept is not possible. For this reason, $HCM\_MDT$ is a much more suited model that $pLSI$ in dealing with datasets containing mixed types of features.

## 3.2 Generative HCM-MDT (GHCM-MDT) model

While $HCM\_MDT$ has nice properties on mixed data types, it still has some imperfections. In this section, we introduce the Generative Hidden Class Model for Mixed Data Types ($GHCM\_MDT$) which is an improved version of $HCM\_MDT$ in the sense that it addresses the weaknesses of $HCM\_MDT$ that we are going to discuss.

### 3.2.1 Prior distributions

To see how to proceed beyond $HCM\_MDT$, let's first have a look at some of it's weaknesses. First, it is important to note that $HCM\_MDT$ is incomplete in that it provides no probabilistic model at the level of records. In $HCM\_MDT$, each record is represented as a list of numbers (the mixing proportions of hidden classes), and there is no generative probabilistic model for these numbers. This leads to several problems: the number of parameters in the model ($KD + K\sum_{f=1}^{J} I_f$)grows linearly with the number of records and it is not clear how to assign a probability to a record outside the observed set.

In $GHCM\_MDT$, we assume that the mixing proportions of hidden classes have a prior distribution. This means that a probability is assigned to each possible distribution of hidden classes. In other terms, each possible distribution of hidden classes $d_i$ has a probability $p_{d_i}$ to happen and the probabilities of all the possible distributions sum to 1, which means $\int_{d_i} p_{di} = 1$.
Here, note that an integral was used to sum over the space of possible distributions. Indeed, the space of possible distributions over $K$ hidden classes is a continuous space defined as follows:

$$\begin{cases} 0 \leqslant p(z_k) \leqslant 1, \forall k \in \{1, ..., K\} \\ \sum_{k=1}^{K} p(z_k) = 1 \end{cases} \tag{3.5}$$

This is actually the $K - simplex$ space.

Dirichlet distribution [**?** ] is a distribution that takes as input a vector $\mathbf{p} = [p_1, ..., p_K]$ of $K$ elements and assigns positive distribution to $\mathbf{p}$ if it belongs to the simplex space. If $\mathbf{p}$ does not belong to the simplex space, it gets a probability of 0. In other terms, the Dirichlet distribution assigns positive probabilities to vectors that belong to the space of distributions and a 0 probability otherwise.
Dirichlet distribution depends on an hyperparameter vector $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_K], \alpha_k > 0, \forall k \in \{1, ..., k\}$. It is the values of $\alpha_k, \forall k \in \{1, ..., k\}$ that decide how the probabilities of Dirichlet are spread over the different possible distributions. For example with $\alpha_k = 1, \forall k \in \{1, ..., k\}$, Dirichlet distribution assigns equal probabilities to all the possible distributions $d_x$. For $\alpha_k \simeq 0, \forall k \in \{1, ..., k\}$, Dirichlet assigns very low probabilities to equally distributed distributions and high probabilities to sparse distributions. In [**?** ], J. Huang gives more detailed overview about the Dirichlet distribution.

Because Dirichlet distribution have these nice properties, we use this distribution of parameters $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_K]$to model the prior distribution of hidden classes in a corpus of records (i.e assign probabilities to the possible distributions of behaviors in a corpus a corpus $R$).
Simirarly, $GHCM\_MDT$ assumes a prior distribution over the mixture coefficients of realizations in a behavior $z_k$. This means that for each feature $f, \forall f \in F$ a Dirichlet distirbution of parameters $\boldsymbol{\beta}_f = [\beta_{f,1}, ..., \beta_{f,I_f}]$ assigns prior probabilities of the presence of of values of $f$ inside a behavior. Note that each feature $f$ has his own Dirichlet distribution and his own $\boldsymbol{\beta}_f$.

Having explained the steps that lead to $GHCM\_MDT$, we are ready to describe the generative process of $GHCM\_MDT$.

### 3.2.2 Generation process

For now we assume that the Dirichlet parameters $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ are known. Let's suppose that we want to generate a record $\mathbf{r}_1$ of size $N$ belonging to a corpus $R = \{\mathbf{r}_1, ..., \mathbf{r}_M\}$. Moreover, let's suppose that the record $\mathbf{r}_1$ we want to generate contains $N_1$ values form the vocabulary of feature 1, $N_2$ values from the vocabulary of feature 2, $N_f$ values from feature $f$ and $N_J$ values from feature $J$ ($\sum_{f=1}^{J} N_f = N$). For now, let's also assume that $\mathbf{r}_1$ contain al least one value from each feature (i.e $N_f \geq 1, \forall f \in F$). $GLMR$ assumes the following generation process:

1. for each hidden class $z_k, k \in [1, ..., K]$:
   (a) for each feature $f \in F$:
      i. select a distribution on the vocabulary of $f$ for hidden class $z_k$, $\boldsymbol{\phi_{f,k}}$ where $\boldsymbol{\phi_{f,k}}$ follows a Dirichlet distribution: $\boldsymbol{\phi_{f,k}} \sim Dir(\boldsymbol{\beta_f})$
2. generate $\mathbf{r}_1$ by doing the following:
   (a) choose a behavior proportions $\boldsymbol{\theta}$ for record $\mathbf{r}$ where $\boldsymbol{\theta}$ is a random variable vector following a Dirichlet distribution: $\boldsymbol{\theta} \sim Dir(\boldsymbol{\alpha})$
   (b) generate the values coming from the vocabulary of feature 1 by doing the following:
      i. generate the $1^{st}$ value $w_1$ by doing the following:
         A. choose one behavior $z_k, k \in [1, K]$ with probability $p(z_k|\boldsymbol{\theta})$, where $Z$ is a random variable that follows a multinomial distribution conditioned on the behavior distribution $\boldsymbol{\theta}$
         B. knowing the behavior $z_k, k \in [1, K]$, select a value from the vocabulary of feature 1, $w_1 \in V_1$ with probability $p(V = w_1|\boldsymbol{\phi_{1,k}})$, where $V$ is a random variable that follows a multinomial distribution conditioned on the behavior and the feature.
      ii. repeat the process **??** to generate the values $w_2, ..., w_{N_1}$
   (c) repeat the process **??** to generate the values of the features $2, ..., J$.
3. repeat the process **??** to generate the records $\mathbf{r}_2, ..., \mathbf{r}_M$.

The generation process described allows to fully generate a corpus of records. It provides ways to generate behaviors distribution for records and also vocabulary distributions for behaviors. Indeed $GLMR$ stars by generating $K$ sets of vocabularies distributions for all the corpus $R$ (i.e they are common to all the records). Each set of this $K$ sets contains $J$ distributions, meaning one distribution for the vocabulary of each feature. Actually,

this $K$ sets of vocabulary distributions, named $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}$, represents the $K$ behaviors describing the corpus $R$. After this, for each record $\mathbf{r}_m$, $GLMR$ generates a behavior distribution $\boldsymbol{\theta}_m$ for $\mathbf{r}_m$. Starting from that point, the generation process of $GLMR$ becomes exactly the same as the one of $LMR$. Actually, the difference between the two models can be expressed as follows. To be able to generates samples, $LMR$ needs to know a-priori $K$ sets of vocabulary distributions $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}$ and behavior distributions $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,...,M\}}$ for each record $\mathbf{r}_m$. For $GLMR$, it only needs to know $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ and generates $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}$ and $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,...,M\}}$ by his own. In this sense, we call it the *Generative LMR* ($GLMR$).

Following the steps of the generative process, we iteratively derive the likelihood $L(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})$ of the corpus $R$. First, $L(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})$ is equal to the likelihood of the corpus $R$ knowing the vocabularies distributions $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}$ summed over the probability of all possible vocabularies distributions.

$$\begin{aligned} L(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F}) &= p(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F}) \\ = \prod_{k=1}^{K} \prod_{f=1}^{J} \int_{\boldsymbol{\phi}_{f,k}} & p(\boldsymbol{\phi}_{f,k}|\boldsymbol{\beta}_f) L(R|\boldsymbol{\alpha}, \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \end{aligned} \tag{3.6}$$

Second, knowing the vocabularies distributions $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}$, the likelihood of the corpus is equal to the product of the likelihood of each record $\mathbf{r}_m$.

$$\begin{aligned} L(R|\boldsymbol{\alpha}, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) &= p(\{\mathbf{r}_1, ..., \mathbf{r}_M\}|\boldsymbol{\alpha}, \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \\ &= \prod_{m=1}^{M} p(\mathbf{r}_m|\boldsymbol{\alpha}, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \end{aligned} \tag{3.7}$$

Then, the likelihood of each record $\mathbf{r}_m$ can be expressed as the likelihood of the record $\mathbf{r}_m$ knowing its behavior distribution $\boldsymbol{\theta}_m$ summed over the probability of all possible behavior distributions.

$$p(\mathbf{r}_m|\boldsymbol{\alpha}, \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) = \int_{\boldsymbol{\theta}_m} p(\boldsymbol{\theta}_m|\boldsymbol{\alpha}) p(\mathbf{r}_m|\boldsymbol{\theta}_m, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \tag{3.8}$$

Finally, the likelihood of a record $\mathbf{r}_m$ knowing its behavior distribution $\boldsymbol{\theta}_m$ and vocabularies distributions $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}$ is expressed as the product of the likelihood of each realization. The latter quantity can then be computed similarly to $LMR$.

$$\begin{aligned} p(\mathbf{r}_m|\boldsymbol{\theta}_m, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) &= \prod_{n=1}^{N_m} p((y_n, w_n)|\boldsymbol{\theta}_m, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \\ &= \prod_{n=1}^{N_m} \sum_{k=1}^{K} p(w_n|\boldsymbol{\phi}_{y_n,k}) p(Z = z_k|\boldsymbol{\theta}_m) \end{aligned} \tag{3.9}$$

This ends our definition of $GHCM\_MDT$ model. We assume that the smartphone logs we observe were generated by $GHCM\_MDT$. An for $HCM\_MDT$. Even if we assumed until now that $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ are known, in reality the parameters that generated this data are unknown and the only observable is the data. Here again, a good assumption is to say that good guesses of those parameters are parameters that maximize the likelihood (i.e the probability) of the observed corpus $L(R)$.
In $GHCM\_MDT$, we note that the parameters that defines the model are the Dirichlet parameters $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$. Thus, the total number of parameters that needs to be estimated in $GHCM\_MDT$ is $K$ parameters for $\boldsymbol{\alpha}$ and $\sum_{f=1}^{J} I_f$ for $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$. We see that the number of parameters ($K + \sum_{f=1}^{J} I_f$ for $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$) to be estimated does not grow with the number of records (contrary to $HCM\_MDT$).

We conclude this section by exposing other nice properties and advantages of $GHCM\_MDT$ compared to $HCM\_MDT$. The prior distribution on the mixture of behaviors allows to model some prior knowledge that one might have. For example by choosing $\alpha_k \simeq 0, \forall k \in \{1, ..., k\}$, we can model the fact that a record should be composed by one or at most two behaviors (as $\alpha_k \simeq 0$ implies sparsity). This is a realistic assumption in the case where the time frame representing a record is small enough to assume that a user cannot have multiple behavior on that time frame.
The prior distribution over the distribution of values in a behavior allows to smooth the model. Indeed, while $HCM\_MDT$ would attribute a 0 probability to a new record containing a value never seen in observed records, $GHCM\_MDT$ would attribute a non null probability. This is thanks to the fact that the Dirichlet distribution do not include distributions that have 0 elements. In [**?** ], H. M. Wallach, D. Mimno and A. McCallum explain in more detail the influence and the power of Dirichlet parameters in giving a valuable prior knowledge to a problem.

Having explained $GHCM\_MDT$ model, the nice properties it comes with and the gain it brings with respect to $HCM\_MDT$, we close the circle n the next section by establishing relationships between $GHCM\_MDT$, $HCM\_MDT$, $pLSI$ and $LDA$ (*Latent Dirichlet Allocation*) [**?** ].

## 3.3 Relationship between GHCM-MDT, HCMC-MDT, pLSI and Latent Dirichlet Allocation (LDA)

*Latent Dirichlet Allocation* ($LDA$) was introduced by D. M. Blei, A. Y. Ng and M. I.Jordan [**?** ] in 2003. It came as an improvement to $pLSI$, and following its publication in 2003, $LDA$[**?** ] has made topic modeling one of the most popular and most successful

paradigms for both supervised and unsupervised learning. LDA has several applications including in entity resolution [**?** ], fraud detection in telecommunication systems [**?** ], and image processing [**?** **?** ], bioinformatics [**?** ] and political science [**?** ] in addition to the large number of applications in the field of text retrieval and computational linguistics [**?** ].

We briefly introduce $LDA$ applying it to the smartphone logs dataset.
To generate a record $\mathbf{r}_1 = [(y_1, w_1), ..., (y_N, w_N)]$ belonging to a corpus $R = \{\mathbf{r}_1, ..., \mathbf{r}_M\}$, we do the following:

1. for each hidden class $z_k, k \in [1, ..., K]$:
   (a) select a distribution $\boldsymbol{\phi_k}$ on the language of $R$ for hidden class $z_k$. The distribution $\boldsymbol{\phi_k}$ is of the size of the language and follows a Dirichlet distribution: $\boldsymbol{\phi_k} \sim Dir(\boldsymbol{\beta})$
2. generate $\mathbf{r}_1$ by doing the following:
   (a) choose a behavior proportions $\boldsymbol{\theta}$ for record $\mathbf{r}$ where $\boldsymbol{\theta}$ is a random variable vector following a Dirichlet distribution: $\boldsymbol{\theta} \sim Dir(\boldsymbol{\alpha})$
   (b) generate $(y_1, w_1)$ by doing the following:
      i. choose one behavior $z_k, k \in [1, K]$ with probability $p(z_k|\boldsymbol{\theta})$, where $Z$ is a random variable that follows a multinomial distribution conditioned on the behavior distribution $\boldsymbol{\theta}$
      ii. knowing the behavior $z_k, k \in [1, K]$, select the realization $(y_1, w_1)$ where $y_1 \in F$ and $w_1 \in I_f$ with probability $p(V = (y_1, w_1)|\boldsymbol{\phi_k})$, where $V$ is a random variable that follows a multinomial distribution conditioned on the behavior
   (c) repeat the same process **??** to generate the values $(y_2, w_2), ..., (y_N, w_N)$
3. repeat the process **??** to generate the records $\mathbf{r}_2, ..., \mathbf{r}_M$.

As described by its generation process, $LDA$ is a fully generative graphical model for describing the latent behaviors of records. $LDA$ models every behavior as a distribution over the realizations of the language, and every record has a distribution over the behaviors. These distributions are sampled from Dirichlet distributions. The realizations of the records are drawn from the realization distribution of a behavior which was just drawn from the behavior distribution of the document. We note that $LDA$ brings the same advantages over $pLSI$ than $GHCM\_MDT$ over $HCM\_MDT$. Indeed, $LDA$ provides prior distributions that enable to input some prior knowledge about the problem, the number of its parameters does not grow with the number of records, it provides a

probabilistic model at the level of records and at the level of hidden classes that enables to fully generate a corpus (and thus provides explicit way to assign a probability to a record outside the observed set) and handles realizations not seen in the observed logs. However, in $LDA$, the probability of values are spread over all the size of the language defined by corpus $R$. Thus, similarly to $pLSI$, $LDA$ fails in representing the *concurrent* and *competing* concepts. In $GHCM\_MDT$, the probabilities of values are spread only over the dictionary defined by their feature and the distributions of the different dictionaries are independent. As for $HCM\_MDT$, this enables $GHCM\_MDT$ to model the *concurrent* and *competing* concepts.
In other terms, $GHCM\_MDT$ combines the advantages of $LDA$ over $pLSI$ and $HCM\_MDT$ over $pLSI$. It is a fully generative model that is suited for data containing mixed types.

## 3.4 GHCM-MDT inference and parameter estimation

So far, we have described the motivation behind $GHCM\_MDT$, it's generation process and illustrated its conceptual advantages over other models.
In this section, we turn our attention to procedures for inference and parameter estimation.

### 3.4.1 Gibbs sampling

We recall that our main goal is to obtain $K$ behaviors expressed as a distribution over the vocabulary for each feature. However, after choosing the values of $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ in $GHCM\_MDT$, the exact distributions over behaviors $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}$ of the records $\{\mathbf{r}_m\}_{\forall m \in \{1,\dots,M\}}$ are not known. Moreover, the distribution of the vocabulary of each feature $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$ in the behaviors $\{z_k\}_{\forall k \in \{1,\dots,K\}}$ is also unknown. In fact, those distributions are needed because they are the ones that model the behaviors and the mixture of behaviors over the observed smartphone logs. In this section we suppose that the hyper-parameters ($\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$) that define the model are known and we describe a method that allows to estimate the distributions $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}$ and $\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$.

We simplify the notations by setting $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}$ and $\boldsymbol{\Phi} = \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$.
Note that $\boldsymbol{\Theta}$ represents $M$ vectors of dimension $K$ and $\boldsymbol{\Phi}$ represents $K.J$ vectors each of dimension $I_f$, where $f$ refers to the feature the vector belongs to. Knowing the corpus and the hyper-parameters, we can express the posterior probability of having $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$:

$$p(\boldsymbol{\Theta}, \boldsymbol{\Phi} | R, \boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F}) = \frac{p(\boldsymbol{\Theta}, \boldsymbol{\Phi}, R | \boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})}{p(R | \boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})} \tag{3.10}$$

Good estimates for $\mathbf{\Theta}$ and $\mathbf{\Phi}$ could be for example the estimates that maximize (**??**) or the expected value of (**??**). In all the cases (**??**) needs to be computed. Unfortunately, because of the normalization over $L(R)$, (**??**) is intractable to compute.

The same problem is encountered in $LDA$, and sophisticated approximations as variational inference [**?** ], expectation propagation [**?** ] and Gibbs sampling [**?** ] have been developed to estimate behaviors distributions in records and language distribution in behaviors. Our strategy for estimating $\mathbf{\Theta}$ and $\mathbf{\Phi}$ is to use the latter approach (Gibbs sampling) because it is intuitive to understand, easy to implement and shows similar performances compared to the other estimation methods [**?** ].
Gibbs sampling [**?** ] is a commonly used technique for $LDA$ and is described in details in that context in [**?** ]. We showed in **??** that $LDA$ and $GHCM\_MDT$ are very much related. This is also the case for the inference process using Gibbs sampling of both $LDA$ and $GHCM\_MDT$. For this reason, using references to works that used Gibbs sampling for $LDA$ allows us to make multiple shortcuts. We give below an overview of the Gibbs sampling method applied to $GHCM\_MDT$.

In this section, let a corpus $R$ be represented as a vector $\mathbf{R} = \{(y_1, w_1), ..., (y_S, w_S)\}$ containing the realizations of all the records where each realization $(y_s, w_s), s \in \{1, ..., S\}$ belongs to some record $\{\mathbf{r}_m\}, m \in \{1, ..., M\}$. Note that $S$ represents the total number of realizations present in the corpus (i.e $S = \sum_{m=1}^{M} N_m$).
Let's imagine that we want to assign a class $z_k$ to each realization $(y_s, w_s), s \in \{1, ..., S\}$ in the corpus, meaning that the realization $(y_s, w_s), s \in \{1, ..., S\}$ was generated by behavior $z_k$. Let $\mathbf{c} = [c_1, ..., c_s]$ represents those assignments, where $c_s \in \{z_1, ..., z_K\}$ represents the class assigned to realization $(y_s, w_s)$.
To guess $\mathbf{\Theta}$ and $\mathbf{\Phi}$, the idea is to estimate the posterior distribution $p(\mathbf{c}|\mathbf{R})$ of the class assignments $\mathbf{c}$ knowing the observed realizations (In fact, in [**?** ], it is shown that $p(\mathbf{c}|\mathbf{R})$ cannot be computed, and thus need to be estimated). Intuitively, $p(c_s|(y_s, w_s))$ represents the responsibility of $(y_s, w_s)$ in generating behavior $c_s$. Then, we can use the estimates of the responsibilities of realizations in generating the different classes ti estimate $\mathbf{\Theta}$ and $\mathbf{\Phi}$.

To estimate the class assignments of each observed realization, we assume a Monte Carlo Markov chain [**?** ] where each state represents a possible class assignments vector $\mathbf{c}$, and transitions follow a simple rule. The next state is sampled sequentially by sampling all variables from their distribution when conditioned on the current values of all other variables and the data $p(c_s|\mathbf{c}_{-s}, \mathbf{R})$ where $\mathbf{c}_{-s}$ the vector of assignments $\mathbf{c}$ from which we remove the $s^{th}$ assignment $c_s$.

Simirarly to [**?** ], this probability can be computed and can be expressed as follows:

$$p(c_s|\mathbf{c}_{-s},\mathbf{R}) \sim \frac{n_{-s,k}^{(y_s,w_s)}+\beta_{y_s,w_s}}{n_{-s,k}^{(y_s,.)}+\sum_{v=1}^{I_{y_s}}\beta_{y_s,v}}.\frac{n_{-s,k}^{r_m}+\alpha_k}{n_{-s,.}^{r_m}+\sum_{h=1}^{K}\alpha_h} \tag{3.11}$$

where:

- $n_{-s,k}^{(y_s,w_s)}$ represents the number of values $v = w_s$ from the dictionary of feature $f = y_s$ that are assigned to behavior $z_k$ in the assignments vector $\mathbf{c}_{-s}$ (i.e without taking into account the assignment $c_s$).
- $n_{-s,k}^{(y_s,.)}$ represents the number of all values $\forall v \in V_{y_s}$ belonging to the dictionary of $f = y_s$ that are assigned to behavior $z_k$ in the assignments vector $\mathbf{c}_{-s}$.
- $n_{-s,k}^{r_m}$ represents the number of realizations from the record $\mathbf{r}_m$ that are assigned to behavior $z_k$ in the assignments vector $\mathbf{c}_{-s}$. Here, $\mathbf{r}_m$ represents the record to which belong the $s^th$ realization $(y_s, w_s)$
- $n_{-s,.}^{r_m}$ represents the number of realizations from the record $\mathbf{r}_m$ that have an assignment in the assignments vector $\mathbf{c}_{-s}$ (i.e $n_{-s,.}^{r_m}$=$N_m - 1$).
- $\beta_{f,v}, f \in F, v \in V_f$ represents $v^{th}$ value of the prior vector of feature $f$, $\boldsymbol{\beta}_f$.

In comparison to the equivalent equation in $LDA$ [**?** ], $GHCM_MDT$ normalizes Eq. (**??**) by summing only over the dictionary of the feature considered, whereas $LDA$ normalizes by summing over the whole language.
Having obtained the full conditional distribution, the Monte Carlo algorithm is then straightforward. The $c_s$ variables are initialized to values in $\{z_1, ..., z_K\}$, determining the initial state of the Markov chain. Then we start moving from state to state following the rule described in Eq. (**??**) until we reach a state where the Markov Chain has converged. The state $\mathbf{c}_{conv}$ we converge to is a good estimate of classes assignments of the corpus $\mathbf{R}$. In practice, we usually record multiple assignments $\mathbf{c}_{conv}$ and combine them to improve the approximation of the class assignments. In [**?** ], the effect of averaging multiple samples is discussed in more details.
When having estimated $\mathbf{c}_{conv}$, approximations of $\boldsymbol{\Theta}$ can be exactly computed as for $LDA$ [**?** ] (Indeed, $GHCM_MDT$ and $LDA$ assume exactly the same probabilistic model at the record level). For each record $\mathbf{r}_m$ and behavior $z_k$, we have:

$$\widehat{\theta}_{m,k} = p(z_k|\mathbf{r}_m) = \frac{n_k^{r_m}+\alpha_k}{n_k^{(.)}+\sum_{h=1}^{K}\alpha_h} \tag{3.12}$$

$\boldsymbol{\Phi}$ can be derived in a similar manner that for $LDA$ [**?** ]. The only difference is that in $GHCM_MDT$, the dictionaries of the different features is considered separately whereas

in $LDA$ all the language is considered at the same time. The approximation $\widehat{\phi}_{f,k,v}$ of the probability to generate the value $v$ of the feature $f$ from the behavior $z_k$ is:

$$\widehat{\phi}_{f,k,v} = p(v|z_k, f) = \frac{n_k^{(f,v)} + \beta_{f,v}}{n_k^{(f,.)} + \sum_{v=1}^{I_f} \beta_{f,v}} \tag{3.13}$$

where $n$ in both Eq. (**??**) and (**??**) is defined as in Eq. (**??**).

### 3.4.2 Hyperparameters estimation

So far, we have considered that $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ are fixed. However, as we discussed in section **??**, we suppose that the observed smartphone logs were generated by a $GHCM\_MDT$ model with parameters that maximize the likelihood $L(R)$ of the observed corpus $R$. This means that the optimal parameters $\widehat{\boldsymbol{\alpha}}_{\boldsymbol{best}}$ and $\{\widehat{\boldsymbol{\beta}}_{\boldsymbol{best}f}\}_{\forall f \in F}$ are parameters that set the derivative of $L(R)$ to 0. However, these equations are not solvable. Indeed, as many models based on latent variables, the result is a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation. In this section, we explain how $\widehat{\boldsymbol{\alpha}}_{\boldsymbol{best}}$ and $\{\widehat{\boldsymbol{\beta}}_{\boldsymbol{best}f}\}_{\forall f \in F}$ can be estimated.

We recall that the hyper-parameter $\boldsymbol{\alpha}$ ($\{\boldsymbol{\beta}_f\}_{\forall f \in F}$) is the concentration parameter to a Dirichlet distribution, from which the multinomial probability vectors $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}$ ($\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$) are generated. Thus, $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}$ ($\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$) are all conditionally independent given $\boldsymbol{\alpha}(\{\boldsymbol{\beta}_f\}_{\forall f \in F})$, and $\boldsymbol{\alpha}(\{\boldsymbol{\beta}_f\}_{\forall f \in F})$ is conditionally independent from all the other variables given $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}(\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}})$. This means that if $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\dots,M\}}$ and $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$ are known, then finding $\widehat{\boldsymbol{\alpha}}_{\boldsymbol{best}}$ and $\{\widehat{\boldsymbol{\beta}}_{\boldsymbol{best}f}\}_{\forall f \in F}$ reduces to the problem of estimating a Dirichlet parameter from the observed multinomial distribution that it generated.

In [**?** ], T. P. Minka shows different methods of inferring a Dirichlet parameter $\boldsymbol{\alpha}$ when samples generated from that Dirichlet are observed. In particular, if a Dirichlet distribution with an unknown parameter vector $\boldsymbol{\alpha}$ (of size $K$) has generated multinomial distributions $\{\boldsymbol{\theta}_m\}$, and if some discrete observable samples $\{z_k\}$ where drawn from $\{\boldsymbol{\theta}_m\}$, then the $\widehat{\boldsymbol{\alpha}}_{\boldsymbol{best}}$ that generated those distributions can be found iteratively using the following equation:

$$\alpha_k^{new} = \alpha_k \frac{\sum_{m=1}^{M} (\Psi(n_k^m + \alpha_k)) - \Psi(\alpha_k))}{\sum_{m=1}^{M} (\Psi(n_{(.)}^m + \sum_{k=1}^{K} \alpha_k) - \Psi(\sum_{k=1}^{K} \alpha_k))} \tag{3.14}$$

where:

- $n_k^m$ represents the number of times the $k^{th}$ sample $z_k$ were observed during the samplings from the $m_{th}$ multinomial distribution $\boldsymbol{\theta}_m$ generated by the Dirichlet distribution.
- $n_{(.)}^m$ represents the number of observed samplings from the $m_{th}$ multinomial distribution $\boldsymbol{\theta}_m$ generated by the Dirichlet distribution..
- $\Psi$ is the digamma function.

Using equation Eq. (**??**), finding the estimates $\widehat{\boldsymbol{\alpha}}_{\boldsymbol{best}}$ and $\{\widehat{\boldsymbol{\beta}}_{\boldsymbol{best}f}\}_{\forall f \in F}$ becomes straightforward. First, we initialize $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ with some random values, then we compute new values for $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ after each Gibbs sampling cycle until convergence using the following equations:

$$\alpha_k^{new} = \alpha_k \frac{\sum_{m=1}^{M}(\Psi(n_k^{r_m}) - \Psi(\alpha_k))}{\sum_{m=1}^{M}(\Psi(n_{(.)}^{r_m} + \sum_{k=1}^{K}\alpha_k) - \Psi(\sum_{k=1}^{K}\alpha_k))} \tag{3.15}$$

- $n_k^{r_m}$ represents the number of realizations from the record $\mathbf{r}_m$ that are assigned to behavior $z_k$ in the assignments vector $\mathbf{c}$ (resulted from Gibbs sampling). Here, $\mathbf{r}_m$
- $n_{(.)}^{r_m}$ represents the number of realizations from the record $\mathbf{r}_m$ that have an assignment in the assignments vector $\mathbf{c}$ (i.e $n_{\cdot}^{r_m}$=$N_m$).

$$\beta_{f,v}^{new} = \beta_{f,v} \frac{\sum_{k=1}^{K}(\Psi(n_k^{(f,v)} + \beta_{f,v}) - \Psi(\beta_{f,v}))}{\sum_{k=1}^{K}(\Psi(n_{(k)}^{(f,.)} + \sum_{u=1}^{I_f}\beta_{f,u}) - \Psi(\sum_{u=1}^{I_f}\beta_{f,u}))} \tag{3.16}$$

- $n_k^{(f,v)}$ represents the number of values $v$ from the dictionary of feature $f$ that are assigned to behavior $z_k$ in the assignments vector $\mathbf{c}$
- $n_{-k}^{(f,.)}$ represents the number of all values $\forall v \in V_f$ belonging to the dictionary of $f$ that are assigned to behavior $z_k$ in the assignments vector $\mathbf{c}$

This ends our discussion about $GHCM_MDT$. In this chapter, we exposed a model that was specifically built to answer our needs. It is a fully generative process, resists to overfitting thanks to a number of parameters that does not grow with the size of the corpus and learns prior knowledge about the problem that enables it to deal with new coming data (new records and new realizations). Moreover, it is able to treat different types separately by representing them on different distributions and in the same time combine them to represent hidden classes that are commonly described by the different types. We also showed efficient methods to estimate the parameters and the hidden variables from the observed data.

# Chapter 4

# Matrix Factorization Models

In Chapter 3, we exposed in details the $DLMR$ model and talked about $LMR$, $pLSI$ and $LDA$ which are all based on a probabilistic modeling of the smartphone logs. However, the probabilistic approach in not the only way to tackle the problem. To have a complete overview, we present in this chapter models based on a matrix factorization approach that can extract common latent characteristics (i.e behaviors) from the smartphone logs. First we introduce *Singular Value Decomposition* ($SVD$) [**?** ] technique. Then, second we talk about *Linearly Constrained Bayesian Matrix Factorization* ($LCBMF$) [**?** ].

## 4.1 Singular Value Decomposition (SVD)

*Singular Value Decomposition* ($SVD$) [**?** ] has is a common technique for analysis of data and has a wide range of applications. In is for example used in $LatentSemanticIndexing$ ($LSI$) [**?** ], the ancestor of $pLSI$. We briefly present in this section how $SVD$ is applied to the smartphone logs.

### 4.1.1 matrix representation of the smartphone logs

In this section, the corpus of smartphone logs $R$ in represented as a matrix $\boldsymbol{X}$ of $I$ rows and $M$ columns. Each column vector represents a record $\mathbf{r}_m$. In this section $\mathbf{r}_m$ is a vector of the size of the language defined by $R$, $I = \sum_{f=1}^{J} I_f$ where each dimension represents one possible realization. In this part, each dimension contains the number of times the corresponding realization is observed.

$SVD$ finds an approximation $\widehat{\boldsymbol{X}}$ of $\boldsymbol{X}$ that minimizes the Frobenius norm $\left\|\boldsymbol{X} - \widehat{\boldsymbol{X}}\right\|^2$. $\widehat{\boldsymbol{X}}$ which is expressed as the product of three matrixes $\boldsymbol{U}$ of dimension $I \times K$, $\boldsymbol{S}$ $K \times K$

and $\boldsymbol{V}$ $K \times M$, such that $\boldsymbol{S}$ is a diagonal matrix, $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrixes. The tree factorizing matrixes can be interpreted as follows. each of the $K$ columns of $\boldsymbol{U}$ could be interpreted as a behavior, where each dimension of the column vector (of size $I$) contains a value indicating how much the corresponding realization is correlated (positively or negatively) with the given behavior. The diagonal vector of matrix $\boldsymbol{S}$ (of size $K$) contains by definition positive values that represent the importance of each behavior in the matrix $X$. Finally, each of the $M$ columns of $\boldsymbol{U}$ could be seen as the behaviors importance in each record, where each dimension of the $m^{th}$ column vector (of size $K$) contains a value indicating how much each behavior is important to describe the record $\mathbf{r}_m$.

### 4.1.2 TF-IDF transformation

As described in **??** $SVD$ objective function is to minimize the absolute distance $\left\|\boldsymbol{X} - \widehat{\boldsymbol{X}}\right\|^2$. This implies that the dimensions $i \in \{1, ..., I\}$ (i.e realizations) that are the most recurrent in $\boldsymbol{X}$ will catch the most the attention of $SVD$ in minimizing its objective function. Thus the most recurrent realizations will get the most importance in the task of finding $\widehat{\boldsymbol{X}}$. While this may seems a good thing in the first look (one could argue that the most recurrent realizations should get the most importance because they are the most representative of $X$), it is not true that the most recurrent realizations are the most representative of the data. Indeed, if *Bob* is always at home and never leaves it, then knowing that Bob is at home does not help in describing current Bob behavior (as it is always the case). Indeed, the very frequent realizations are not carrying meaning and for this reason they should not concentrate all the importance of $SVD$. This problem is also faced in document corpus when using $SVD$ (in $LSI$).

To face this problem, we use the $Term\ Frequency\ Inverse\ Document\ Frequency$ transformation ($TF_IDF$) [**?** ] on the matrix $\boldsymbol{X}$. This transformation applies a weighting to the different dimensions of $\boldsymbol{X}$ so that it gives more importance to the occurrences that are the most meaningful in describing the data. Each occurrence in $\boldsymbol{X}$ is wighted according to to two criteria: its importance inside the record and its importance inside the corpus. It translates the following intuitions: A realization that is frequent is a record is important in describing this record. A realization that is rare in the corpus is carries meaning when it is present in a record.

## 4.2 Linearly Constrained Bayesian Matrix Factorization (LCBMF)

While $SVD$ provides nice and easy ways in factorizing matrixes and expressing statistical dependence of data, it suffers from some non desirable properties. First, $\boldsymbol{U}$ and $\boldsymbol{V}$ contains both positive and negative values which make them complicated are not intuitive to interpret. Second, there is not an objective measurement indicating the quality of the resulting factorization (estimation). *Bayesian Non Negative Matrix Factorization* [**?** ] came as a response to these problems. In fact, they factorize a matrix $\boldsymbol{X}$ in factorizing matrixes containing only positive values and provide uncertainty measures of the factorizations.
*Linearly Constrained Bayesian Matrix Factorization* ($LCBMF$) is bayesian matrix factorization that provides the possibility to impose any kind of linear constrains to the factorizing matrixes [**?** ]. It factorizes a matrix $\boldsymbol{X}$ into two matrixes $\boldsymbol{A}$ and $\boldsymbol{B}$ where the elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ are subject to some equality and inequality constrains. In this section, first we briefly show some applications where $LCBMF$ is used and how the constrains were chosen. Then, we explain the constrains chosen for our problem.

### 4.2.1 LCBMF applications and examples

$LCBMF$ is evaluated in [**?** ] to for blind source separation. Handwritten digits grayscale images are superposed randomly to create a dataset of mixed images (each image is a mix of two hand written digits), then the task is to see how well can the sources that generated those images (i.e the handwritten digits) can be recovered. Ideally, one would expect to find nine separate sources where each source represents a different digit. Noting the dataset of mixed images as a matrix $\boldsymbol{X}$ where each column vector represents a mixed image, $\boldsymbol{X}$ $I \times M$ expressed as the factor of two matrixes $\boldsymbol{A}$ $I \times K$ and $\boldsymbol{B}$ $K \times M$ where $K$ is the number of the target hidden sources. By imposing that each column vector in $\boldsymbol{B}$ sums to 1 and each element in $\boldsymbol{A}$ to be between 0 and 1 (note that input grayscale images represent intensities of gray that are between 0 and 1), we force the original matrix $\boldsymbol{X}$ to be expressed as the sum of original grayscale sources ($\boldsymbol{A}$) mixed with different proportions (coefficients $\boldsymbol{B}$) (Indeed, each original element $x_{im}$ is expressed as the sum of $x_{im} = \sum_{k=1}^{K} a_{ik} b_{km}$). The linear constrains imposed to $\boldsymbol{A}$ and $\boldsymbol{B}$ allowed $LCBMF$ to adapt to the specific problem of finding handwritten sources digits from mixed images. The results in [**?** ] shows that $LCBMF$ was successful to recover the original digits which is not the case of the other factorizing approaches.

in [**?** ], Ma et al. wanted to extract common behaviors of users based on their smartphone logs. They had a matrix $\boldsymbol{X}$ $I \times M$ where the each row represents a type of behavior

and each column represent a user. The column $m^{th}$ column $\boldsymbol{x}^m$ represents the behaviors of the user $m$ where each dimension $i$ contains the number of times the behavior $i$ was observed in the user $m$. To find common users' behaviors (called super-behaviors) from this matrix, they assumed that each user act as a mixture of super-behaviors, where each super behavior is represented as a vector of size $I$ containing the importance of each simple behavior. Thus, they used $LCBMF$ to decompose $\boldsymbol{X}$ into $\boldsymbol{A}$ $I \times K$ and $\boldsymbol{B}$ $K \times M$, where $k^{th}$ column of $\boldsymbol{A}$ represents the super-behavior $k$ and the $m^{th}$ column of $\boldsymbol{B}$ represents the mixture coefficients of the super-behaviors for the user $m$. For this, they imposed that each column vector in $\boldsymbol{B}$ sums to 1 (representing mixture over super-behaviors) and element in $\boldsymbol{A}$ to positive 0.

Those two examples show the ability of $LCBMF$ to adapt to different problems and expose the large degree of freedom provided by the ability to fix linear constrains. In the next part, we show the linear constrains that we used to answer our problem.

### 4.2.2 Linear constrains for smartphone logs matrix

In this part, we represent smartphone logs as a matrix $\boldsymbol{X}$ of $I$ rows and $M$ columns. As in , each column vector represents a record $\mathbf{r}_m$. Moreover, $\mathbf{r}_m$ is also a vector of the size of the language defined by $R$, $I = \sum_{f=1}^{J} I_f$ where each dimension represents one possible realization. However, in this part, each dimension contains the number of observation of a given realization $(f, v)$ normalized by the total number of realizations observed for feature $f$. This means that in each records, the values attributed to the different realizations belonging to same feature sum to 1. For example, if Bob opened 2 times his preferred news application and 1 time his e-mail application (during the time frame of a record observation) then the realization corresponding to ("*application launch*", "*news app*") would get the score of $\frac{2}{3}$ and the realization corresponding to ("*application launch*", "$e - mail\ app$") would get the score of $\frac{1}{3}$. The realizations corresponding to the other application launches would get the score of 0. By doing this, the records would represent the probabilities of the user's actions.

Having $\boldsymbol{X}$, the goal is to extract the user's behaviors. Thus, we use $LCBMF$and we impose the matrix $\boldsymbol{A}$ $I \times K$ to represent the behaviors and the matrix $\boldsymbol{B}$ $K \times M$ to represent the mixture coefficients of the behaviors in each record. We do this by setting each column vector $\mathbf{b}_m$ in $\boldsymbol{B}$ sums to 1 (to represent the coefficients of behaviors in record $m$). Concerning the columns of $\boldsymbol{A}$, we impose the values attributed to the realizations of the same feature to sum to 1 (the same as the columns of the original matrix). Thus, an observed record $\mathbf{r}_m$ would be obtained by the mixture of the different behaviors

represented by the columns $\{\mathbf{a}_k\}_{k=\{1,\dots,K\}}$ of matrix $\boldsymbol{A}$. Indeed a record vector $\mathbf{r}_m$ is expressed as by $\mathbf{r}_m = \sum_{k=1}^{K} b_{k,m}\mathbf{a}_k$.

This concludes the chapter about matrix factorization models. In this chapter we presented two different matrix factorization techniques that are candidate to solve our problem. We talked about $SVD$ a common and widely used matrix factorization technique and $LCBMF$ a sophisticated and powerful technique that we adapted according to our needs.

# Chapter 5

# Evaluation metrics

Our goal is to find clusters that represent behaviors of a user from his smartphone logs. Thus, the clusters found by the different models must be really representative of the behavior of the user and not just some random realizations clustered together. For this reason, a way to verify this point is needed. So far, we presented different models that are able to complete the task of finding clusters from smartphone logs. Some of them use a probabilistic approach ($GLMR$, $LMR$, $LDA$, $pLSI$) whereas the others rely on matrix factorization techniques ($SVD$, $LCBMF$). We develop in this chapter metrics we use to evaluate how much the hidden classes found by a given model are representative of the life of the user. Recalling that smartphone logs are a subsample of a user's life, the common idea that drives those metrics is the following: If the clusters found by analyzing smartphone logs are able to well describe future coming logs (i.e unseen data) from the same user, then those clusters correspond to behaviors that the user use to follow.

## 5.1 Features prediction

To see how well the clusters resulting from a model $\mathfrak{M}$ describe an unseen data, the ability of the model to guess the values of a missing feature from a new given record is a good indicator. In other terms, if the realization of a feature $f$ is removed from new a record $\mathbf{r}$, the ability of $\mathfrak{M}$ in guessing this feature by observing the context present in $\mathbf{r}$ indicates how well it generalizes in the data. Indeed, making a good guess means that the clusters produced by $\mathfrak{M}$ were able to represent the context described by a record $\mathbf{r}$ that was not previously seen. For example, if $\mathfrak{M}$ is able to learn that Bob loves listening to music from his smartphone while driving (i.e one of the clusters $z$ of $\mathfrak{M}$ represents the behavior $z =$ "$listening\_to\_music\_in\_car$"), and if a record of Bob listening to music

(from which the activity of Bob is removed) is given to $\mathfrak{M}$, then $\mathfrak{M}$ would guess that Bob is probably driving (i.e feature $Activity = "in_vehicle"$). In the same way, if $\mathfrak{M}$ is able to learn that Bob goes to the gym on Saturdays morning, and if a record showing that Bob is in the Gym is given to $\mathfrak{M}$, then $\mathfrak{M}$ could guess that this record is probably happening a Saturday morning. Because it is a criterion representative of clustering quality, predicting (i.e guessing) missing entries from new data inputs is a common criterion used to compare and evaluate the performance of latent class models. It is used by T. Hofmann [Probabilistic Latent Semantic Indexing][**?** ] to proof that $pLSI$ performs better than it's ancestor $LSI$[**?** ] and used again by D. M. Blei, A. Y. Ng and M. I.Jordan [Latent Dirichlet Allocation][**?** ] to show that $LDA$ performs better than $pLSI$. It is also used in [**?** **?** **?** ].

Concretely, this is done by splitting the corpus of smartphone logs $R$ in two parts: a train corpus $R_{train}$ and a test corpus $R_{test}$. $R_{train}$ is used to fit the parameters of the model $\mathfrak{M}$ and find the clusters. Then, the ability of $\mathfrak{M}$ to correctly guess missing the feature $f$ is evaluated in the records belonging to $R_{test}$ from which the realizations belonging to $f$ are removed. We note those vectors $[\mathbf{r}_1^{-f}, ..., \mathbf{r}_{M_{test}}^{-f}]$ where $M_{test}$ is the number of records in $R_{test}$. We spread the values $V_f$ in different target categories that we want to guess (for example we divide the dictionary of feature *Location* in 3 categories $most\_frequent\_location$, $second\_most\_frequent$, $others$). Then for each record $\mathbf{r}_m^{-f}, m \in \{1, ..., M_{test}\}$, the model $\mathfrak{M}$ makes a prediction $v_m^{pred} = \mathfrak{M}(\mathbf{r}_m^{-f}), v_m^{pred} \in V_f$. We say that $\mathfrak{M}$ made a good guess for $\mathbf{r}_m^{-f}$ if $v_m^{pred}$ belongs to the right category.
We define $Accuracy_{\mathfrak{M}}$ as the rate of good guesses made by model $\mathfrak{M}$. In other terms,

$$Accuracy_{\mathfrak{M}} = \frac{n_{good\_guesses}}{n_{tries}} \tag{5.1}$$

where

- $n_{good\_guesses}$ represents the number of good guesses made by $\mathfrak{M}$.
- $n_{tries}$ represents the attempts made by $\mathfrak{M}$. Note that $n_{tries} = M_{test}$.

However, while $Accuracy_{\mathfrak{M}}$ seems to be a nice indicator for clustering and generalization performance, it suffers from a problem when dealing with unbalanced categories' sizes. Indeed, if Bob is 80% of the time at home, and leaves sometimes his home to go at work or to the gym, then a naive classifier that would always guess the most frequent location of Bob would make a high score of 0.8 of good guesses. For this reason, we combine $Accuracy_{\mathfrak{M}}$ with another indicator that computes the rate of good predictions for each category separately and average them. We call this indicator $Average\_Accuracy_{\mathfrak{M}}$. Noting $C = \{C_1, C_2, ...\}$ the set of the different target categories,

we can express $Average\_Accuracy_{\mathfrak{M}}$ as follows:

$$Average\_Accuracy_{\mathfrak{M}} = \frac{1}{|C|} \sum_{C_i \in C} \frac{n_{good\_guesses,C_i}}{n_{(.,C_i)}} \tag{5.2}$$

where

- $n_{good\_guesses,C_i}$ represents the number of times $\mathfrak{M}$ guessed class $C_i$ and the guess was correct.
- $n_{(.,C_i)}$ represents the number of times $\mathfrak{M}$ should have guessed class $C_i$. Note that $n_{(.,C_i)}$ is equal to the number of records containing a value belonging to $C_i$ (before removal of values).

Note that naive classifier described above would have a score of $Average\_Accuracy_{naive} = \frac{1}{3}(Accuracy_{Home} + Accuracy_{Work} + Accuracy_{Others}) = 0.33$, which represents a low score. To evaluate the generalization performances of a model, one should always look at the scores of both$Accuracy$ and $Average\_Accuracy$.

To make the guesses $v_m^{pred} = \mathfrak{M}(\mathbf{r}_m^{-f})$, probabilistic models use maximum likelihood while matrix factorization models use projection space techniques. In the next sections, we explain each of the two techniques.

## 5.2 Maximum Likelihood

We presented in chapter 3 methods that model a corpus of records as a probability distribution and provide ways to compute the likelihood $L$, which is the probability of having some observed samples. In this section, we take profit from this characteristic to make guesses by selecting the most probable guess (i.e the guess that has the maximum likelihood).

In $LMR$, the probability of a value $v \in V_f$ given a record $\mathbf{r}^{-f}$ is obtained from the posterior distribution over behaviors (i.e clusters):

$$p(v|\mathbf{r}^{-f}) = \sum_{k=1}^{K} p(v|Z = z_k, F = f) p(Z = z_k|\mathbf{r}^{-f}) \tag{5.3}$$

$p(v|Z = z_k, F = f)$ are the values distributions in behaviors already estimated during training, while $p(Z = z_k|\mathbf{r}^{-f})$ is estimated using the $Expectation\ Maximization$ $(EM)$ algorithm described in [**?** ].

In $GLMR$ and $LDA$, the probability of a value $v \in V_f$ given a record $\mathbf{r}^{-f}$ is obtained by integrating over the posteriors. For $GLMR$ this gives:

$$p(v|\mathbf{r}^{-f}) = \int_{\boldsymbol{\theta}_{test}} p(\boldsymbol{\theta}_{test}|\mathbf{r}^{-f}, \boldsymbol{\alpha}) \sum_{k=1}^{K} p(w_n|\widehat{\boldsymbol{\phi}}_{y_n,k}) p(Z = z_k|\boldsymbol{\theta}_{test}) \tag{5.4}$$

where $\widehat{\boldsymbol{\Phi}} = \{\widehat{\boldsymbol{\phi}}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\dots,K\}}$ are the vocabularies distributions estimated on $R_{train}$. As seen in **??**, integrating over $\boldsymbol{\theta}_{test}$ is not possible. Thus, one could estimate a behavior distribution $\widehat{\boldsymbol{\theta}}_{test}$ for record $\mathbf{r}^{-f}$ by proceeding as behaviors distributions are estimated for training set **??**. In our work, we use a variant of Collapsed Gibbs Sampling, a technique developed by Y. Papanikolaou, T. N. Rubin and G. Tsoumakas in [**?** ] for treating unseen records. Indeed, it is shown that this estimation technique for unseen records performs better than the traditional ones. This technique was initially developed for $LDA$, but thanks to the similarities between $LDA$ and $GLMR$, it is easily adapted to $GLMR$. Moreover, to obtain the final estimation $\widehat{\boldsymbol{\theta}}_{test}$, we average multiple intermediate samples using the method described in [**?** ].

Similarly for $LDA$, the probability of a value $v \in V_f$ given a record $\mathbf{r}^{-f}$ is expressed as

$$p(v|\mathbf{r}^{-f}) = \int_{\boldsymbol{\theta}_{test}} p(\boldsymbol{\theta}_{test}|\mathbf{r}^{-f}, \boldsymbol{\alpha}) \sum_{k=1}^{K} p((y_n, w_n)|\widehat{\boldsymbol{\phi}}_k) p(Z = z_k|\boldsymbol{\theta}_{test}) \tag{5.5}$$

where $\widehat{\boldsymbol{\Phi}} = \{\widehat{\boldsymbol{\phi}}_k\}_{\forall k \in \{1,\dots,K\}}$ are the language distributions estimated on $R_{train}$. Here again, we use [**?** ] and [**?** ] to estimate the behaviors distribution $\widehat{\boldsymbol{\theta}}_{test}$ of the record $\mathbf{r}^{-f}$.

We have shown how each of the probabilistic models proceed to attribute likelihood to unobserved values from new records. Then, the predicted value $v$ given by those models is simply the value $v$ belonging to $f$ that has the biggest likelihood. In the next section, we discuss how matrix factorization models proceed to do the same task.

## 5.3 Space Projection

We recall that both $SVD$ and $LCBMF$ express the smartphone logs matrix as a product between matrixes. In both of the models, there is a resulting matrix that represents the behaviors expressed by the data ($\boldsymbol{U}$ for $SVD$, $\boldsymbol{A}$ for $LCBMF$) and another that expresses the concentration of behaviors in each record ($\boldsymbol{V}$ for $SVD$, $\boldsymbol{B}$ for $LCBMF$). Moreover, both $SVD$ and $LCBMF$ represent a record $\mathbf{r}$ as a vector of the size of the language (i.e number of possible realizations) where each dimension contains the number of times a given realization was observed. In this context, a record $\mathbf{r}$ from which all the

realizations of the feature $f$ were removed is represented as a vector $\mathbf{r}^{-f}$ that contains 0 for the realizations of feature $f$ and the same values as $\mathbf{r}$ elsewhere.
When having a new record $\mathbf{r}^{-f}$, the idea for the two models is to map $\mathbf{r}^{-f}$ into the space of the transformation (to express the concentration of behaviors in $\mathbf{r}^{-f}$) and then project it back to the original space (using the realizations importance in the different behaviors) to obtain an approximation $\widehat{\mathbf{r}}$ of $\mathbf{r}$. Intuitively, the values in $\widehat{\mathbf{r}}$ would represent the values that the model would have guessed for each of the possible realizations. Thus, Having $\widehat{\mathbf{r}}$, a natural choice to make is to guess the value $v$ belonging to the feature $f$ that has the highest estimation in the approximation $\widehat{\mathbf{r}}$ (i.e the value that the model would have attributed the maximum number of realizations from all the values of $f$).

For $SVD$, we recall that it expresses the data $\boldsymbol{X}$ as $\boldsymbol{X} \simeq \boldsymbol{USV}$. $\widehat{\mathbf{r}}$ is obtained from $\mathbf{r}^{(-f)}$ by doing the following:

1. compute the $tf\_idf$ of $\mathbf{r}^{(-f)}$, $\mathbf{r}^{(-f)}_{tf\_idf} = tf\_idf(\mathbf{r}^{(-f)})$.
2. map $\mathbf{r}^{(-f)}_{tf\_idf}$ into matrix $V$ $\boldsymbol{v}_{\boldsymbol{r}_{tf\_idf}} = \boldsymbol{U}^{-1}\boldsymbol{S}^{-1}\boldsymbol{r}^{(-f)}_{tf\_idf} = \boldsymbol{U}^{t}\boldsymbol{S}^{-1}\boldsymbol{r}^{(-f)}_{tf\_idf}$. Here we use the fact that $\boldsymbol{U}$ is an orthogonal matrix and thus $\boldsymbol{U}^{-1} = \boldsymbol{U}^{t}$. $\boldsymbol{v}_{\boldsymbol{r}_{tf\_idf}}$ is a vector that contains the behaviors concentration of record $\mathbf{r}_{tf\_idf}$.
3. project back $\boldsymbol{v}_{\boldsymbol{r}_{tf\_idf}}$ to the original space to obtain an approximation of $\mathbf{r}_{tf\_idf}$, $\widehat{\mathbf{r}}_{tf\_idf} = \boldsymbol{USv}_{\boldsymbol{r}_{tf\_idf}}$.
4. do the inverse $tf\_idf$ transformation to obtain an approximation of $\mathbf{r}$, $\widehat{\mathbf{r}} = tf\_idf^{-1}(\widehat{\mathbf{r}}_{tf\_idf})$

Concerning $LCBMF$ it expresses the data $\boldsymbol{X}$ as $\boldsymbol{X} \simeq AB$. $\widehat{\mathbf{r}}$ is obtained from $\mathbf{r}^{-f}$ by doing the following:

1. map $\mathbf{r}$ into matrix $\boldsymbol{B}$ $\boldsymbol{b_r} = \boldsymbol{A}^{-1}\boldsymbol{r}^{(-f)}$. Note that matrix $\boldsymbol{A}$ is not invertible by definiton, however there exists methods to compute the pseudo inverse at the right of a non invertible matrix. $\boldsymbol{b_r}$ is a vector that contains the behaviors concentration of record $\mathbf{r}$.
2. project back $\boldsymbol{b_r}$ to the original space to obtain an approximation of $\widehat{\mathbf{r}} = \boldsymbol{A}b_r$.

# Chapter 6

# Results and Discussion

So far, we have developed an advanced probabilistic model that is able to infer user's behavior from smartphone logs. We exposed in details its nice properties and the theoretical advantages he has with respect to the other probabilistics models. To evaluate its performance in completing this task, we decide to confront it to state of the art current methods for doing the same job. To have a complete rigorously overview, we considered different methods relying on different approaches. In particular, we considered an advanced new approach of matrix factorization that was used by recent researches ($LCBMF$) and that was shown to perform well. Moreover, we considered the state of the art probabilistic approach in hidden class modeling ($LDA$) that showed impressive results since years in representing latent clusters from an observable data.

We also discussed in details meaningfull metrics that can be used to objectively compare the performance of these models.

In this chapter, we introduce the dataset used to test these models. Then, we present the results obtained by the different models($GLMR$, $LMR$, $LCBMF$, $SVD$).

## 6.1 Presenting the Dataset

We test our models on the smartphone logs of five different users. Those smartphone logs belong to five Sony employees in Tokyo and were recorded using an internal Sony software.

Those logs contain the records of the users during several months of observation (seven months for the newest user and ten months for the oldest). The table **??** presents the duration of the observation period of the different users. Each record represents one hour of the period of the observation.

TABLE 6.1: Period of observation for each user in days.

| | User 1 | User 2 | User 3 | User 4 | User 5 |
|---|---|---|---|---|---|
| Period | 300 | 231 | 229 | 249 | 224 |

Those logs contain the following features :

1. *Activity* that represents the activity the user is doing. It takes the values *in_bicycle*, *in_vehicle*, *on_foot*, *tilting* and *still*.
2. *ApplicationLaunch* that represents the smartphone apps launched by the users.
3. *Bluetoothpaired* represents the bluetooth devices paired with the smartphone of the user.
4. *Day* represents the day of the week of the record.
5. $Hour$ represents the time of the day of the record.
6. *Location* represents the location of the users.
7. $Notification$ represents the notifications received by the user.

## 6.2 Feature prediction results

To test the different model, we divide the dataset into 80 % for training and 20 % for testing. We choose to make predictions on the *Location* feature. Indeed, one can argue that many individual behaviors of a human are correlated to the location he is in. For this reason, choosing the *Location* feature as a target feature seens to be a natural choice. This tests the ability of the different models to guess the location in which the user is from the context described by his activity, the application launched, the notifications received, the day of the week, the time of the day and bluetooth devices paired to the smartphone. In this test target categories are $\{mostfrequentlocation, secondmostfrequentlocation, other\}$. Note that for most of the users, either $Home$ is the most frequent location and $Work$ the second most frequent location or the opposite.
For each model, we run the prediction tests for the 5 users and then we average the prediction accuracies results (accuracy and average accuracy) between the 5 users. For each user, the predicted location is a good prediction if it belongs to the right category (and bad prediction otherwise).

Figures **??** and **??** shows, respectively, the accuracy and average accuracy prediction results of the five models ($GLMR$, $LMR$, $LDA$, $LCBMF$ and $SVD$) tested for different number of behaviors (i.e hidden classes).

FIGURE 6.1: Accuracy results of the prediction of the five models.

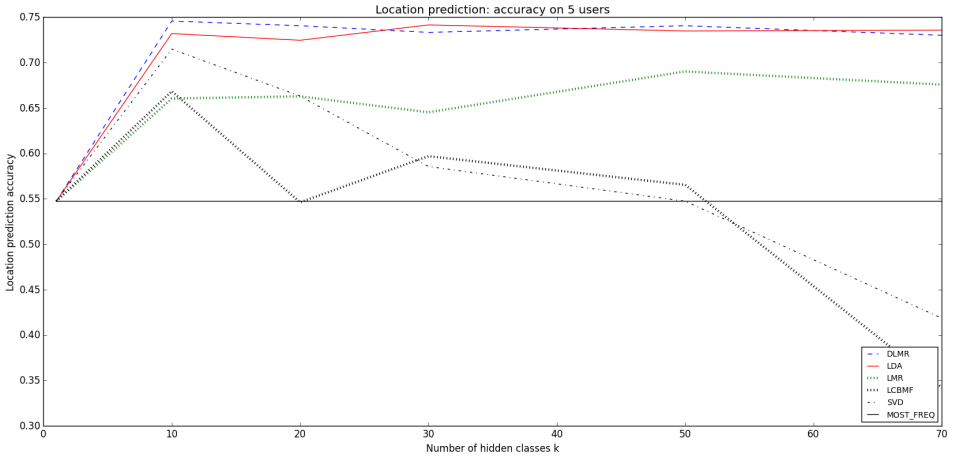
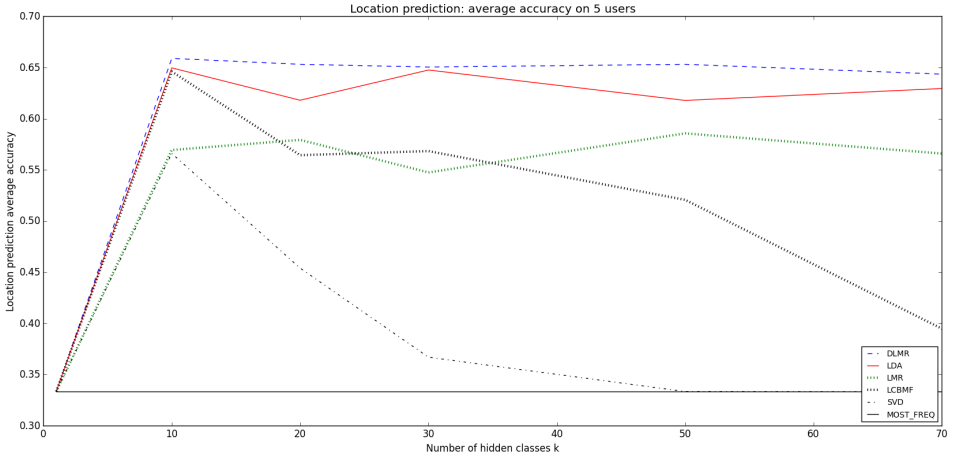



FIGURE 6.2: Average accuracy results of the prediction of the five models.




First, we can see that the three probabilistic models, $GLMR$, $LDA$ and $LMR$ perform widely better than the matrix factorization models $LCBMF$ and $SVD$.
Indeed, when the number of behaviors increases ($K \geqslant 10$) the performance of $SVD$ and $LCBMF$ decreases heavily whereas $DLMR$ $LDA$ and $LMR$ stay relatively stable. This might be due to an effect of overfitting.

Deepen our observation int the three probabilistic models, we can see that $GLMR$ largely outperforms $LMR$. Indeed, the prediction scores og $GLMR$ are about 10% better than $LMR$ for both average accuracy and accuracy for all the number of behaviors K. This confirms the intuition developed in chapter 3. While $LMR$ learns the behaviors expressed specifically by the records seen, $DLMR$ tries to catch a more general structure of the corpus by learning how the seen records generate the behaviors. This allows $DLMR$ to describe better an unseen data. Moreover, while both models show good stability to high number of behaviors, we can observe that for $K \geqslant 50$, $LMR$ score start to decrease while $GLMR$ is still enough stable for $K = 70$. The average accuracy plot shows this effect (average accuarcy of $LMR$ decreased by 3% between $K = 50$ and $K = 70$ while average accuracy of $GLMR$ decreased only by 1 % between $K = 50$ and $K = 70$, which is more due to the randomness of the test and train set rather than to overfitting). The same effect is present in the accuracies plots.
As explained in chapter 3, $GLMR$ has $K + I$ ($I$ is the size of the language and equals $\sum_{f=1}^{J} I_f$) parameters to estimate while $LMR$ has $KM + KV$ parameters. Thus, when K increases the number of parameters to estimate increases much faster in $LMR$ than in $GLMR$. This is what might explain the observed effect in the sense that the big number of parameters in $LMR$ caused overfitting.

Concerning $LDA$, we see that it is also strong to overfitting (and stronger than $LMR$) and accurate in predictions. In fact, $LDA$ underlies the same properties than $GLMR$ (learn how records generate behaviors, small number of parameters to estimate). This is what explains its performance. However, $GLMR$ performs better than $LDA$ in prediction scores. Indeed, while they perform very close, $GLMR$ performs better than $LDA$ in average accuracy while keeping a similar score (or even slightly better score) to $LDAinaccuracy$. This means that $GLMR$ is able to guess more less frequent categories without loosing in general accuracy. This implies that $GLMR$ is better in detecting the rare events or contexts. This is explained by the structure of the model of $GLMR$ in chapter 3. $GLMR$ imposes the probabilities of the values belonging to the same feature to sum to 1. This represents a much more realistic approximation of the real life than $LDA$ that spreads the probabilities over the space of all possible realizations. While frequent contexts (or events) can be outlined by an acceptable model of the real life, detecting rare contexts requires a much more precise representation.
Finally, evaluating $GLMR$ performance independently, we see that it performs around 74% of good guesses and 67% of average good guesses per class. This is a good prediction performance that shows that $GLMR$ is able to predict the future actions that the user might take. This implies that $GLMR$ was able to represent and learn the general behaviors according to which a user behaves.

As a sum up, the results show that the probabilistic models developed perform better in guessing right behaviors of users than the matrix factorization models exposed. Comparing the three probabilistic models $GLMR$, $LMR$ and $LDA$, $GLMR$ shows better results than both $LMR$ and $LDA$. This confirms the intuitions developed in chapter 3 that led us to the $GLMR$ model. Finally, the accuracies rate of $GLMR$ shows good generalization performance of unseen data, and thus proves that $GLMR$ is able to discover user behaviors from their logs.

## 6.3 Feature prediction results

We recall that the final goal that drives our work is to discover the particular behaviors of a user from his smartphone logs. In this concluding section, we close the circle by showing examples of behaviors of the 5 users discovered by $DLMR$.

First of all, and as one might expect, the 5 users exhibit behavior that are similar. Indeed, there is at least one behavior for each user representing the fact that the user works in the week days during the day. There also others expressing the fact that the users are more often at home during the week ends, and others indicating that the users are almost always at home during the night (for all the days). An example of each of these 3 behaviors is shown for one of the users in Figure .
However, one could have supposed those behaviors without analyzing smartphone logs. For this reason, they are not the most interesting habits that we aim to discover. We present them here for two reasons. First, it is a way to verify that $DLMR$ was able to discover some a-priori expected behaviors (as if we label users with some behaviors and then check that $DLMR$ is able to discover them). Second, we present them to give an intuition on how exactly the behaviors are represented in $DLMR$.

Now, we pay attention to examples of behaviors that are more interesting in the sense that they describe a specific user's habit. For example, the behavior in Figure () shows that $user1$ likes to do some readings on Sundays night. Indeed, *sony.drbd.reader.other.jp* (link) is the package name of a Japanese smartphone application for purchasing and reading books. Alternatively, $user1$ would play *monster strike* (link) or browse in the web using his smartphone ($boatbrowser.free$ is the package name of a web browser (link)).
The behavior in Figue () shows that when the $user4$ is not at work during the day (in the week days), he would probably reads some news or watch some TV program. Indeed *gocro.smartnews.android* (link), $sony.nfx.app.sfrc$ (link) and *gunsosy.android* (link) are all news applications while *tsvsdeview.phone* (link) is a smartphone TV app.
Finally, we end our example review with the behavior in Figure (). It shows that $user2$ is often playing ingress while being in *other places* during the week ends (during the

day). In fact, ingress (*nianticproject.ingress* (link)) is an augmented reality playing location based game. In other terms, ingress is a game that transforms the real world in a game map, where players go from one place to the other trying to solve challenges. That's what explains the behavior of $user2$.

All these examples show that $DLMR$ is able to discover user's behaviors from his smartphone logs. In other terms, it is able to decompose a user's life (smartphone logs are a subsample of user's life) as a set of behaviors that are driving his life. Moreover, it is able to do thus by extracting both general behaviors (as sleeping at home, working in the day) and by more specific habits (reading on Sundays night, playing ingress on the week end).

Let's take a step back and recall the initial wish that led us to try to discover users' habbits. We wanted to enable smartphones to build a personal relationship with their owner by learning to know their habits and reacting to their specific needs. Thus we conclude this part by showing how the examples discussed above could be turned out to that end. For example, the smartphone of $user1$ could suggests to him new trendy books to read on Sundays night (even if $user1$ was not planning to read, maybe because he is bored with the books he is currently reading), or even proposes new games that he could play. Concerning $user2$, if his smartphone knows that it will be rainy next Sunday, then it could alert it's owner and suggests him to schedule his ingress session in another day if he is planning to play. Those are only few examples of possible applications and are far from being exhaustive. The work developed in this thesis does not already enable smartphones to act as described (smartphones need to be able to interpret the behaviors they extracted to do so), but allowing them to catch users' behaviors is a necessary and important step towards that direction.

# Chapter 7

# Conclusion

In this work, we have described *Dirichlet Latent Multimodal Representation* ($DLMR$), a flexible generative probabilistic model for multimodal data (i.e data containing multiple types). $DLMR$ treats the different types of data separately by representing them in different distributions and in the same time combine them to underline common meaning expressed by these different types. We applied this model to smartphone logs dataset to extract behaviors of individual users from their logs. In this work, our goal was to develop a model that enables extracting user's behaviors and habits from his smartphone logs (and to see how well this can be done).
To that end we developed a model that can specifically fit a multimodal data (i.e a data that contains multiple types) by treating each type separately and in the same time combining the different types to discover common relations expressed by the data. We called this model the *Dirichlet Latent Multimodal Representation* ($DLMR$).
We compared $DLMR$ to a complete overview of state of the art methods that could accomplish the same task. In this sense, we both considered matrix factorization approaches and probabilistic modeling approaches.
The results obtained show that $DLMR$ performs better than these models and in particular better than *Latent Dirichlet Allocation* ($LDA$) and *Linearly Constrained Bayesian Matrix Factorization* ($LCBMF$). Moreover, $DLMR$ shows good performances in discovering both general habits and specific behaviors of users based on their smartphones logs.

During this work, the need of developing $DLMR$ came from the specificity of the dataset we are considering. Indeed, while existing models are built to fit homogenous datasets containing only one type of information, $DLMR$ came from the need of modeling the heterogeneity of the smartphone logs dataset.
However, smartphone logs is not the only kind of dataset that has these characteristics.

On the contrary, with the exponential increase of applications and platforms accessing to web, the rise of objects connected to the internet (sensor equipped cars, gym equipment machines) and the rapid growth of wearable devices (smart watch, smart glasses) and mobile mini-computers, datasets contains more and more multiple types of data. Because it was built by essence to model the multimodality of data, $DLMR$ can be applied to all these kind of datasets because and can be used for different purposes. It can for example be used to give a meaning to a non intuitive dataset, to predict future events, to classify data elements or to compress the size of a large dataset.