Chapter 1

Dirichlet Latent Multimodal Representation (DLMR)

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.

1.1 Latent Multimodal Representation (LMR)

The first model we introduce is the Latent Multimodal Representation (LMR). As the title implies it, this model is used to model hidden (i.e unobserved) classes for a data that contains mixed types (i.e multimodal data). 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 LMR model. Second, to better understand the utility and the intuitions that lead us to build the LMR 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).

1.1.1 LMR model

Let's consider the corpus representation of the smartphone logs. Smartphone logs are represented by a corpus 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 $\{z_1, ..., z_K\}$.
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 1st 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 when 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 when 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 on 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 are two different kinds of features. On the one hand, there are features that take values during all the time range of the observation. Examples of 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 are features that take values only in certain points in the time range of the observation. Examples are "$Application\_launches$", "$Notifications$" 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 $\mathbf{r}_m, m \in \{1, ..., M\}$ of the corpus R accordingly by adding the realization $(f, v_{non\_present})$ to each record that do 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:

$$\begin{aligned} p([(y_1, w_1), ..., (y_N, w_N)]|\mathbf{r}) &= \prod_{n=1}^{N} p((y_n, w_n)|\mathbf{r}) \\ &= \prod_{n=1}^{N} \sum_{k=1}^{K} p(w_n|Z = z_k, F = y_n) p(Z = z_k|\mathbf{r}) \end{aligned} \tag{1.1}$$

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

$$\begin{aligned} p([(y_1, w_1), ..., (y_N, w_N)], \mathbf{r}) &= p([(y_1, w_1), ..., (y_N, w_N)]|\mathbf{r}) p(\mathbf{r}) \\ &= \frac{1}{M} \prod_{n=1}^{N} \sum_{k=1}^{K} p(w_n|Z = z_k, F = y_n) p(Z = z_k|\mathbf{r}) \end{aligned} \tag{1.2}$$

Where M represents the number of the records in the corpus.
Using Eq. (1.1) and Eq. (1.2), we can express the probability of a corpus $R =$

$\{\mathbf{r}_1, ..., \mathbf{r}_M\}$ as the product of the probabilities of each record $\mathbf{r}_m$:

$$p(R) = p(\{\mathbf{r}_1, ..., \mathbf{r}_M\}) = \prod_{m=1}^{M} p([(y_1, w_1), ..., (y_N, w_N)]|\mathbf{r}) \tag{1.3}$$

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. (1.1):

$$L(R) = p(R) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} \sum_{k=1}^{K} p(w_n|Z = z_k, F = y_n)p(Z = z_k|\mathbf{r}) \tag{1.4}$$

LMR 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 this 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 LMR, we assume that in each time frame, a user has a mixture of behaviors. When he is involved in some behaviors, he selects actions 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 would act following LMR generation process in some Saturday morning according to the LMR 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 "*do_gym_on_Saturday*". In, the behavior "*listening_to_music_in_car*", 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_1) 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 that would be generated by Bob following the generation model of LMR describes well Bob's life in some Saturday morning.

This example concludes the model description of LMR. In the next part, we establish a relation between LMR 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 LMR, that enables him to fit well the problem of modeling smartphone logs.

1.1.2 Relationship between LMR 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 when 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 when 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 LMR. 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, realizations are independent between each other, and a realization is independent from the record it appears in conditioned in the behavior.

When given hidden class z_k, LMR 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 LMR. 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 either 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 its values allows LMR 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, thhis is the main difference between LMR 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, LMR is a much more suited model that $pLSI$ for representing datasets containing mixed types of features.

1.2 Generative LMR (DLMR) model

While LMR has nice properties on mixed data types, it still has some imperfections. In this section, we introduce the Dirichlet Latent Multimodal Representation ($DLMR$) which is an improved version of LMR in the sense that it addresses the weaknesses of LMR that we are going to discuss.

1.2.1 Prior distributions

To see how to proceed beyond LMR, let's first have a look at some of it's weaknesses. First, it is important to note that LMR is incomplete in that it provides no probabilistic model at the level of records. In LMR, 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 $DLMR$, 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 (the simplex) defined as follows:

$$\begin{cases} 0 \leqslant p(z_k) \leqslant 1, \forall k \in \{1, ..., K\} \\ \sum_{k=1}^{K} p(z_k) = 1 \end{cases} \tag{1.5}$$

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 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 has 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, $DLMR$ 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 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 $DLMR$, we are ready to describe the generative process of $DLMR$.

1.2.2 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 $\mathbf{r}_1$ 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 $N_f \geq 1, \forall f \in F$).
$DLMR$ 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 $\boldsymbol{\theta}_1$ for record $\mathbf{r}$ where $\boldsymbol{\theta}_1$ is a random variable vector following a Dirichlet distribution: $\boldsymbol{\theta}_1 \sim Dir(\boldsymbol{\alpha})$
   (b) generate the values coming from the vocabulary of feature 1 by doing the following:
      i. generate the 1st value w_1 by doing the following:
         A. choose one behavior $z_k, k \in [1, K]$ with probability $p(z_k|\boldsymbol{\theta}_1)$, where Z is a random variable that follows a multinomial distribution when conditioned on the behavior distribution $\boldsymbol{\theta}_1$
         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 when conditioned on the behavior and the feature.
      ii. repeat the process 2(b)i to generate the values $w_2, ..., w_{N_1}$
   (c) repeat the process 2b to generate the values of the features $2, ..., J$.
3. repeat the process 2 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 $DLMR$ starts 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 $\mathbf{r}_m$, $DLMR$ generates a behavior distribution $\boldsymbol{\theta}_m$ for $\mathbf{r}_m$. Starting from that point, the generation process of $DLMR$ 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 M sets of behavior distributions $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,...,M\}}$ (one for each record $\mathbf{r}_m$). For $DLMR$, 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 ($DLMR$).

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, $L(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})$ equals 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.

$$\begin{aligned} L(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F}) &= p(R|\boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F}) \\ &= \prod_{k=1}^{K} \prod_{f=1}^{J} \int_{\boldsymbol{\phi}_{f,k}} p(\boldsymbol{\phi}_{f,k}|\boldsymbol{\beta}_f) L(R|\boldsymbol{\alpha}, \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \end{aligned} \tag{1.6}$$

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 $\mathbf{r}_m$.

$$\begin{aligned} L(R|\boldsymbol{\alpha}, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) &= p(\{\mathbf{r}_1, ..., \mathbf{r}_M\}|\boldsymbol{\alpha}, \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \\ &= \prod_{m=1}^{M} p(\mathbf{r}_m|\boldsymbol{\alpha}, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \end{aligned} \tag{1.7}$$

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.

$$p(\mathbf{r}_m|\boldsymbol{\alpha}, \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) = \int_{\boldsymbol{\theta}_m} p(\boldsymbol{\theta}_m|\boldsymbol{\alpha}) p(\mathbf{r}_m|\boldsymbol{\theta}_m, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \tag{1.8}$$

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.

$$\begin{aligned} p(\mathbf{r}_m|\boldsymbol{\theta}_m, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) &= \prod_{n=1}^{N_m} p((y_n, w_n)|\boldsymbol{\theta}_m, \{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,...,K\}}) \\ &= \prod_{n=1}^{N_m} \sum_{k=1}^{K} p(w_n|\boldsymbol{\phi}_{y_n,k}) p(Z = z_k|\boldsymbol{\theta}_m) \end{aligned} \tag{1.9}$$

This ends our definition of $DLMR$ model. We assume that the smartphone logs we observe were generated by $DLMR$. As for LMR, 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 only the data is observable. 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 $DLMR$, 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 $DLMR$ 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 LMR).

We conclude this section by exposing other nice properties and advantages of $DLMR$ compared to LMR. 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 a unique time frame.
The prior distribution over the distribution of values in a behavior allows to smooth the model. Indeed, while LMR would attribute a 0 probability to a new record containing a value never seen in observed records, $DLMR$ 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 $DLMR$ model, the nice properties it comes with and the gain it brings with respect to LMR, we close the circle n the next section by establishing relationships between $DLMR$, LMR, *pLSI* and *LDA* (*Latent Dirichlet Allocation*) [**?**].

1.3 Relationship between DLMR, LMR, pLSI and Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (*LDA*) was introduced by D. M. Blei and al. [**?**] 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 $\boldsymbol{\theta}_1$ for record $\mathbf{r}$ where $\boldsymbol{\theta}_1$ is a random variable vector following a Dirichlet distribution: $\boldsymbol{\theta}_1 \sim Dir(\boldsymbol{\alpha})$
   (b) generate (y_1, w_1) by doing the following:
      i. choose one behavior $z_k, k \in [1, K]$ with probability $p(z_k|\boldsymbol{\theta}_1)$, where Z is a random variable that follows a multinomial distribution conditioned on the behavior distribution $\boldsymbol{\theta}_1$
      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 2b to generate the values $(y_2, w_2), ..., (y_N, w_N)$
3. repeat the process 2 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 $DLMR$ over LMR. 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 $DLMR$, the probabilities of values are spread only over the dictionary defined by their feature and the distributions of the different dictionaries are independent. This enables $DLMR$ to model the *concurrent* and *competing* concepts. In other terms, $DLMR$ combines the advantages of LDA over $pLSI$ and LMR over $pLSI$. It is a fully generative model that is suited for data containing mixed types.

1.4 DLMR inference and parameter estimation

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

1.4.1 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 $DLMR$, the exact distributions over behaviors $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}}$ of the records $\{\mathbf{r}_m\}_{\forall m \in \{1,\ldots,M\}}$ are not known. Moreover, the distribution of the vocabulary of each feature $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\ldots,K\}}$ in the behaviors $\{z_k\}_{\forall k \in \{1,\ldots,K\}}$ 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 $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}}$ and $\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\ldots,K\}}$.

We simplify the notations by setting $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}}$ and $\boldsymbol{\Phi} = \{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\ldots,K\}}$. 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 $\boldsymbol{\Phi}$:

$$p(\boldsymbol{\Theta}, \boldsymbol{\Phi} | R, \boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F}) = \frac{p(\boldsymbol{\Theta}, \boldsymbol{\Phi}, R | \boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})}{p(R | \boldsymbol{\alpha}, \{\boldsymbol{\beta}_f\}_{\forall f \in F})} \tag{1.10}$$

Good estimates for $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ could be for example the estimates that maximize (1.10) or the expected value of (1.10). In all the cases (1.10) needs to be computed. Unfortunately, because of the normalization over $L(R)$, (1.10) 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 1.3 that LDA and $DLMR$ are very much related. This is also the case for the inference process using Gibbs sampling of both LDA and $DLMR$. 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 $DLMR$.

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 estimates $\mathbf{\Theta}$ 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:

$$p(c_s|\mathbf{c}_{-s}, \mathbf{R}) \sim \frac{n_{-s,k}^{(y_s,w_s)} + \beta_{y_s,w_s}}{n_{-s,k}^{(y_s,.)} + \sum_{v=1}^{I_{y_s}} \beta_{y_s,v}} . \frac{n_{-s,k}^{r_m} + \alpha_k}{n_{-s,.}^{r_m} + \sum_{h=1}^{K} \alpha_h} \tag{1.11}$$

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* [**?**], *DLMR* normalizes Eq. (1.11) 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. (1.11) 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, *DLMR* and *LDA* assume exactly the same probabilistic model at the record level). For each record $\mathbf{r}_m$ and behavior z_k, we have:

$$\widehat{\theta}_{m,k} = p(z_k|\mathbf{r}_m) = \frac{n_k^{r_m} + \alpha_k}{n_k^{(.)} + \sum_{h=1}^{K} \alpha_h} \tag{1.12}$$

$\boldsymbol{\Phi}$ can be derived in a similar manner that for *LDA* [**?**]. The only difference is that in *DLMR*, 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:

$$\widehat{\phi}_{f,k,v} = p(v|z_k, f) = \frac{n_k^{(f,v)} + \beta_{f,v}}{n_k^{(f,.)} + \sum_{v=1}^{I_f} \beta_{f,v}} \tag{1.13}$$

where n in both Eq. (1.12) and (1.13) is defined as in Eq. (1.11).

1.4.2 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 1.2.1, we suppose that the observed smartphone logs were generated by a $DLMR$ 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 $\boldsymbol{\alpha}$ is the concentration parameter to a Dirichlet distribution, from which the multinomial probability vectors $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}}$ are generated. Thus, $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}}$ are all conditionally independent given $\boldsymbol{\alpha}$, and $\boldsymbol{\alpha}$ is conditionally independent from all the other variables given $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}}$. The same applies for $\{\boldsymbol{\beta}_f\}_{\forall f \in F}$ and $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\ldots,K\}}\}$. This means that if $\{\boldsymbol{\theta}_m\}_{\forall m \in \{1,\ldots,M\}\}}$ and $\{\{\boldsymbol{\phi}_{f,k}\}_{\forall f \in F, \forall k \in \{1,\ldots,K\}}\}$ 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:

$$\alpha_k^{new} = \alpha_k \frac{\sum_{m=1}^{M}(\Psi(n_k^m + \alpha_k)) - \Psi(\alpha_k))}{\sum_{m=1}^{M}(\Psi(n_{(.)}^m + \sum_{k=1}^{K}\alpha_k) - \Psi(\sum_{k=1}^{K}\alpha_k))} \tag{1.14}$$

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..
- Ψ is the digamma function.

Using equation Eq. (1.14), 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:

$$\alpha_k^{new} = \alpha_k \frac{\sum_{m=1}^{M}(\Psi(n_k^{r_m}) - \Psi(\alpha_k))}{\sum_{m=1}^{M}(\Psi(n_{(.)}^{r_m} + \sum_{k=1}^{K}\alpha_k) - \Psi(\sum_{k=1}^{K}\alpha_k))} \tag{1.15}$$

- $n_k^{r_m}$ 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).
- $n_{(.)}^{r_m}$ represents the number of realizations from the record $\mathbf{r}_m$ that have an assignment in the assignments vector $\mathbf{c}$ (i.e $n_{.}^{r_m}$=N_m).

$$\beta_{f,v}^{new} = \beta_{f,v} \frac{\sum_{k=1}^{K}(\Psi(n_k^{(f,v)} + \beta_{f,v}) - \Psi(\beta_{f,v}))}{\sum_{k=1}^{K}(\Psi(n_{(k)}^{(f,.)} + \sum_{u=1}^{I_f}\beta_{f,u}) - \Psi(\sum_{u=1}^{I_f}\beta_{f,u}))} \tag{1.16}$$

- $n_k^{(f,v)}$ 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 *DLMR*. 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 of *DLMR* from the observed data.