ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Learning &
Adaptive Systems

Master Thesis

Spatial Summarization of Image Collections

Spring Term 2016

This dissertation is submitted for the degree of *Master of Science ETH in Computer Science*

Supervised by:
Prof. Dr. Andreas Krause
Dr. Sebastian Tschiatschek
Alkis Gotovos, M.Sc.

Author:
Diego Alfonso Ballesteros Villamizar

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

List of Figures

List of Tables

Preface

Acknowledgments and such …

Abstract

The Abstract …

Symbols

Symbols

η	Learning rate
γ	Model parameter
κ	Largest subset size
ν	Noise-to-data ratio
σ	Gaussian distribution's standard deviation
$\mathcal{D}$	Set of data samples
K	Latent coherence dimensions
L	Latent diversity dimensions
M	Number of features
$\mathcal{N}$	Set of noise samples
$\mathbb{R}$	Real numbers
S, T	Subset
S_t	Sequence
V	Ground set
Z	Normalization constant

Matrices

$\mathbf{B}$	Feature diversity weights
$\mathbf{E}$	Feature coherence weights
$\mathcal{I}$	Identity
$\mathbf{W}^b$	Coherence weights
$\mathbf{W}^e$	Diversity weights
$\mathbf{X}$	Features

Vectors

$\boldsymbol{\theta}$	Model parameters
$\mathbf{a}$	Utility weights for features
$\mathbf{u}$	Utility weights
$\mathbf{G}$	Accumulated gradient

Indices

i, j	Element
k	Feature
c	Coherence dimension
d	Diversity dimension

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
FLID	Facility Location Diversity
FLDC	Facility Location Diversity and Coherence
FFLDC	Featurized Facility Location Diversity and Coherence
MLE	Maximum Likelihood Estimation
NCE	Noise Contrastive Estimation
LAS	Learning & Adaptive Systems
SGD	Stochastic Gradient Descent
TVD	Total Variation Distance

Chapter 1

Introduction

The amount of information available on the internet is staggering, even conducting a simple search for *St Peter's Basilica* can turn up millions of results. Fortunately, search engines ensure the first results are the most relevants and often times all we need to look at. However, this may not be true for all types of searches.

Imagine a person planning their a trip to Rome for two days. Where to start? Suppose they start searching for attractions to visit or things to do. A popular site for this kind of search is *tripadvisor* which shows over 2000 attractions and activities along with user reviews and recommendations, clearly it is impossible for someone to go through all these results so picking the top results and filtering for some type of activities is a good strategy. Afterwards, they must find the location of each place and organize them in a schedule considering travel times and visit times, all to fit many attractions in a short time. This certainly sounds like a complicated endeavor.

An alternative would be to find a curated itinerary that contains all the places to visit and in which order. There are many for Rome since it is such a touristic city, but what about smaller cities? Could it be possible to produce such itineraries in an automatic fashion, such that it can be scaled to multiple cities?

This is a complex problem with various questions to answer. How to obtain data about a city that is available regardless of size or popularity? How to extract meaningful information about locations and activities from such data? How to select a subset of those locations and activities that are of interest for a particular user and present it in an organized manner?

We focus on a particular use case for this problem. Consider a user that is planning a single day trip and maybe knows some attractions that they definitely want to visit, however they would like to see more than that. Therefore, a method is needed that can complete or propose a set of attractions to be visited in a city, this should be generated automatically from data so that it can be applied to different cities. This addresses all the questions presented before although it excludes the problems of scheduling based on travel and visit times, which could be done as a post-processing step given the sets of locations.

Photographies of a place are usually an important factor when selecting destinations, and it can influence the perception that a visitor has of a place. When a tourist takes a photo of some location and uploads it to the internet, they are giving it an implicit value (Donaire et al., 2014). Several authors have used internet photo collections for the task of tourist route planning and recommendation. This is why we chose photos as the starting point for identifying popular attractional, in particular geo-tagged photos are of interest because they can be directly associated with a place.

One source for these photos is Flickr, a popular site where users can upload and manage their photos for free. Having over 92 million users among 63 countries makes Flickr an excellent source of photos for many locations around the world, it was reported back in

2014 that around 1 million photos were shared every day [1]. Many of the photos in Flickr are public and available through an API which allows researchers to collect and analyze this data.
The next step after data collection is to translate it into information and this is related to the problem of summarization. Although more commonly addressed in the context of documents, summarization techniques aim to extract information from large scale datasets and present it in a condensed manner that can be easily understood by a user. One particular kind of model that has been used recently are submodular set functions, which are a natural fit for modeling diversity, an often desired property of summaries. This work studies a submodular model known as FLID proposed by Tschiatschek et al. (2016) which was used for image collection summarization and product recommendation from Amazon data, we extend it and show how this can be also applied for the problem of recommending routes for tourists.
The contributions of this work are:

- Proposing a super-modular extension to the FLID model to improve its ability for estimating probability distributions.
- Proposing an extension to the FLID model able to generalize through feature representations.
- Exploring the behavior of NCE learning for the FLID model and its extensions, and the effect of different parameters in the learning results.
- Presenting an application of the proposed models to solve a recommendation task for touristic locations in a city.

The remainder chapters are organized as follows, Chapter 2 describes the existing literature and methods for the problems of summarization of image collections and tourist route recommendation based on geotagged photos, as well as other applications of submodular models. Chapter 3 explains in detail the existing models and techniques used throughout this work. Chapter 4 introduces the proposed models and their characteristics, then Chapter 5 shows how they can be learned from data and the performance of the learning method. Chapter 6 describes the methods used for the task of recommending tourist locations and how the proposed models were applied to real data, the results of the experiments are then presented and discussed in Chapter 7. Finally, Chapter 8 summarizes this work and presents some conclusions and future directions.

[1] http://techcrunch.com/2014/02/10/flickr-at-10-1m-photos-shared-per-day-170-increase-since-making-1tb-free/

Chapter 2

Related Work

2.1 Probabilistic Sub/super-modular models

Talk about the importance of these models and the recent interest in them, e.g. DPPs, Sampling.

2.2 Image collection summarization

Generally about the problem and the different approaches.

2.3 Tourist routes recommendation

The initial papers I found on this, basically Markov chains and clustering.

Chapter 3

Background

This chapter presents the supporting concepts and methods that will be used throughout this work.

3.1 Submodular Set Functions

Functions over sets are of great interest and can be applied to diverse settings. In particular, submodular set functions have been utilized in multiple domains, such as document summarization and information gathering (Krause and Golovin, 2014).
A set function is defined on the powerset 2^V of a ground set V, i.e. $F : 2^V \rightarrow \mathbb{R}$ and it is submodular if it exhibits a "diminishing returns" property (Krause and Golovin, 2014), namely if it satisfies:

$$\forall S, T \subseteq V : S \subseteq T, i \notin T \mid F(S \cup i) - F(S) \geq F(T \cup i) - F(T) \tag{3.1}$$

Intuitively this indicates that adding an element to a smaller set yields a larger effect, i.e. in terms of the set function F value, than adding it to a larger one. This is a natural property in the context of summarization where adding more information to a large summary can be less effective than adding it to a smaller one. This property also makes them good candidates for modeling the concept of diversity (Tschiatschek et al., 2016).
Analogous to the use of submodular functions to model diversity, supermodular functions can be used to model complementarity or coherence between elements.
A set function is supermodular if it satisfies the condition in (3.1) with the inequality sign reversed, i.e. a set function $F(S)$ is supermodular iff $-F(S)$ is submodular. Supermodularity is used extensively in economics to model complementarity between strategies in games (Amir, 2005).
Finally, if a function F is both submodular and supermodular, i.e. it satisfies condition (3.1) with equality, then it is said to be a modular function. These are the simplest examples of submodular or supermodular functions and are akin to linear functions but in a discrete domain (Krause and Golovin, 2014).

3.2 Probabilistic Log-sub/supermodular Models

Set functions may also express probability distributions, in which case they are known as Probability Set Functions. Formally, a set function F is a probability set function if it satisfies that $\forall S \subseteq V, P : 2^V \rightarrow \mathbb{R} \mid 0 \leq P(S) \leq 1$ and $\sum_{S \subseteq V} P(S) = 1$ (Bruno et al., 1999).
This work is interested in a particular class of probability set functions, namely probabilistic log-submodular and log-supermodular models, which are probabilities $P(S)$ of the form (Djolonga and Krause, 2014):

$$P(S) = \frac{1}{Z} \exp(F(S)) \tag{3.2}$$

Where $F(S)$ is a submodular or supermodular function, respectively. Djolonga and Krause (2014, 2015) show that these models encompass many practical probabilistic models such as repulsive Ising models, Deterministic Point Processes (DPPs) and binary pairwise Markov Random Fields (MRFs).
This model class is of interest because the contributions of this work focus on extensions to a novel probabilistic log-submodular model proposed by Tschiatschek et al. (2016) which will be described in the next section.
The normalization constant Z is also known as the *partition function* and it is necessary to fully determine the normalized model and compute quantities such as marginal probabilities, however its exact computation is known to be #P-complete in general(Jerrum and Sinclair, 1990). This makes it impossible in practice to perform such computations except for special cases of the model.

3.3 Facility Location Diversity Model (FLID)

Every modular function can be written as a sum of individual weights for each element $i \in V$ assuming a normalization such that $F(\emptyset) = 0$ (Krause and Golovin, 2014), i.e.

$$F(S) = \sum_{i \in S} u(i) \tag{3.3}$$

Where $u(i)$ is some function $u : V \to \mathbb{R}$. For convenience, denote $\mathbf{u}$ as the vector of modular weights where $u_i = u(i)$. Therefore any log-modular function can be written as:

$$P(S) \propto \exp\left(\sum_{i \in S} u_i\right) = \prod_{i \in S} \exp(u_i) \tag{3.4}$$

This is the simplest log-submodular probability function and is the modular part for the FLID model proposed by Tschiatschek et al. (2016). In this model, $\mathbf{u}$ can be thought of as modeling the relevance or utility of each element, for example in the context of spatial summarization this utility could be proportional to the popularity of a location or to how many times it has been photographed.
However, these utilities alone can not capture interactions between the elements in a set. To address this, Tschiatschek et al. (2016) propose an additional term that models set diversity, its objective is to characterize redundant elements and assign lower probabilities to sets that contain them.
This term is based on representing each element i with an L-dimensional vector $\mathbf{w}^b \in \mathbb{R}^L_{\geq 0}$, where each dimension $1 \leq d \leq L$ captures a concept related to set diversity and $w^b_{i,d}$ quantifies the relevance of each element i for each concept d (Tschiatschek et al., 2016). Hereby, define $\mathbf{W}^b \in \mathbb{R}^{|V| \times L}_{\geq 0}$ as the matrix where the i-th row is the representation $\mathbf{w}^b$ of element i.
For each dimension d, the diversity of a set S is quantified by $\max_{i \in S} w^b_{i,d} - \sum_{i \in S} w^b_{i,d}$. This assigns a negative value for sets that contain more than one element with nonzero weight $w^b_{i,d}$ in that dimension (Tschiatschek et al., 2016). Equation (3.5) shows the complete term, this sums over all L dimensions to account the diversity in each concept.

$$\mathrm{div}(S) = \sum_{d=1}^{L} \left(\max_{i \in S} w^b_{i,d} - \sum_{i \in S} w^b_{i,d} \right) \tag{3.5}$$

Finally, the complete FLID model proposed by Tschiatschek et al. (2016) combines these two terms and is given by:

Table 3.1: FLID probability distribution for the scenario in Example 3.3.1.

S	$P(S)$
$\{h,s\},\{h,f\}$	≈ 0.41
$\{h\},\{s\},\{f\}$	≈ 0.06
$\{\},\{s,f\},\{h,s,f\}$	≈ 0.00

$$P(S) = \frac{1}{Z}\exp\left(\sum_{i\in S} u_i + \sum_{d=1}^{L}\left(\max_{i\in S} w^b_{i,d} - \sum_{i\in S} w^b_{i,d}\right)\right) \tag{FLID}$$

As mentioned before, the computation of the partition function is generally intractable for log-submodular models. However, for FLID this can be computed in time $O(|V|^{L+1})$ which is polynomial on the size of the ground set and significantly better than the cost of enumerating the powerset of V, i.e. $O(2^{|V|})$ (Tschiatschek et al., 2016). Nevertheless, it is worth noting that this complexity is exponential in L which means that the partition function computation is only pratical for a limited range of FLID models, namely those with $L \ll |V|$.

3.3.1 Example: Two Landmarks

In order to better illustrate the model, consider a town with 3 popular locations: A town hall (h), a statue (s) and a fountain (f). Assume that historic data shows that visitors only take photos at either the town hall and the statue, or at the town hall and the fountain. This can be modeled with FLID by introducing a latent dimension representing some concept that is present in both the statue and the fountain but not in the town hall, e.g. "is not a building".
Concretely, let $V = \{h,s,f\}$ and $\mathbf{u} = (2,2,2)^\intercal$, indicating that all locations are equally popular. A suitable diversity weight matrix would then be $\mathbf{W}^b = (0,20,20)^\intercal$. Table 3.1 shows the resulting probabilities of the subsets, accurately representing the aforementioned problem. Note that the probabilities are not exactly $0.5/0.5$ for the sets of interest but this could be easily fixed by increasing the utilities to ensure that sets of size 2 have a larger unnormalized magnitude compared to individual ones.

3.4 Learning from Data

Much of the interest in log-submodular models is concentrated on performing inference for known functions (Tschiatschek et al., 2016). However, Tschiatschek et al. (2016) explore how to learn such models from data, i.e. estimate the model parameters from observations with an unknown distribution.

3.4.1 Learning Log-modular Models

A special case of log-submodular models where the parameters can be estimated efficiently using Maximum Likelihood Estimation (MLE) is the log-modular model presented in Equation (3.4). For this basic model, it is possible to estimate the utilities $\mathbf{u}$ through the maximum likelihood estimator for the marginals $\hat{P}(i \in S)$. These marginals are given by:

$$P(i \in S) = \frac{1}{1+\exp(-u_i)} \tag{3.6}$$

And its maximum likelihood estimator for a dataset $\mathcal{D}$ is:

$$\hat{P}(i \in S) = \frac{N(\#i \in S)}{|\mathcal{D}|} \tag{3.7}$$

Therefore, the utilities can be estimated as:

$$u_i = -\log\left(\frac{1}{\hat{P}(i \in S)} - 1\right) \tag{3.8}$$

This model also allows computing the partition function in closed form, this is given by:

$$Z = \prod_{i \in S}(1 + \exp(u_i)) \tag{3.9}$$

Because these quantities can be computed in closed form, it is possible to efficiently estimate the parameters of a log-modular model and sample from it. This is useful for producing noise samples and constructing baselines for the experiments.
Even though MLE is the commonly used method for the task of parameter estimation from data, it only works efficiently for normalized models (Gutmann and Hyvärinen, 2012). This makes it intractable for the general class of log-submodular models, and also a wide range of FLID models because the computation of Z is exponential on L. Gutmann and Hyvärinen (2012) propose an alternative method for parameter estimation in unnormalized models known as Noise Contrastive Estimation (NCE) and this is the method used by Tschiatschek et al. (2016) to learn the FLID model.

3.4.2 NCE Learning

The idea behind NCE is to transform the unsupervised learning task of estimating a probability density from data into a supervised classification task. In order to do this, the observed data $\mathcal{D}$, assumed to be drawn from an unknown distribution P_d, is compared to an artificially generated set of noise samples $\mathcal{N}$ drawn from a known distribution P_n that can be efficiently normalized. The classification task is to maximize the likelihood of correctly discriminating each sample as either observed data or artificial noise.
Formally, denote $\mathcal{A}$ as the complete set of labeled samples, i.e. $\mathcal{A} = \{(S, Y_s) : S \in \mathcal{D} \cup \mathcal{N}\}$ where $Y_s = 1$ for $S \in \mathcal{D}$ and $Y_s = 0$ for $S \in \mathcal{N}$. Additionally, let ν be the noise-to-data ratio, i.e. $\nu = {|\mathcal{N}|}/{|\mathcal{D}|}$.
The goal is to estimate the posterior probabilities $P(Y_s = 1 \mid S; \boldsymbol{\theta})$ and $P(Y_s = 0 \mid S; \boldsymbol{\theta})$, in order to discriminate noise from data samples. These probabilities are given by equations (3.10) and (3.11).

$$P(Y_s = 1 \mid S; \boldsymbol{\theta}) = \frac{\hat{P}_d(S; \boldsymbol{\theta})}{\hat{P}_d(S; \boldsymbol{\theta}) + \nu P_n(S)} \tag{3.10}$$

$$P(Y_s = 0 \mid S; \boldsymbol{\theta}) = \frac{\nu P_n(S)}{\hat{P}_d(S; \boldsymbol{\theta}) + \nu P_n(S)} \tag{3.11}$$

It is worth noting that $\hat{P}_d$ is used instead of P_d, because the real density is not known. As indicated by Gutmann and Hyvärinen (2012), $\hat{P}_d$ can be an unnormalized distribution for NCE where the partition function Z is included in the set of parameters $\boldsymbol{\theta}$ as $\hat{Z}$, hence resulting in an approximately normalized distribution after the optimization.
In order to obtain these posterior probabilities, the following conditional log-likelihood objective is maximized (Gutmann and Hyvärinen, 2012):

$$g(\boldsymbol{\theta}) = \sum_{S \in \mathcal{D}} \log P(Y_s = 1 \mid S; \boldsymbol{\theta}) + \sum_{S \in \mathcal{N}} \log P(Y_s = 0 \mid S; \boldsymbol{\theta}) \tag{3.12}$$

This maximization can be performed using a gradient-based optimization method such as Stochastic Gradient Descent (SGD).

Theoretical Considerations

A couple of theoretical conditions are necessary for obtaining good estimates with NCE (Gutmann and Hyvärinen, 2012):

1. The parameterized probability function $\hat{P}_d(S;\theta)$ must be of the same family as the real distribution P_d, i.e. $\exists\boldsymbol{\theta}^* \mid \hat{P}_d(S;\boldsymbol{\theta}^*) = P_d$.
2. The noise distribution P_n is nonzero whenever P_d is nonzero.

Practical Considerations

Additionally, the following statements must be considered when choosing the noise distribution P_n (Gutmann and Hyvärinen, 2012):

1. A distribution that can be sampled easily.
2. Noise that is as similar as possible to the data, otherwise the classification problem could be too easy.
3. A noise sample as large as possible.

3.4.3 Learning FLID via NCE and SGD

Stochastic Gradient Descent was used by Tschiatschek et al. (2016) to learn the FLID model through NCE. SGD is a gradient-based method that has proven effective in large-scale learning tasks due to its efficiency when the computation time is a limiting factor (Bottou, 2010; Zhang, 2004), hence making it appropriate for the scale of data sourced from the internet, such as public user photos in Flickr.
In each iteration, the gradient $\nabla \log P(Y_s = y \mid S;\boldsymbol{\theta})$ must be computed. For FLID, the parameter vector $\boldsymbol{\theta}$ is composed by $\mathbf{u}, \mathbf{W}^b$ and $\hat{Z}$, and the corresponding gradients are given by:

$$\nabla \log P(Y_s = y \mid S;\boldsymbol{\theta}) = \left(y - \frac{1}{1 + \nu \frac{P_n(S)}{\hat{P}_d(S;\boldsymbol{\theta})}}\right) \nabla \log \hat{P}_d(S;\boldsymbol{\theta}) \tag{3.13}$$

$$\hat{P}_d(S;\boldsymbol{\theta}) = \frac{1}{\hat{Z}} P(S;\mathbf{u},\mathbf{W}^b) \tag{3.14}$$

$$\left(\nabla_{\mathbf{u}} \log \hat{P}_d(S;\hat{Z},\mathbf{u},\mathbf{W}^b)\right)_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

$$\left(\nabla_{\mathbf{W}^b} \log \hat{P}_d(S;\hat{Z},\mathbf{u},\mathbf{W}^b)\right)_{i,d} = \begin{cases} -1 & \text{if } i \in S \text{ and } i \neq \operatorname{argmax}_{j \in S} w^b_{j,d} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

$$\nabla_{\hat{Z}} \log \hat{P}_d(S;\hat{Z},\mathbf{u},\mathbf{W}^b) = \frac{-1}{\hat{Z}} \tag{3.17}$$

Where $P(S;\mathbf{u},\mathbf{W}^b)$ is the unnormalized (FLID) equation, $(\nabla_{\mathbf{u}} \cdot)_i$ represents the i-th entry of the gradient with respect to $\mathbf{u}$ and $(\nabla_{\mathbf{W}^b} \cdot)_{i,d}$ represents the (i,d)-th entry of the sub-gradient with respect to $\mathbf{W}^b$ (Tschiatschek et al., 2016).

Computational Complexity

As described by Tschiatschek et al. (2016), the running time requirement for calculating the sub-gradient for FLID is $O(L|S|)$, and for a complete pass over data and noise samples is then $O(|\mathcal{D} \cup \mathcal{N}|\kappa L)$ where $\kappa = \max_{S \in \mathcal{D} \cup \mathcal{N}} |S|$. This shows that learning is efficient as it is only linear on the size of the data and the noise, assuming small values for κ and L.

3.4.4 AdaGrad

A commonly used learning rate function for SGD is $\eta(t) = \eta_0 t^{-p}$, which offers the best yergence speed in theory and also does it often in practice (Bottou, 2012), however it can lead to poor performance due to slow rates of convergence to the solution (Darken and Moody, 1992). By contrast, choosing a larger η_0 may not lead to better results due to the instability in the parameters for small t (Darken and Moody, 1990).
This behavior was observed during the experiments with real data to be presented in later sections, therefore an alternative for the learning rate was sought. In particular, Adaptive Gradient (AdaGrad) was implemented.
AdaGrad was proposed by Duchi et al. (2011), the idea of this method is to adapt the learning rate for each parameter in $\boldsymbol{\theta}$ individually in a way that frequently updated parameters have slower learning rates while infrequent parameters have larger ones. The intuition is that an update to a rare parameter is more informative than one to a parameter that is frequently updated.
Concretely, with AdaGrad the update step for each parameter γ in $\boldsymbol{\theta}$ is given by:

$$\theta_\gamma^{\tau+1} \leftarrow \theta_\gamma^\tau - \frac{\eta}{\sqrt{G_\gamma}} \nabla g(\theta)_\gamma^\tau \tag{3.18}$$

Where $\mathbf{G}^\tau$ is the vector of accumulated gradients at time τ and each component G_γ^τ is given by:

$$G_\gamma^\tau = \sum_{t=1}^{\tau} \left(\nabla g(\theta)_\gamma^t \right)^2 \tag{3.19}$$

Where $\nabla g(\theta)_\gamma^t$ denotes the gradient of the γ parameter at time step t.
It is worth noting that AdaGrad does not incur in an significant increase in running time or memory requirements for the training which makes it inexpensive to include in the implementation of NCE.

3.5 Mean-shift Clustering

Mean-shift clustering is a technique proposed by Cheng (1995), it is commonly used for image segmentation tasks (Comaniciu and Meer, 2002). However, as proposed by Kleinberg et al. (2009), it can also be used for the task of identifying highly photographed locations in an image collection.
It is a non-parametric technique for locating the modes of a probability distribution over a feature space, this works by using Kernel Density Estimation (KDE) to estimate the gradient of the underlying distribution from the data and then performing a mean-shift procedure to identify the local maxima or modes, this is also known as hill-climbing. This can be naturally applied to a collection of geotagged photos by considering the feature space of latitude and longitude coordinates.
This approach assumes that there is an underlying distribution where the modes are locations of interest or landmarks where many of the photos were taken. Mean-shift has an advantage over other clustering techniques such as k-means for this problem, because the bandwidth used for the kernel has a physical meaning, namely it defines the radius of the area around each landmark.

Chapter 4

Extending FLID

4.1 Beyond Diversity: FLDC

Diversity is an important property in the context of summarization (Tschiatschek et al., 2016), however it is not the only one. Coherence is also a desired property of summaries, particularly for structured summaries (Yan et al., 2011). Balancing coherence and diversity is considered a challenge and maxiziming diversity alone may lead to disconnected summaries, while focusing on coherence can hinder the breadth of the resulting summaries (Shahaf et al., 2012).

To better understand why coherence is important, consider the construction of a spatial summary for photos taken in a city. A model based on spatial diversity would create sets containing photos taking at distant locations in the city, however another possibility is to have summaries that contain photos that were taken close to some starting point, in this case the desired property is not spatial diversity but rather coherence.

Therefore, a natural extension to the FLID model is to add the capacity of modeling coherence along with diversity. We propose the addition of a log-supermodular term analogous to the log-submodular term (3.5) in order to model the complementarity between elements.

This supermodular term introduces a representation of the elements using K-dimensional vectors $\mathbf{w}^K \in \mathbb{R}^K_{\geq 0}$, where each dimension $1 \leq c \leq K$ captures a concept that relates to set coherence and $w_{i,c}$ quantifies the contribution of each element i to each coherence concept c. The coherence of a set S with respect to each concept c is quantified as $\sum_{i \in S} w^e_{i,c} - \max_{i \in S} w^e_{i,c}$ which is a supermodular term mirroring the submodular term in FLID. Summing over all the concepts results in the proposed coherence term. Hereby, define $\mathbf{W}^e \in \mathbb{R}^{|V| \times K}_{\geq 0}$ as the matrix where the i-th row is the representation $\mathbf{w}^e$ of element i.

Adding the coherence term to FLID results in the extended model, which we refer to as the Facility Location Diversity and Coherence (FLDC) model. Its probability distribution is defined as:

$$P(S) = \frac{1}{Z} \exp\left(\sum_{i \in S} u_i + \operatorname{div}(S) + \sum_{c=1}^{K} \left(\sum_{i \in S} w^e_{i,c} - \max_{i \in S} w^e_{i,c} \right) \right) \tag{FLDC}$$

For the FLDC model, the partition function can be computed in $O(|V|^{L+K+1})$ time. This complexity can be derived using a similar algorithm as the one presented by Tschiatschek et al. (2016) for FLID, the key observation is that the algorithm they present only requires the ordering of weights to determine elegible element for a set given a set of maximum weights per dimension, therefore this can be easily extented to include the the ordering for the coherence weights as well. This requires evaluating K more dimensions which

Table 4.1: Probability distribution for Example 4.1.2.

S	$P(S)$
$\{1,2\}$	0.5
$\{3,4\}$	0.5
Otherwise	0.0

results in the added term to the exponent of the complexity expression, which makes the computation of the partition function more impractical than in the FLID case.

4.1.1 Learning FLDC Models

As with FLID, the parameters for an FLDC model can be estimated using NCE. For FLDC the vector $\boldsymbol{\theta}$ is composed of $[\mathbf{u}, \mathbf{W}^b, \mathbf{W}^e, \hat{Z}]$ and its gradient is the same as in equations (3.13)-(3.17) with the addition of a gradient with respect to the new parameter, $\mathbf{W}^e$, which is given by:

$$\left(\nabla_{\mathbf{W}^e} \log \hat{P}_d(S; \hat{Z}, \mathbf{u}, \mathbf{W}^e, \mathbf{W}^e)\right)_{i,c} = \begin{cases} 1 & \text{if } i \in S \text{ and } i \neq \operatorname{argmax}_{j \in S} w^e_{j,c} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Where $(\nabla_{\mathbf{W}^e} \cdot)_{i,c}$ represents the (i,c)-th entry in the gradient with respect to $\mathbf{W}^e$.
This also affects the overall complexity of the learning algorithm. Computing the gradient takes $O(L+K)|S|$ for FLDC at each step, therefore the running time of a full pass over the data and noise samples takes $O(|\mathcal{D} \cup \mathcal{N}|\kappa(L+K))$. This is not significantly more expensive than learning FLID under the assumption that $K \simeq L$.

4.1.2 Example: Two Non-overlapping Clusters

As an example of the extended model, consider the distribution presented in Table 4.1 for $V = \{1,2,3,4\}$, representing a set of two non-overlapping clusters with 2 elements each. Intuitively, this can be modeled by considering that there is diversity between elements in different clusters while inside a cluster there is a coherence component that ties them together.
Concretely, the weight matrices $\mathbf{W}^b, \mathbf{W}^e$ in Figure 4.1 illustrate one possible instance of the model. The corresponding utility vector is $\mathbf{u} = \overrightarrow{0}$, because there is no indication that an individual element is favored over the pairs. This model is easily interpretable and accurately realizes the distribution.

4.2 Generalizing through Features: FFLDC

An important characteristic of the FLID model, and by extension the FLDC one, is that it is agnostic to the type of elements in the ground set, this allows its application to a wide range of problems without prior knowledge. However the downside is that the model has no capability to make use of information about the elements, if available, to improve the modeling of the data. Moreover, if a new element is added to the set there is no way to generalize the existing knowledge about similar elements.
In order to solve these problems, we propose a further extension to the model. Firstly, an element ($i \in V$) is represented as a vector $\mathbf{x}_i \in \mathbb{R}^M$ where each entry is a feature, e.g. for venues one feature could be its aggregated rating while another indicates whether it is indoors or outdoors. The representations of all elements is collected by matrix $\mathbf{X} \in \mathbb{R}^{|V| \times M}$ where each row i corresponds to the repreesntation $\mathbf{x}_i$ of element i.

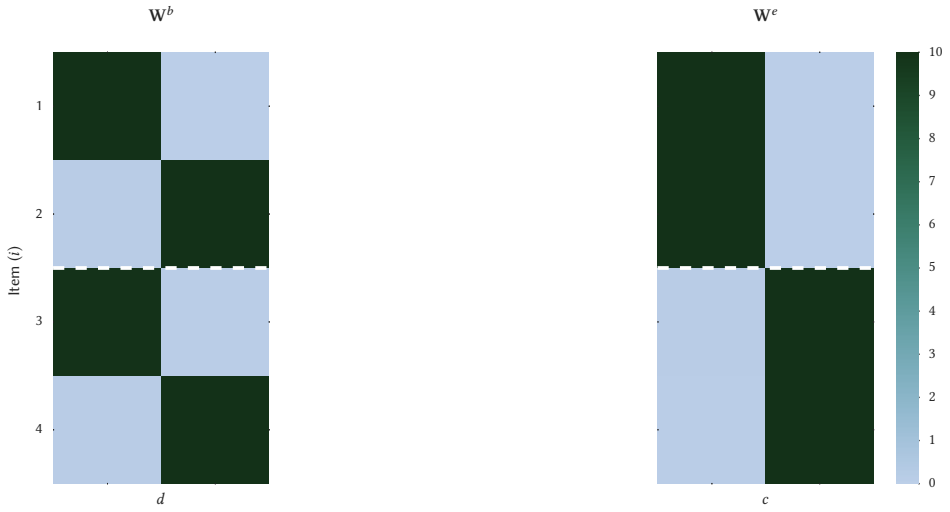


Figure 4.1: Diversity and coherence weights for FLDC model in Example 4.1.2. The dotted line divides the clusters.

Then, the utility vector $\mathbf{u}$ and weight matrices $\mathbf{W}^b, \mathbf{W}^e$ are factorized to include this feature matrix $\mathbf{X}$ as follows:

$$\mathbf{u} = \mathbf{X}\mathbf{a} \tag{4.2}$$

$$\mathbf{W}^b = \mathbf{X}\mathbf{B} \tag{4.3}$$

$$\mathbf{W}^e = \mathbf{X}\mathbf{E} \tag{4.4}$$

Where $\mathbf{a} \in \mathbb{R}^M$ represents the contribution of each feature to the total utility of an item, while $\mathbf{B} \in \mathbb{R}^{M \times L}$ and $\mathbf{E} \in \mathbb{R}^{M \times K}$ encode the contribution of each feature to each latent diversity and coherence dimension, respectively. The intuition behind this factorization is that the information about the items can enhance the latent representations, hence producing a richer model. For example if $M < |V|$ and the features are sufficient to represent the items, it is expected that learning will be easier because of the reduced number of parameters with respect to learning the equivalent FLDC model.
We refer to the extended model as the Featurized Facility Location Diversity and Coherence (FFLDC) model and its probability distribution is defined as:

$$P(S) = \frac{1}{Z} \exp\left(\sum_{i \in S} \sum_{k=1}^{M} x_{i,k} a_k + fdiv(S) + fcoh(S) \right) \tag{FFLDC}$$

$$fdiv(S) = \sum_{d=1}^{L} \left(\max_{i \in S} \sum_{k=1}^{M} x_{i,k} b_{k,d} - \sum_{i \in S} \sum_{k=1}^{M} x_{i,k} b_{k,d} \right) \tag{4.5}$$

$$fcoh(S) = \sum_{c=1}^{K} \left(\sum_{i \in S} \sum_{k=1}^{M} x_{i,k} e_{k,c} - \max_{i \in S} \sum_{k=1}^{M} x_{i,k} e_{k,c} \right) \tag{4.6}$$

Where a_i is the i-th entry of $\mathbf{a}$, $b_{k,d}$ is the (k,d)-th entry of $\mathbf{B}$, $e_{k,d}$ is the (k,e)-th entry of $\mathbf{E}$ and $x_{i,k}$ is the (i,k)-th entry of X.

Remark. If $\mathbf{X} = \mathcal{I}$, then FFLDC is equivalent to FLDC, with $\mathbf{a} = \mathbf{u}$, $\mathbf{W}^b = \mathbf{B}$ and $\mathbf{W}^e = \mathbf{E}$.

The use of features also allows the application of the model to previously unseen elements, hence solving the aforementioned problem of generalization. This is possible because the

model is defined on the space of features and not items, and for a previously unseen element j only its representation $\mathbf{x}_j$ is required in order to evaluate the probability of sets containing this element.

The computation of the partition function for this model can be done using the fact that any FFLDC model can be transformed into an equivalent FLDC model through the factorizations in equations (4.2)-(4.4). This transformation requires $O(|V|M(L+K))$ time for the matrix multiplications, therefore the overall complexity of calculating the partition function for an FFLDC model is $O(|V|M(L+K)+|V|^{L+K+1})$. This expression can be simplified to $O(|V|^{L+K+1})$ because the exponential term is significantly larger assuming sensible values for M. Given that the complexity of computing the partition function for FFLDC is equivalent to the FLDC model, it is also intractable in practice.

4.2.1 Learning FFLDC Models

For FFLDC, the parameter vector $\boldsymbol{\theta}$ is different from the previous models, it is composed by $[\mathbf{a}, \mathbf{B}, \mathbf{C}, \hat{Z}]$ and the gradient of the NCE objective $g(\boldsymbol{\theta})$ is given by:

$$\left(\nabla_{\mathbf{a}} \log \hat{P}_d(S; \hat{Z}, \mathbf{a}, \mathbf{B}, \mathbf{E})\right)_m = \sum_{i \in S} x_{i,m} \tag{4.7}$$

$$\left(\nabla_{\mathbf{B}} \log \hat{P}_d(S; \hat{Z}, \mathbf{a}, \mathbf{B}, \mathbf{E})\right)_{m,d} = x_{i^d,m} - \sum_{i \in S} x_{i,m} \tag{4.8}$$

$$\left(\nabla_{\mathbf{E}} \log \hat{P}_d(S; \hat{Z}, \mathbf{a}, \mathbf{B}, \mathbf{E})\right)_{m,c} = x_{i^c,m} - \sum_{i \in S} x_{i,m} \tag{4.9}$$

Where $i^d = \operatorname{argmax}_{i \in S} \sum_{k=1}^{M} x_{i,k} b_{k,d}$ and $i^c = \operatorname{argmax}_{i \in S} \sum_{k=1}^{M} x_{i,k} e_{k,c}$. $(\nabla_{\mathbf{a}} \cdot)_m$ represents the m-th entry of the sub-gradient with respect to a, $(\nabla_{\mathbf{B}} \cdot)_{m,d}$ represents the (m,d)-th entry of the sub-gradient with respect to B and $(\nabla_{\mathbf{E}} \cdot)_{m,c}$ represents the (m,c)-th entry of the sub-gradient with respect to E.

Remark. For $\mathbf{X} = \mathcal{I}$, the sub-gradients in 4.7-4.9 are equivalent to the FLDC sub-gradients. Because $x_{i,m} = 1$ only when $i = m$.

Regarding running time complexity, the FFLDC model requires more operations for the gradient due to the inclusion of features. Concretely, computing the gradient for a set S takes $O(|S|M(L+K))$ which makes the running time of a single pass over data and noise samples $O(|\mathcal{D} \cup \mathcal{N}|M(L+K)\kappa)$. Even though this is larger than the complexity of FLID or FLDC, it is still linear on the data.

4.2.2 Example: Rated Locations

A small town has 6 popular locations, each of them has been rated from 0 to 5, where 0 represents a bad review and 5 an excellent review. Collected data shows that a typical visitor is more likely to visit and take photos at places with high ratings, however this is not the only factor driving their behavior. For example, an average visitor does not visit two places that serve food on the same day instead the data shows that visitors usually go to one place that serves food and then to an outdoor one, or they just visit one that has both characteristics. This data is summarized in Table 4.2, the task is to model this behavior using the FFLDC model.

In this example there is knowledge about the items and what features are relevant for the data. These features are summarized in equation (4.10). The first column corresponds to the aforementioned rating, the second and third are binary features indicating whether the location is an outdoor one and whether it serves food, respectively.

Table 4.2: Synthetic data for Example 4.2.2.

Locations visited (S)	$P(S)$
$\{1,3\}$	0.30
$\{3,4\}$	0.25
$\{3,6\}$	0.15
$\{2\}$	0.10
$\{1\}$	0.06
$\{3\},\{4\}$	0.04
$\{5\},\{6\}$	0.03
Otherwise	0.00

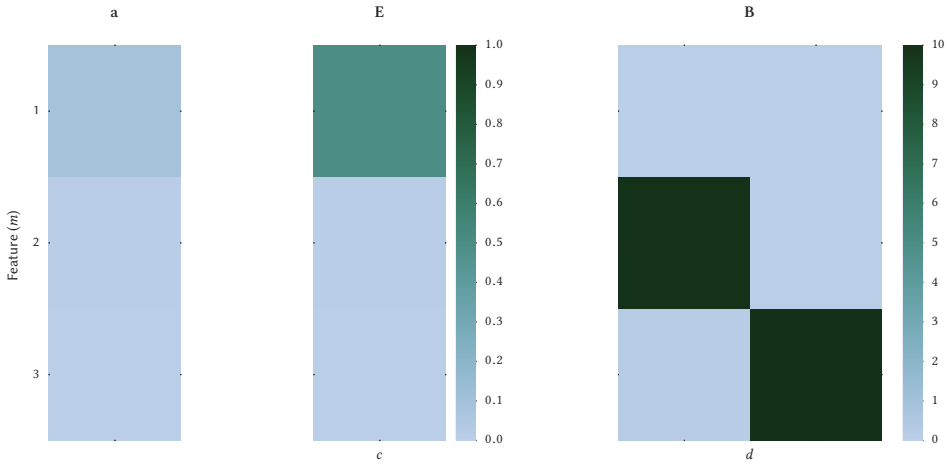


Figure 4.2: FFLDC sample model for Example 4.2.2.

$$\mathbf{X} = \begin{pmatrix} 4 & 1 & 0 \\ 4 & 1 & 1 \\ 3 & 0 & 1 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix} \tag{4.10}$$

A possiblity is an FFLDC model that encourages diversity on the second a third feature, i.e. models the idea that visitors do not go to more than one place that serves food or is outdoor, while assigning a positive utility and coherence value to the first feature, i.e. modeling the preference for places with high ratings. One such model is presented in Figure 4.2, this approximates distribution from Table 4.2 and illustrates the type of model that is useful in this example.

If a new location j is considered then it is possible to apply the model and estimate the probability of sets that include it. For example, consider a seventh location with the following features $\mathbf{x}_7 = [5, 1, 0]$, using the same model parameters a new distribution can be computed without changing the model parameters, as it would be the case with FLID or FLDC. The updated distribution is shown in Table 4.3 and it can be considered a sensible distribution given the problem description and the high rating of the new location, hence

Table 4.3: Synthetic data for Example 4.2.2 after adding a new item.

Locations visited (S)	$P(S)$
$\{3,7\}$	0.24
$\{1,3\}$	0.21
$\{3,4\}$	0.19
$\{3,6\}$	0.10
$\{7\}$	0.06
$\{1\},\{2\}$	0.04
$\{3\},\{4\}$	0.03
$\{5\},\{6\}$	0.02
Otherwise	0.00

showing the model's capacity to generalize.

Chapter 5

Synthetic Experiments

This chapter describes experiments done on synthetic data to explore the learning performance of the methods described in the background section.

5.1 Learning Setup

This section describes implementation details of NCE and SGD, these are applicable for both the experiments in this chapter as well as those performed on real data, which will presented in subsequent chapters.

5.1.1 Noise Distribution

As mentioned in section 3.4.2, there are several considerations when choosing the noise distribution. A log-modular distribution is a natural choice for the noise when learning the FLID model (Tschiatschek et al., 2016), and this distribution will also be used for the noise with FLDC and FFLDC models.
The parameters of a log-modular distribution can be estimated efficiently from data through MLE, as given by Equation (3.8). Also the partition function can be computed in closed form, as shown in Equation (3.9). This makes the distribution a good candidate because it can be used efficiently for sampling as well as for computing the normalized probability of sets, i.e. $P_n(S)$ during the learning procedure.

5.1.2 Initialization

The log-modular model also provides a sensible initialization for vector $\mathbf{u}$ in the FLID and FLDC models. For FFLDC, the initialization for $\mathbf{a}$ is done by solving the linear system derived from the factorization in Equation (4.2), this linear system is given by:

$$\mathbf{a} = \mathbf{X}^{-1}\mathbf{u} \tag{5.1}$$

Unfortunately, this system can be under/over-determined in many cases so an approximated solution must be used. The chosen method for finding this solution is Ridge regression which minimizes the squared error $\sum_{i \in S}(u_i - (\mathbf{Xa})_i)^2$ with a regularization term on the l_2 norm of $\mathbf{a}$.
The weight matrices, e.g. $\mathbf{W}^e, \mathbf{B}$, are initialized with random values drawn from a uniform distribution $\mathcal{U} \sim [0, 0.001]$. These is the same initialization setting proposed by Tschiatschek et al. (2016).
In the cases where AdaGrad is used, the accumulated gradient $\mathbf{G}$ is initialized with a constant value for each parameter γ, namely $G^0_\gamma = 0.01$.

5.1.3 Learning Rate

When not using AdaGrad, the learning rate used in the experiments follows the power law mentioned in section 3.4.4, i.e.

$$\eta(t) = \frac{\eta_0}{t^p} \tag{5.2}$$

5.1.4 Projection in SGD

The weight matrices, e.g. $\mathbf{W}^e, \mathbf{B}$, are defined only for positive values. Therefore, they must be projected in each gradient step to ensure that the solution stays in the feasible space. This is done following the procedure proposed by Tschiatschek et al. (2016) which produces better results than simply clipping the values to 0. Let θ_h be an element of these matrices, e.g. $\theta_h = w^e_{i,d}$, then the projection is defined as:

$$\theta_h = \begin{cases} \theta'_h & \text{if } \theta'_h \geq 0 \\ \mathcal{U} \sim [0, 0.001] & \text{otherwise} \end{cases} \tag{5.3}$$

Where θ'_h is the unnormalized parameter after the gradient update and $\mathcal{U} \sim [0, 0.001]$ means that the value is drawn from an uniform distribution between $[0, 0.001]$ as in the initialization.

5.1.5 Column Restarts

One common problem observed during the experiments is when two columns in the weight matrices are very similar and therefore redundant. This can happen for some initialization settings, e.g. when i^d and i^e in Equations (4.8) and (4.9) are the same initially and increase at the same rate, hence leading to a local maximum.
In order to address this problem, a heuristic was added to the implementation that resets one of the columns when it is too similar to another one. This similarity is measured as:

$$\text{sim}_{\mathbf{W}}(a, b) = \hat{\mathbf{W}}^{\mathsf{T}}_{*,a} \cdot \hat{\mathbf{W}}_{*,b} \tag{5.4}$$

Where a and b are column indexes for a weight matrix $\mathbf{W}$, e.g. $\mathbf{W}^b$, and $\hat{\mathbf{W}}_{*,a}$ and $\hat{\mathbf{W}}_{*,b}$ are the normalized a-th and b-th columns of $\mathbf{W}$, respectively.
The column pair with maximum similarity is identified every N_r iterations and if the similarity is larger than a threshold δ then the first column is added to the second one, and the first one is restarted to random values as done in the initialization step. The expectation is that the reset column will learn different information with the new initialization values. Usually $N_r = 500$ and $\delta = 0.9$ was used.

5.1.6 Termination Criteria

The termination criteria for learning was the number of epochs, i.e. full passes over the data and noise samples.

5.2 Example Datasets

This section presents the results obtained after learning the datasets from the examples given for each model. For these experiments, the settings were:

- 1000 samples of data drawn from the corresponding distribution, i.e. $|\mathcal{D}| = 1000$.
- 2000 noise samples, i.e. $\nu = 2$.
- 100 epochs.

Table 5.1: Learned distribution for Example 4.2.2

S	$P(S)$
$\{h, s\}$	0.48
$\{h, f\}$	0.47
$\{h\}$	0.03
$\{h, s, f\}$	0.02

- η_0 is 0.005 and $p = 0.1$ for the learning rate, without AdaGrad.

It is worth noting that the presented results for these experiments correspond to the best of multiple runs. It was observed that the learning algorithm does not always arrive to a good solution. This is because $g(\boldsymbol{\theta})$ is not a concave function and some initializations lead to local maxima.

5.2.1 FLID: Two Landmarks

Recall the following characteristics about the dataset in Example 3.3.1:

- There are 3 elements, h, s, f.
- There are only 2 possible sets, $\{h, s\}$ and $\{h, f\}$. Both equally likely, i.e. $P(S) = 0.5$.
- A model can be constructed with a single diversity dimension, i.e. $L = 1$.

After estimating the parameters from this data, the resulting utility vector and weight matrix are:

$$\mathbf{u} = (5.42, 2.82, 2.79)^\intercal$$
$$\mathbf{W}^b = (0.02, 6.10, 6.10)^\intercal$$

This is a slightly different model from the one suggested in section 3.3.1, however the normalized probability distribution for this model presented in Table 5.1 shows that it closely approximates the distribution.

5.2.2 FLDC: Two Non-overlapping Clusters

Recall the following characteristics about the dataset in Example 4.1.2:

- There are 4 items, i.e. $V = \{1, 2, 3, 4\}$.
- There are only 2 possible sets, $\{1, 2\}$ and $\{3, 4\}$. Both equally likely, i.e. $P(S) = 0.5$.
- A model can be constructed with two diversity and two coherence dimensions, i.e. $L = 2, K = 2$.

Figure 5.1 shows the resulting utility vector $\mathbf{u}$ and weight matrices $\mathbf{W}^b, \mathbf{W}^e$ after the learning procedure.
The model shown in the figure displays the desired properties of diversity between the clusters while having coherence between the items in each one. Despite being different from the proposed model in section 4.1.2 it also approximately realizes the desired distribution.

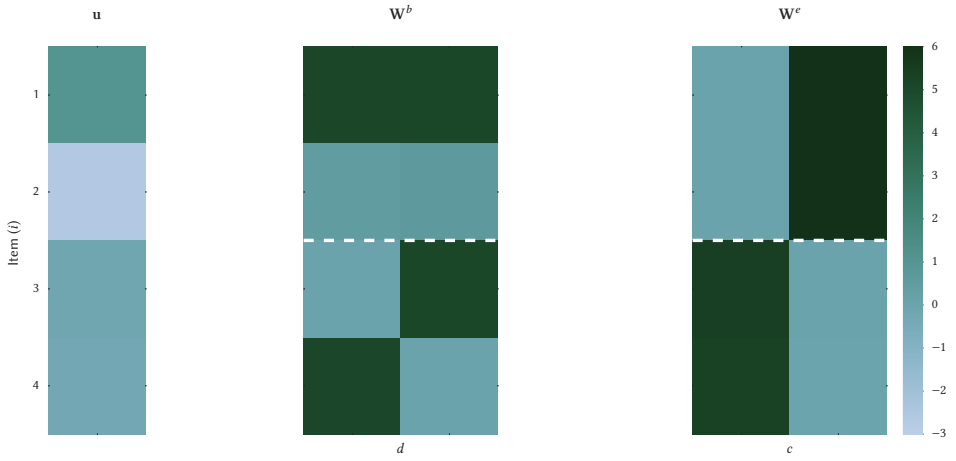


Figure 5.1: Learned model for example 4.1.2. The white line divides the clusters described in the example.

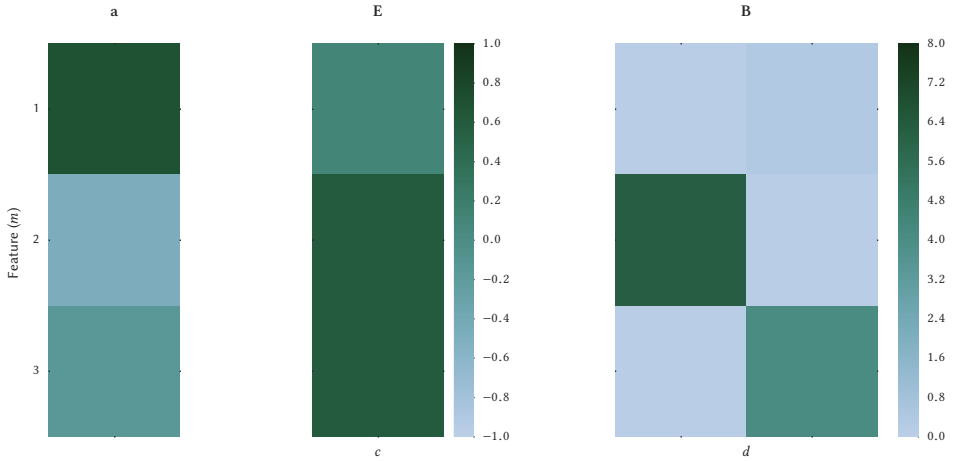


Figure 5.2: Learned model for example 4.2.2

FFLDC: Rated Locations

Recall the following characteristics about the dataset in Example 4.2.2:

- There are 6 items, i.e. $V = \{1, 2, 3, 4, 5, 6\}$.
- The full distribution is related to the features and is given in Table 4.2.
- The proposed model has 2 diversity dimensions and one coherence dimension.

Figure 5.2 shows the resulting model after learning. Unfortunately, this model can be not as easily interpreted as the one proposed in figure 4.2, nevertheless it accurately models the distribution from the example thus showing the learning algorithm's ability to estimate a distribution using features.

5.3 Learning Rate Sensitivity

This section explores the learning rate's effect on the learning performance for the synthetic dataset from Example 4.2.2.

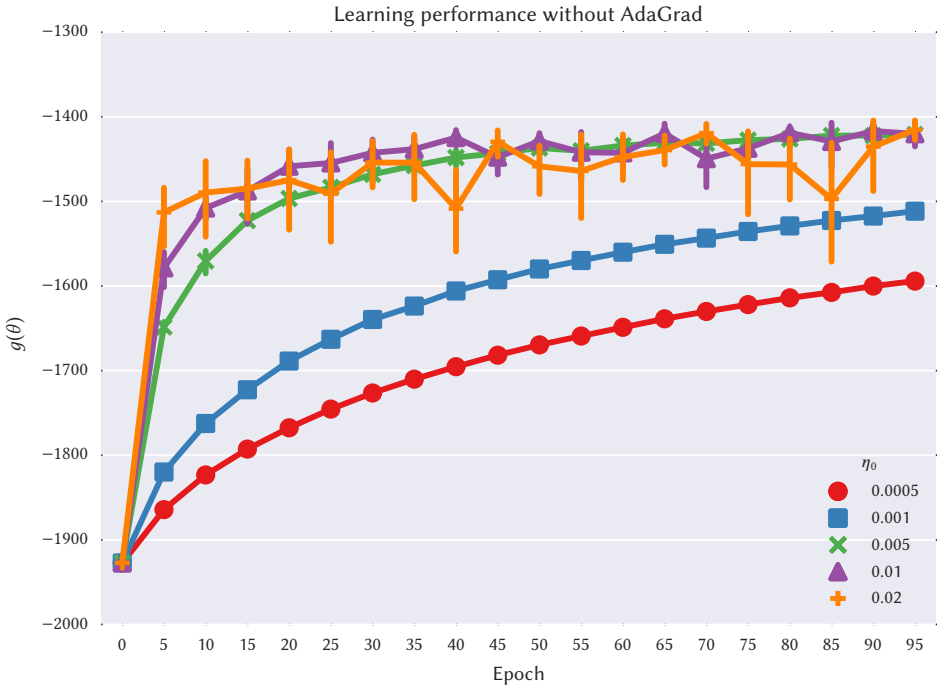


Figure 5.3: Objective function for various values of η_0 without using AdaGrad.

The first experiment considers the stability of the optimization without AdaGrad as the learning rate increases. The settings are the same as for the previous experiments with the following additions:

- η_0 is increased between 0.0005 and 0.02
- The learning is performed independently 8 times to obtain statistics on the results.
- For these experiments the reset heuristic for columns was disabled to prevent sudden changes in the objective function due to the resets.

Figure 5.3 shows $g(\boldsymbol{\theta})$ during the optimization, each point corresponds to the mean value calculated after a given number of epochs and the error bars show the corresponding 95% confidence interval for the mean.
The results in figure 5.3 show that for small values of η_0 the learning can be significantly slow and that large values of η_0 may lead to instability, as discussed in section 3.4.4. Concretely, for $\eta_0 \leq 0.01$ the learning is slow but stable, while for $\eta_0 \geq 0.01$ the objective value converges quickly but oscillates across epochs. This oscillation is problematic because it breaks the guarantee that the objective value is better as the number of epochs increase. Fortunately, in the particular case of this synthetic dataset there exists a learning rate for which the objective function converges and is stable, i.e. $\eta_0 = 0.005$ however this may not be always the case for real data.
By contrast, figure 5.4 shows the objective function for η_0 between 0.005 and 0.05 after incorporating AdaGrad to the implementation. These results show that training with AdaGrad is stable for a larger range of eta_0 without sacrificing convergence speed.
Finally, figure 5.5 shows a direct comparison between training with and without AdaGrad using the best values of η_0 for each configuration. In this figure both algorithms converge to an optimal value, however it also shows that training with AdaGrad converges significantly faster while mantaining a more stable mean. These combined results indicate that using AdaGrad is the best strategy for learning the FFLDC model.

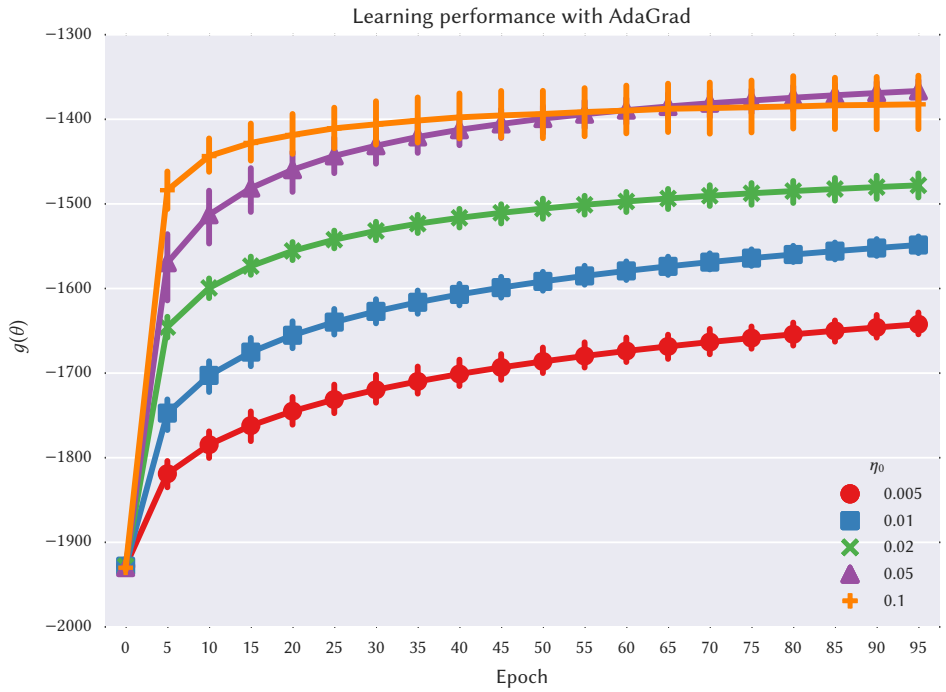


Figure 5.4: Objective function during learning for various values of η_0 with AdaGrad.

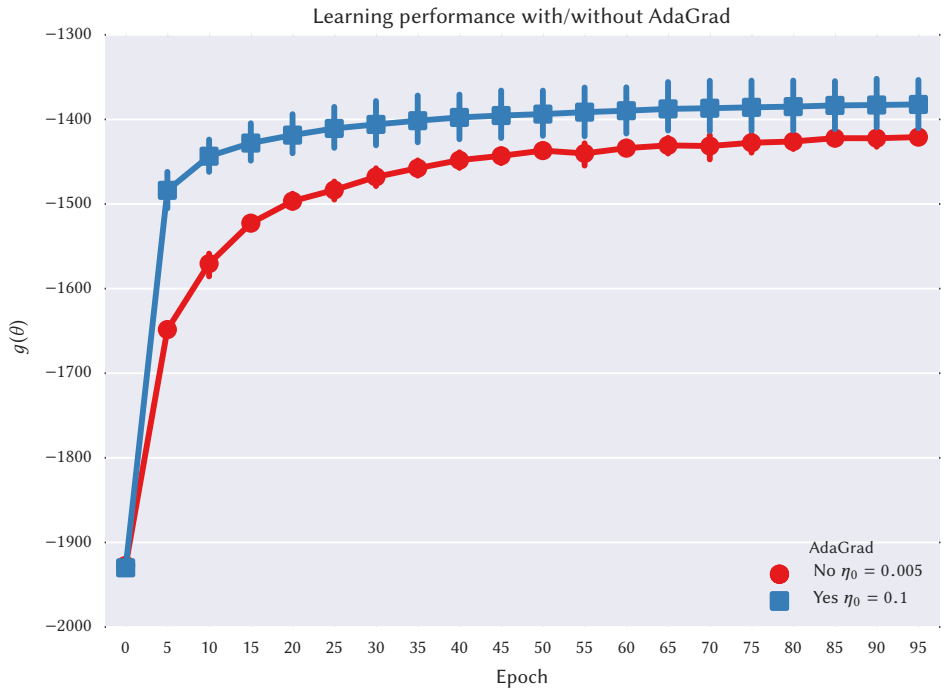


Figure 5.5: Comparison of learning performance with and without AdaGrad.

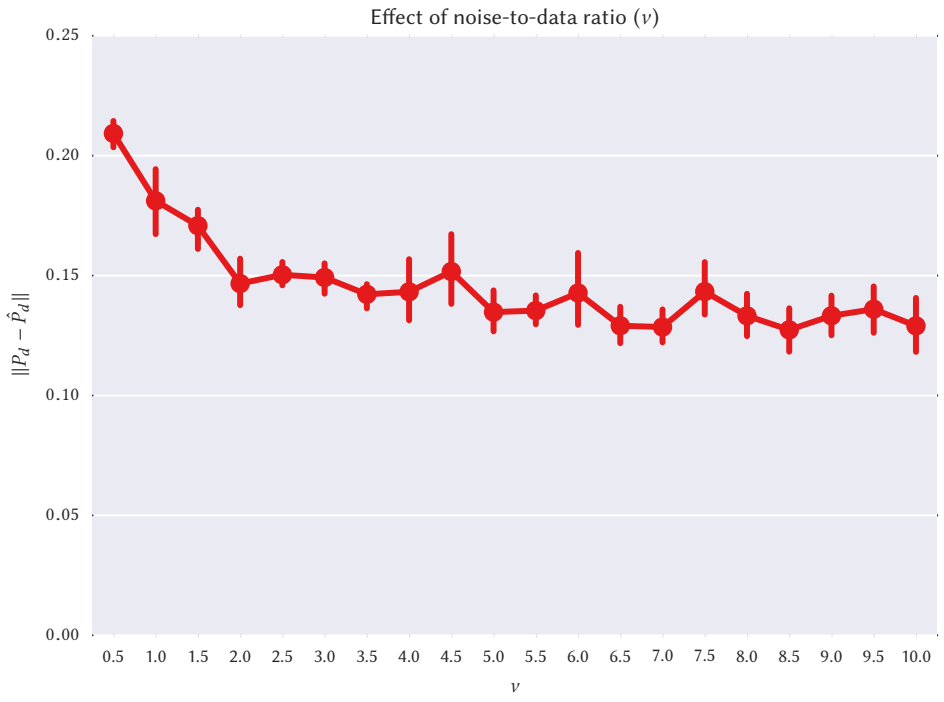


Figure 5.6: Total variation distance for different values of ν.

5.4 Noise-to-data Ratio

The noise-to-data ratio ν is another hyperparameter for NCE learning, the recommendation is to have as many noise samples as computationally possible (Gutmann and Hyvärinen, 2012). In order to better understand the effect of ν, an experiment was performed where different models for the dataset from Example 4.2.2 are learned with different values of ν. In order to compare the models, the Total Variation Distance (TVD) between the target distribution P_d, from Table 4.2, and the learned distribution $\hat{P}_d$ was used, intuitively this measures how good is the approximation of the learned model to the real distribution. This is defined as (Levin et al., 2008, chap. 4):

$$\|P_d - \hat{P}_d\|_{TV} = \frac{1}{2} \sum_{S \subseteq V} |P_d(S) - \hat{P}_d(S)| \tag{5.5}$$

The setting is the same as for the learning rate experiments, but with $\eta_0 = 0.05$ and always using AdaGrad. The noise parameter ν was varied between 0.5 and 10.
The results for this experiment are shown in Figure 5.6. It shows that increasing the number of noise samples reduces the variation distance of the resulting model as expected. However, it only decreases signficantly for small values of ν, this may be because the number of epochs is not sufficient for larger ν. This last assumption was tested by running another experiment with fixed $\nu = 10$ but increasing the number of epochs from 100 to 1500, the results are presented in Figure 5.7.
Figure 5.7 show that increasing the number of epochs can significantly improve the quality of the learned model. The decreasing distance indicates that it is possible for the NCE learning procedure to converge to the true distribution given sufficient noise samples and epochs.

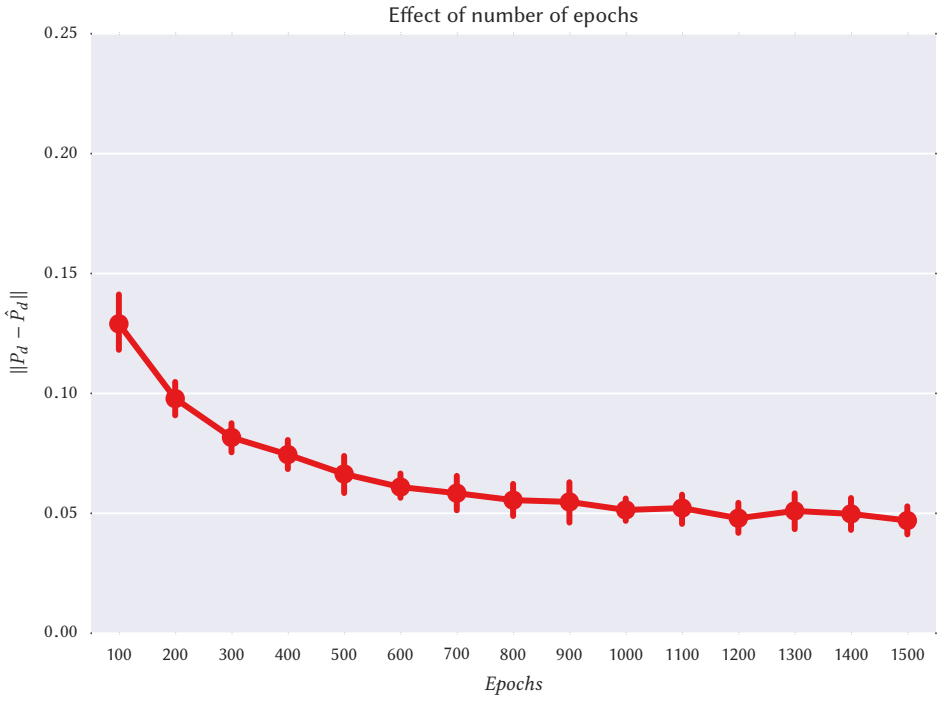


Figure 5.7: Total variation distance for different number of epochs.

Chapter 6

Modeling an Image Collection - Setup

This chapter describes an application of the models presented in the Chapter 4 to a real dataset. As mentioned in the introduction, finding and recommending paths for tourists is an interesting research topic. These paths can be understood as spatial summaries of a city, representing the most important locations to visit and photograph in a city.
These tourist paths can be discovered from a large collection of geotagged images by using the accompanying timestamps as explained in section 6.2. Even though these paths are naturally ordered, they can also be used as unordered sets of locations that represent the behavior of different visitors taking photographs in the city, which can be modeled using the FLID, FLDC and FFLDC models. The modeling quality of these proposed models was compared to other existing methods from the literature as explained in section 6.4. The procedure used for evaluating the results are described in section 6.5.

6.1 Flickr Dataset

For the experiments on real data, public geotagged photos from Flickr were used. There are close to two million geo-tagged photos available on Flickr[1], this offers an unique and rich dataset for the problem of spatial summarization. But as described before, this work focuses on the city level. Therefore, the experiments were performed on a subset of the Flickr data, namely on public geotagged photos taken within Zürich, Switzerland.
The Flickr API[2] provides multiple functions to explore and use the data, in particular the search endpoint allows developers to retrieve photo records according to a specific criteria. Each photo record contains the following information:

Owner The ID of the user who uploaded and owns the photo.

Title A title given by the user.

Description A description given by the user.

Visibility Indicates if a photo is public, private or semi-private, i.e. shared only with friends.

Dates Dates when the photo was taken, uploaded and last modified.

Geo Indicates if the photo is geotagged, the geotag's precision, latitude and longitude.

[1] https://www.flickr.com/map reports 1,833,307 geotagged photos as of April 2016.
[2] `https://www.flickr.com/services/api/`

Figure 6.1: Bounding box used for filtering photos taken in the Zürich area.

License Copyright license.

Tags Text tags set by users.

Machine tags Automatic text tags set by Flickr.

URL The URL to download the photo.

Most of these properties can be used to filter the search results, for collecting the desired data for Zürich the following criteria was used:

- The photo was taken between 2005 and 2015.
- The photo is marked as public and has no copyright restrictions.
- The photo is geotagged and has street level accuracy. The location of the photo must be within the area shown in Figure 6.1.

The API sets some limitations on the search function, namely it does not allow more than 3600 requests per hour and it restricts the search results to the first 4000 records. This means that it was not possible to issue a single search request with the outlined criteria, instead the retrieval was done over the course of several hours though single month queries. This ensured that each query had less than 4000 results, hence retrieving all the available records. This search yielded 168608 photos taken by 6159 users.
Figure 6.2 shows the distribution of photos taken in 2015. The majority of them is concentrated at the city center also around locations of interest outside the center like Uetliberg, Oerlikon and the Zoo. This indicates that the dataset contains good information regarding the places to visit and take photographs in Zürich. Figure 6.3 shows zoomed in areas around the city showing that the data is mostly concentrated in touristic places like the historic center, where Fraumünster, Grossmünster and the city hall are.

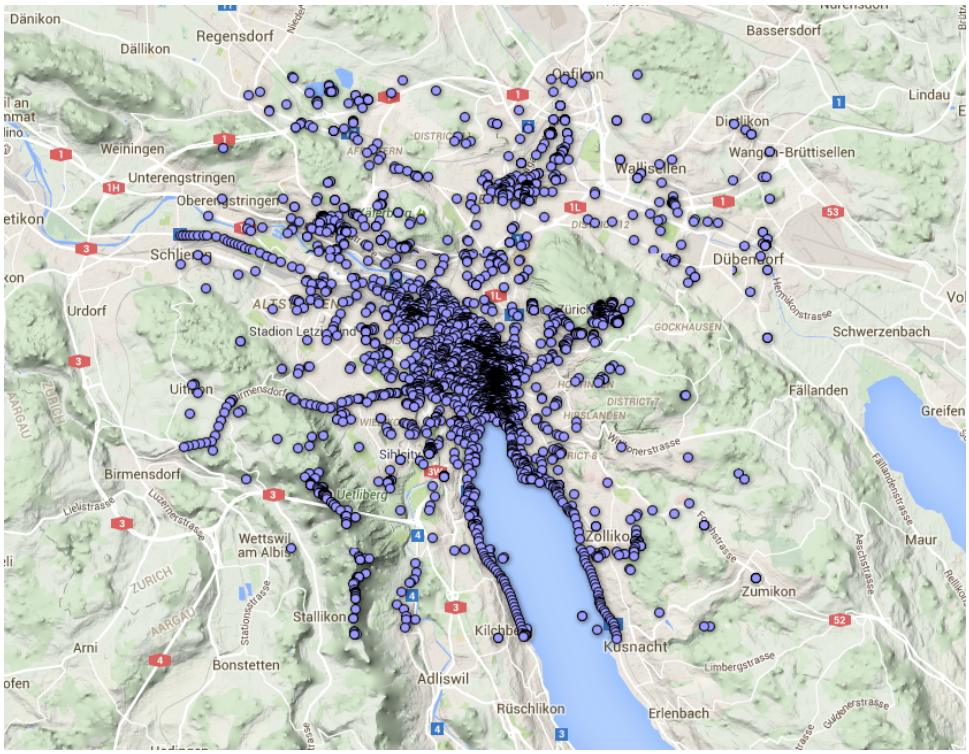
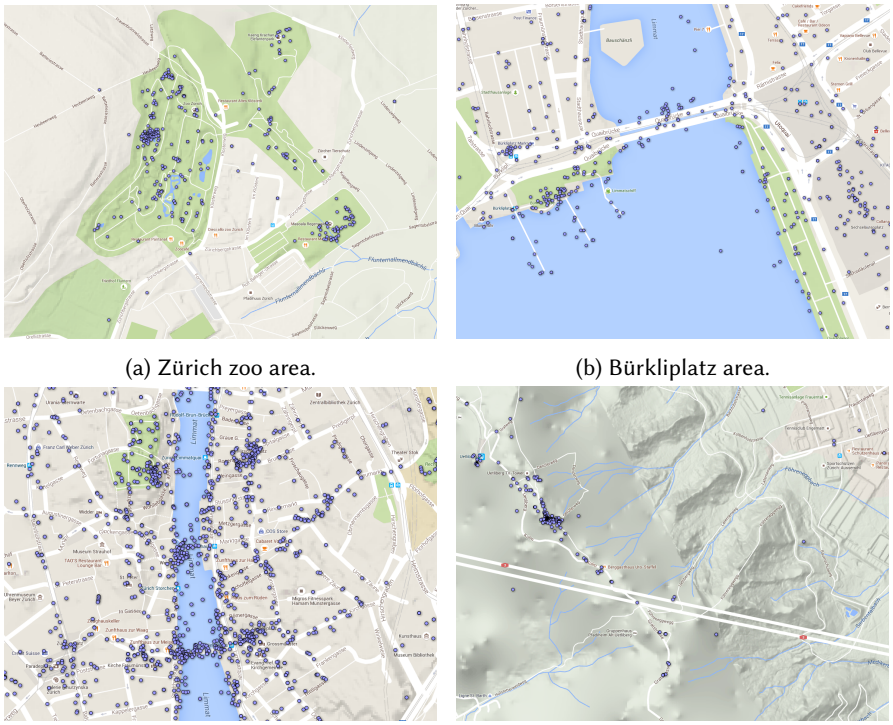


Figure 6.2: Feature map with the photos from the Zürich dataset taken in 2015.



(a) Zürich zoo area.

(b) Bürkliplatz area.

(c) Historic center area.

(d) Uetliberg area.

Figure 6.3: Zoomed in maps displaying popular locations in Zürich.

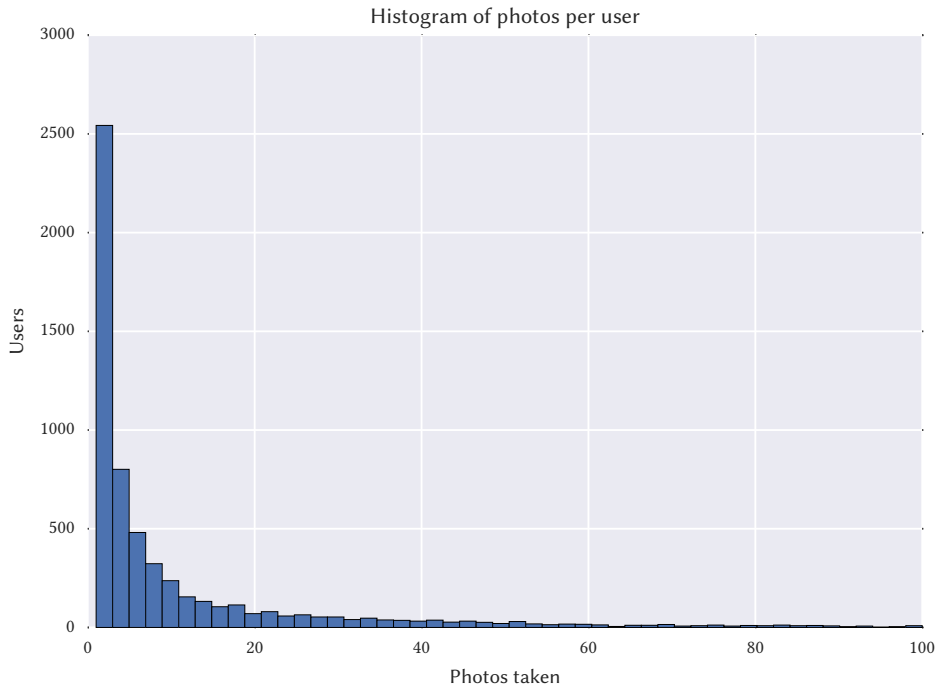


Figure 6.4: Histogram of photos taken per user.

Additional to the spatial coverage of the dataset, one interesting property to explore is how the photos are distributed across users. The histogram of photos taken by each user is shown in figure 6.4, this is cut off at 100 photos because the distribution has a heavy tail that is unimportant. This heavy tail is due to professional users like newspapers and event venues that have uploaded hundreds of photos under a single user, these users are not of particular interest for the use case of this work because they do not reflect the interests of a typical visitor. The histogram shows that most of the users uploaded one or two pictures which indicates that most of the photos come from very active users that take hundred of photos, this should be taken into account when analyzing the results.

6.2 Path Finding

The photos collected in the previous section were used to identify the important locations that are often photographed by many people. These locations define the ground set V for the models. In order to identify the popular locations automatically, the method proposed by Kleinberg et al. (2009) was used.

Kleinberg et al. (2009) define the problem of finding highly-photographed locationes as a clustering problem in a two-dimensional feature space, namely the space defined by the spatial coordinates of each photo. The clustering method used for this problem is mean-shift clustering, which was described in the background section.

Scikit[3] was used for the mean-shift clustering implementation. The clustering was performed on the complete photo dataset with the unnormalized latitude and longitude features using a gaussian kernel. The kernel's bandwidth was 0.001 degrees which approximately represents a radius of 100m without significant errors for the geographic scale of Zürich.

[3] `http://scikit-learn.org/stable/`

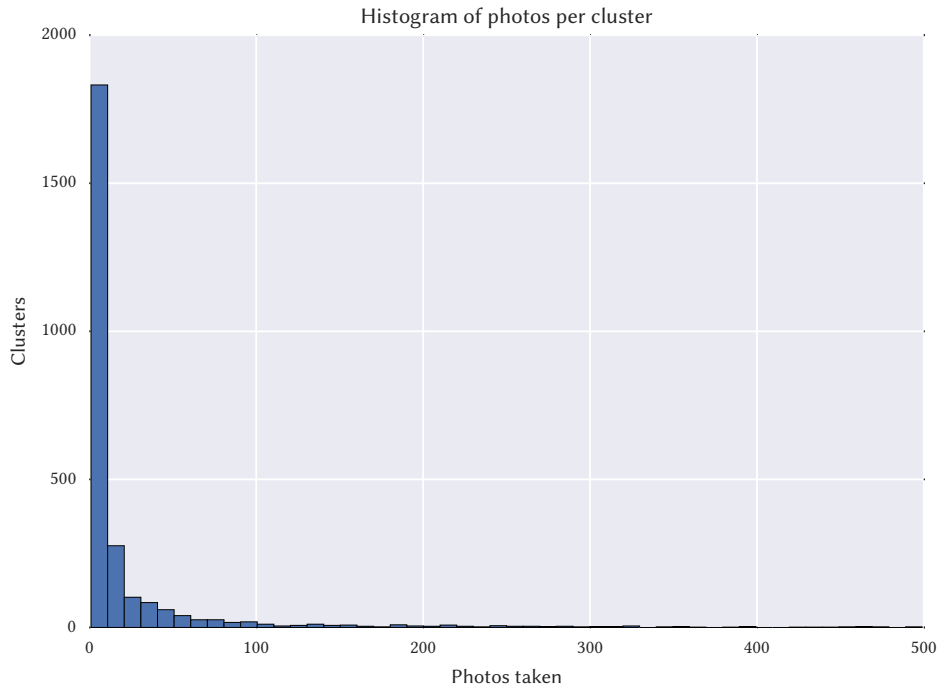


Figure 6.5: Histogram of the number of photos per cluster. The graph is cut off at 500 because the distribution has a heavy tail.

The clustering procedure yielded 2686 clusters covering 137019 photos, the left over photos represent outliers that are not inside any of the resulting kernels. Figure 6.5 displays a histogram of photos per cluster, it shows that over half of the clusters only contain one or two photos. This indicates that many of the clusters are actually not landmarks or highly photographed places, so in order to concentrate the model around the most interesting locations only the top N clusters sorted by the number of photos were used.
Two datasets were constructed, a small one with the top 10 clusters and a large one with the top 100 clusters. The small dataset was heavily used during the development of the models to test the implementation and try out different settings. On the other hand, the large dataset was mainly used for assessing the quality of the final models and comparing it against the baseline models.
Figures 6.6 and 6.7 show the selected clusters in the small and large setting, respectively. Additionally, Table 6.1 presents additional information about the top 10 clusters in the small dataset.
To extract the paths taken by individual users from the photo dataset, the procedure was as follows:

1. The photos were grouped by user and date when they were taken.
2. The photos in each user and date group represent a single path, ordered by date and time.
3. The photos were replaced in the sequences by their corresponding clusters, including the cluster only once. If the photo did not belong to one of the top clusters, then the photo would be discarded.
4. The resulting sequence of clusters represent the paths taken by users through the city.

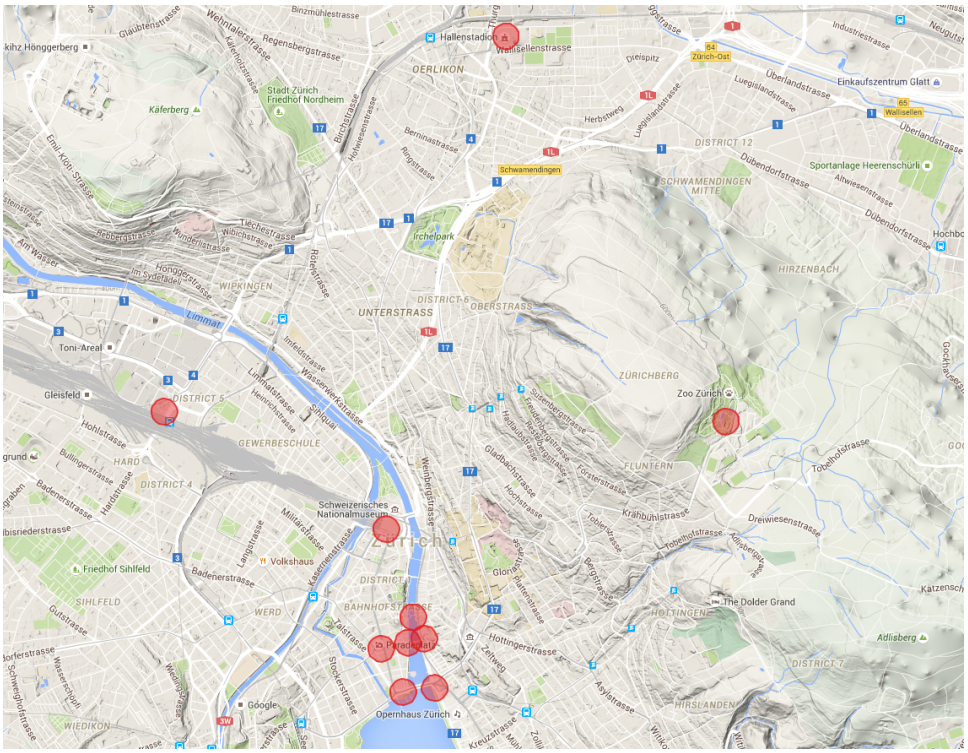


Figure 6.6: Top 10 clusters in Zürich according to the photo count.

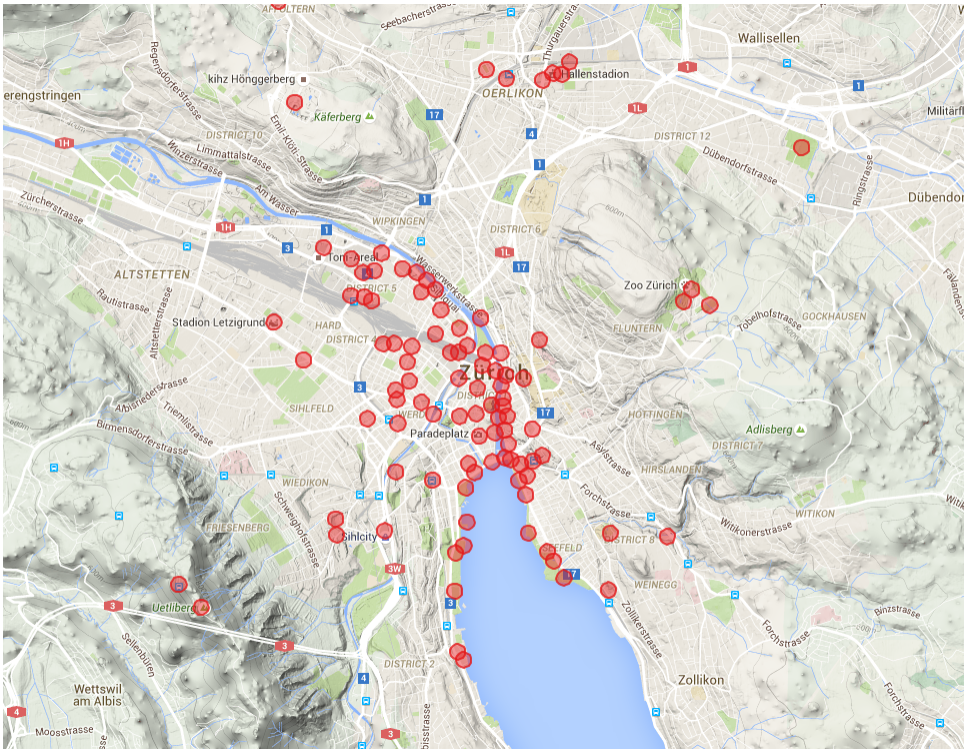


Figure 6.7: Top 100 clusters in Zürich according to the photo count.

Table 6.1: Top 10 clusters for Zürich.

Landmark	Photos taken	Unique users
Hauptbahnhof (Main train station)	4224	932
Grossmünster (Church)	3797	951
Bürkliplatz (Outdoors)	3728	612
Hallenstadion (Event venue)	3575	82
Fraumünster (Church)	3414	889
Bellevueplatz (Outdoors)	2584	437
Rathaus (Town hall)	2569	690
Zoo	2462	151
Paradeplatz (Outdoors)	2372	438
Prime Tower	1930	257

Table 6.2: Path dataset statistics for Zürich.

	Small ($\|V\| = 10$)	Large ($\|V\| = 100$)
Included photos (% of total)	30655 (18%)	91661 (54%)
Paths extracted, i.e. $\|\mathcal{D}\|$	6781	16439
Singleton paths, i.e. $\|S\| = 1$	5396	12927
Pair paths, i.e. $\|S\| = 2$	799	1980

These sequences were used as the unordered sets for the set models, e.g. FLID, FLDC, or as ordered sequences for the Markov model baseline. Table 6.2 show the statistics for the final datasets.
Additionally, Figures 6.8 and 6.9 show the histograms of the length of the sets in the small and large datasets, respectively. This shows that most of the sets are either singletons or pairs, which matches the fact that many users just take a few photos as it was shown in Figure 6.4.

6.3 Models

For the experiments, the FLID, FLDC and FFLDC models were used. For most of the experiments the following implementation details were fixed:

- The data was split in 8 random folds, with 87.5% of the data used for training and 12.5% for testing in each.
- The noise was generated using the log-modular model as described for the synthetic experiments.
- AdaGrad was used and η_0 was 1 and 0.1 for the small and large datasets, respectively.
- The number of epochs was 1000.
- The noise-to-data ratio ν was 5.

If any experiment uses a different setting, it will be stated along with the results.

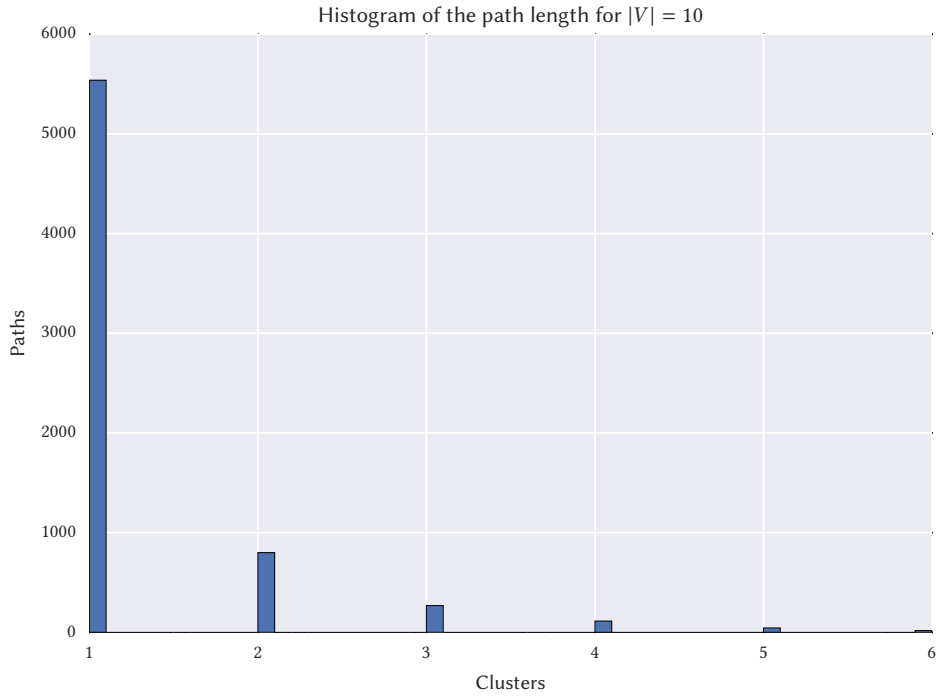


Figure 6.8: Histogram of set/path length for the small dataset.

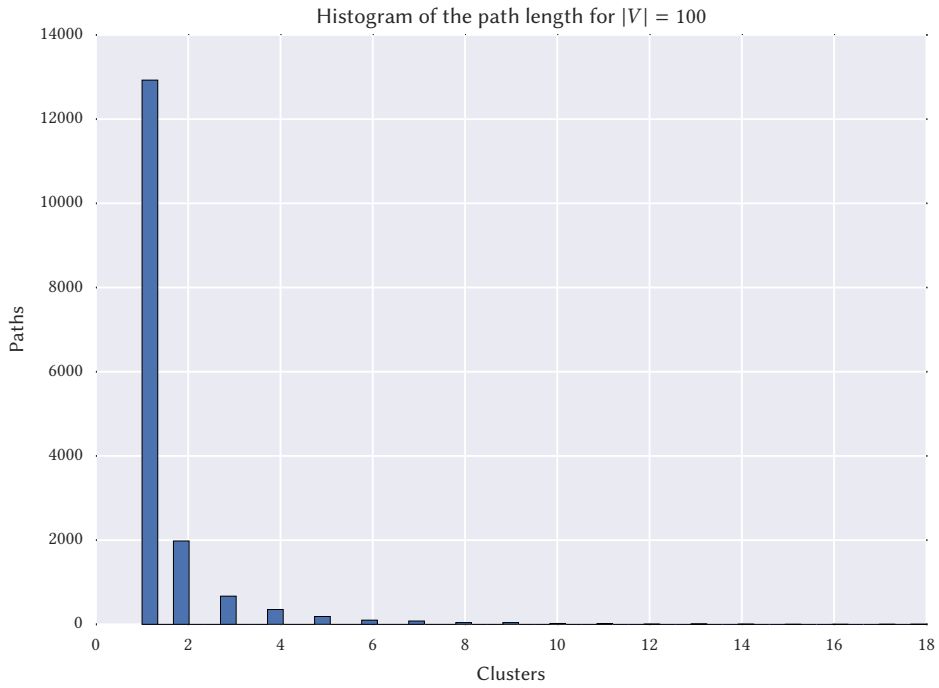


Figure 6.9: Histogram of set/path length for the large dataset.

6.4 Baselines

This section describes the baseline models that were used as a comparison for the proposed models.

6.4.1 Markov

A Markov chain can be used to model the behavior of Flickr users, assuming that each user selects the next location to visit and photograph only based on the current location, i.e. the Markov property. This model depends on the transition probabilities between the different landmarks in the dataset, concretely $P(l_{t+1} = i \mid l_t = j)$ where l_{t+1} represents the next location and l_t the current one. These probabilities can be estimated by counting all transitions in the dataset, i.e.

$$\hat{P}(l_{t+1} = i \mid l_t = j) = \frac{N(i,j)}{N(*,j)} \tag{6.1}$$

Where $N(i,j)$ is the number of times that the location i was photographed immediately after location j, and $N(*,j)$ is the number of times that any location was photographed after j. This model was used successfully by Kurashima et al. (2010) with good results for recommending tourists routes for a city based on Flickr data, therefore it is a natural choice for comparison against the newly proposed models. A full path can be constructed by selecting a starting location and choosing the next one according to the highest transition probability, then repeating the procedure until a desired length is reached.
Note that this model can be estimated efficiently from data, its running time is $O(|\mathcal{D}|)$.

6.4.2 Heuristic Markov

A strict Markov model can only consider one transition to make decisions, however a natural heuristic that can improve the quality of the model is to consider the complete path to avoid recommending locations that were already visited. This will be known as the Heuristic Markov model.
The transition probabilities are then defined as:

$$P(l_{t+1} = i \mid l_{1\ldots t}) = \begin{cases} 0 & \text{if } i \in l_{1\ldots t} \\ \frac{N(i,j)}{N(*,j)} & \text{otherwise} \end{cases} \tag{6.2}$$

This heuristic is included to provide a more fair comparison to the set models, e.g. FLID, which consider all the other elements in the set when recommending an element to include.

6.4.3 Proximity

Another, possibly naive, model is to assume that users photograph the next closest location to their current one. This can be useful to model paths that are concentrated in certain areas, for example in the Zürich dataset many of the paths only visit locations in the center and users seldom take photographs at the Zoo and a landmark in the center on the same day.
Formally, this model defines a series of distance $d(i,j)$ calculated from the coordinates of elements i and j. This can be used to predict paths that minimize the distance between elements. For example to predict the next element s_{t+1} in a sequence S_t, the element closest to s_t is selected, i.e. $s_{t+1} = \text{argmax}_{i \notin S_t}(d(s_t, i))$.
This is a natural choice for a baseline model because it provides a very quick model that is not completely random, the distances in this model can be computed in $O(|V|^2)$ time, which is less than $|\mathcal{D}|$ for both datasets. To construct a path, the same procedure as for the Markov chain can be used.

6.4.4 Log-modular

As described before, a log-modular model is the simplest log-submodular model which makes it a natural comparison for the FLID and FLDC models. The log-modular model can also be used with features by solving the linear system in Equation (5.1) to make it comparable to the FFLDC model.
The log-modular can be estimated from data following the procedure defined in Section 3.4.1, which only requires $O(|\mathcal{D}|)$ time.
This model is significantly different from the two previous baselines because it does not model the order of the elements, simply the membership. However, it is also possible to build a path by first constructing a set of fixed size n and then ordering the elements in order to minimize the total travel distance. A set of size n can be constructed by selecting the n locations with highest utility, i.e. $\text{argmax}_i\, u_i$.

6.5 Evaluation

The evaluation of the models aims to assess their quality for the task of recommending additional locations to visit based on an initial set of locations that the user wants to visit. This recommendation can be structured as a path either directly from the output of the model, e.g. for the Markov chain case, or as a post-processing step, i.e. for the set based models. The evaluation procedure focuses on one-step recommendations which can be used both to construct a full path greedily or for an interactive system where the user requests a single location at each step.
The evaluation task is to predict the missing location in a sequence given the other elements, both before and after. This can be constructed by removing one element at a time from the paths in the test dataset. Thus each path of length k generates k partial paths for testing with a missing element. For example the path $1, 5, 6$, generates 3 partial paths, namely $?, 5, 6$, $1, ?, 6$ and $1, 5, ?$. Note that only paths of at least size 2 are used in the task.
After each model is trained with the corresponding training dataset, the predictions for each path $S_t = [s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_t]$ where S_j is missing, are made as follows:

FLID, FLDC, FFLDC, Log-modular The element with maximum probability, that is not already in the path is predicted, i.e. $\text{argmax}_{i \notin S} P(S \cup \{i\})$ where S is the set of the elements in the path S_t.

Markov, Heuristic Markov The element with the highest probability given s_{j-1} and s_{j+1} is predicted, i.e. $\text{argmax}_i P(s_{j+1} \mid i) P(i \mid s_{j-1})$. When using the heuristic the prediction becomes, $\text{argmax}_{i \notin S_t} P(s_{j+1} \mid i) P(i \mid s_{j-1})$

Proximity Using the unordered set S of the elements in S_t, predict the element closest to all elements in the set, i.e. $\text{argmin}_{i \notin S} \sum_{j \in S} d(i, j)$.

Proximity Ordered Similar to the previous model, but considering the order. The element closest to $j - 1$ and $j + 1$ that is not already in the path is predicted, i.e. $\text{argmin}_{i \notin S_t} d(j - 1, i) + d(i, j + 1)$.

The metric used for the quality of the predictions is overall accuraccy, i.e. the percentage of correct predictions. Formally,

$$Acc_{model} = \frac{1}{|\mathcal{T}|} \sum_{S_t \in \mathcal{T}} (\mathbb{I}_{s_j(i)}) \tag{6.3}$$

Where $\mathcal{T}$ is the set of all partial paths available for testing and $\mathbb{I}_{s_j}$ is an indicator function that evaluates to 1 when the predicted element i is equal to the missing element in the sequence s_j and 0 otherwise.

Accuracy is the most used metric for recommender systems (Herlocker et al., 2004) and it was the metric used by Kurashima et al. (2010) for evaluating single step recommendations in tourist routes, although they only perform forward evaluation because its dealing only with sequence models such as the Markov Chain. This accuracy metric also is similar to the leave-one-out procedured used by Tschiatschek et al. (2016) to evaluate the FLID model for product recommendation. Therefore, it is expected to provide a good assesment on the quality of the proposed models for this particular task while balancing the fact that some models work on paths and others deal with sets. The accuracy results will be presented as the average of the various data folds with 95% confidence intervals, unless stated otherwise.
An alternative metric is the log-likelihood relative improvement (LLRI) as proposed by Tschiatschek et al. (2016). This measures the model fit on the test dataset and indicates its relative improvement over the log-likelihood of the log-modular model. Formally,

$$\text{LLRI} = 100\frac{\mathcal{L}_{model} - \mathcal{L}_{log-modular}}{|\mathcal{L}_{log-modular}|} \tag{6.4}$$

Where $\mathcal{L}_{log-modular}$ is the log-likelihood of the test dataset for the log-modular model and $\mathcal{L}_{model}$ for the evaluated model, e.g. FLID, FLDC.

Chapter 7

Modeling an Image Collection - Results

This chapter presents the results for different experiments performed on the datasets introduced in the previous chapters. Different models are evaluated and their outcome is discussed.

7.1 The Problem with Singletons

The histograms in Figures 6.8 and 6.9 show that most of the path datasets are composed on singleton paths representing users that only took photos around a small area on a single day. However, the evaluation procedure described in Section 6.5 only deals with paths of at least 2 items, this represents a problem because the kind of data used for learning is not the same as for the evaluation phase. Moreover, initial experiments showed that the distribution of singleton paths is significantly different from that of paths with 2 or more elements.
In order to illustrate this, a special dataset only containing paths of length 2 or more was constructed. The log-modular model was then trained and evaluated on both the original and special dataset. This was done for both the small ($|V| = 10$) and large ($|V| = 100$) datasets.
The accuracy of the resulting models for the test data is shown in figure 7.1. It shows that there is a significant improvement in the accuracy when only the distribution of non-singleton paths are considered for the log-modular model. This effect is also expected for the more complex models, e.g. FLID, because their initial point is based on the log-modular model. Note that for Markov there is no difference between including singletons or not because transition probabilities can only be learned from paths of 2 or more elements, on the other hand the Proximity model is also agnostic to these differences because the model is estimated from the spatial features of the elements and not the data about paths. Because of these results, the rest of the experiments will be performed on the modified datasets without singletons because those paths are of more interest than invidual photo spots.

7.2 Small Dataset

This section presents the experiments performed on the small dataset. These show the ability of the proposed models to offer accurate predictions for completing paths based on the data. Additionally, the effect of the number of dimensions is explored in order to find the best possible model.

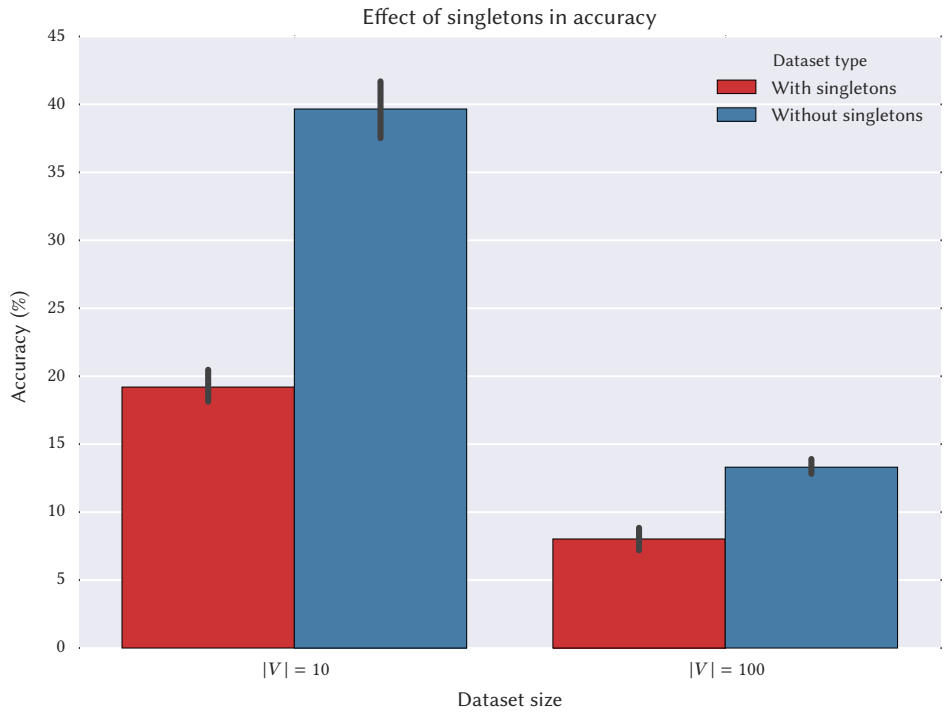


Figure 7.1: Accuracy of log-modular models for different dataset configurations.

First, the results from the baseline models are presented in Figure 7.2. These show that the log-modular model outperforms the Markov baselines in this case. This is encouraging as it serves as the base for the proposed models. The results also show that the heuristic Markov model outperforms the strict Markov, this is expected in all cases because the dataset was constructed in such a way that elements are not repeated, hence the proposed heuristic always improves the prediction by removing the already visited elements from the possible choices. The proximity model is the best for the small dataset, and the results show that adding ordering information improves the score.

7.2.1 Diversity & Coherence

The first approach to the problem was to use the FLID model, i.e. modeling only diversity. Figure 7.3 shows the model accuracy for different number of dimensions. Comparing these values with the log-modular model shows that there is no significant improvement even with a large number of diversity dimensions. Increasing the number of noise samples or epochs did not improve these results either.

It is possible that for this dataset, diversity is not sufficient to capture the underlying distribution. Therefore, the next step was to use the FLDC model. After executing a grid search for $0 \geq L \geq 10$ and $0 \geq K \geq 0$ with the FLDC model the accuracy did not improve either.

The fact that the FLID and FLDC models do not result in improved accuracy with respect to the log-modular baseline can be explained by looking at the distribution of the paths in the data. The top 10 paths and their proportions in the dataset are listed in Table 7.1.

The paths in Table 7.1 are all pairs, which are harder to predict because there is only information about one item and there is no benefit from diversity in that case. Simply ordering by frequency should give a high accuracy, as shown by the log-modular model's accuracy in Figure 7.2.

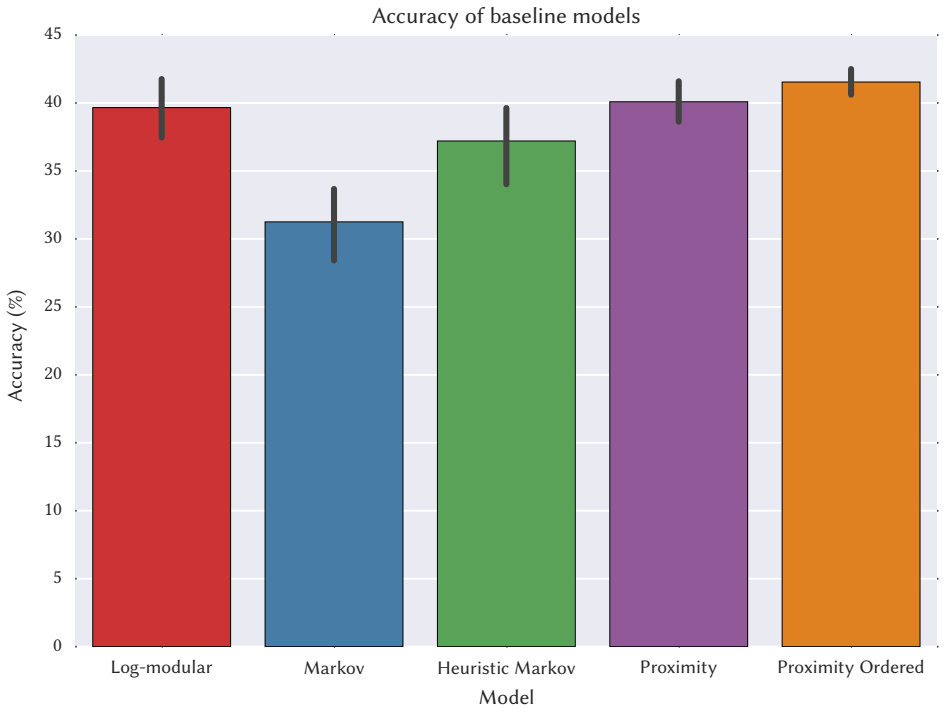


Figure 7.2: Accuracy of baseline models for the small dataset.

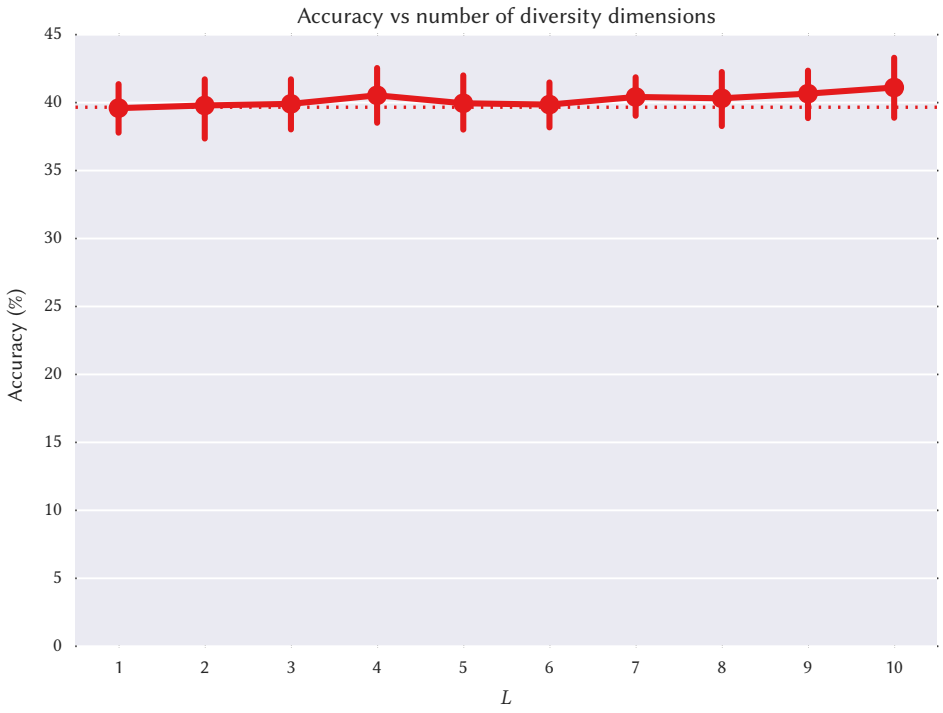


Figure 7.3: Accuracy for various values of L. The dotted line shows the mean accuracy of the log-modular model.

Table 7.1: Most frequent paths in the small Zürich dataset.

S	$N(S)$
$[1,4]$	71(5.7%)
$[4,1]$	58(4.7%)
$[6,1]$	47(3.8%)
$[6,4]$	43(3.5%)
$[1,6]$	37(3.0%)
$[4,6]$	33(2.6%)
$[2,4]$	26(2.1%)
$[2,5]$	25(2.0%)
$[5,2]$	24(1.9%)
$[4,8]$	23(1.8%)

However, the lack of improvement in accuracy does not mean that the FLID and FLDC models are not learning beyond the log-modular model. This can be seen by comparing the total variation distance between the empiric distribution of the paths, after removing the ordering, and the distributions produced by the learned models. Note that in this case the number of items is small enough that the computation of the partition function is tractable even using the exhaustive method that requires $O(2^{|}V|)$ time. Figure 7.4 shows the total variation distance for differnt FLID and FLDC models. It shows that combining diversity and coherence improves the approximation of the model to the empiric distribution, as a comparison the TVD for the log-modular model was 0.40.

Finally, Figure 7.5 shows the LLRI metric for the FLDC models. This shows more clearly that there is an improvement over the log-modular model, even though it is not reflected on the accuracy metric. The best choice of dimensions is $L = 5, K = 5$ according to both the LLRI and TVD metrics.

7.3 Large Dataset

The baseline models were trained on the large dataset, i.e. $|V| = 100$, and the accuracy results are presented in Figure 7.6. In this case, there is a more significant difference between the log-modular and heuristic Markov models. This indicates that simply choosing the most frequent element is not as good of a strategy as for the small dataset, therefore complex models like FLDC and FFLDC should yield better results. Finally, the significant higher score of the Heuristic Markov and Proximity Ordered models shows how adding information about the ordering, i.e. using paths, can improve the results in comparison to the models that work only on sets.

Different FLDC models were trained on the large dataset, in order to find the best setting for the dimensions L and K a grid search was performed. The average accuracy for each setting is presented in Figure 7.7. This shows that the FLDC model is significantly better for the path completion tasks than the log-modular baseline, however there is not a visible difference between the various settings for the model. These results suggest that models with larger number of dimensions have worse accuracy in the test set, this is likely due to the large number of parameters in the model which require more data to estimate properly.

Finally, the best FLDC model is compared to the baselines in Figure 7.8. Even though the FLDC model is an improvement over the log-modular, it still does not reach the accuracy of the Markov or Proximity Ordered models. This can be explained by the fact that the first three models in the figure include information about the order of the elements in a

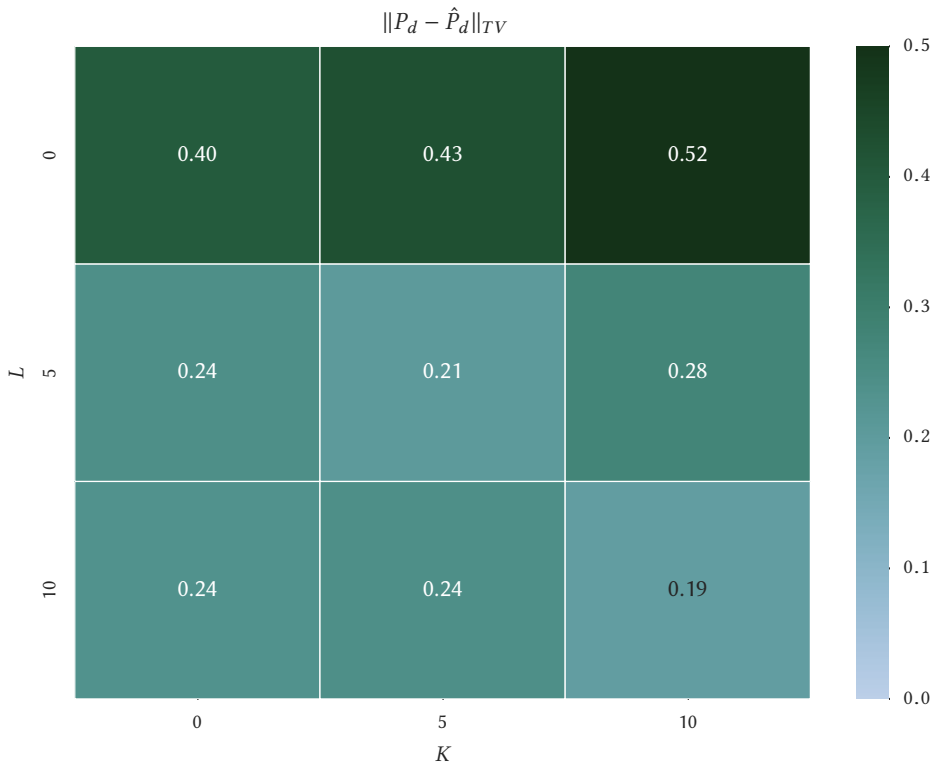


Figure 7.4: Total variation distance for FLDC models on the small dataset.

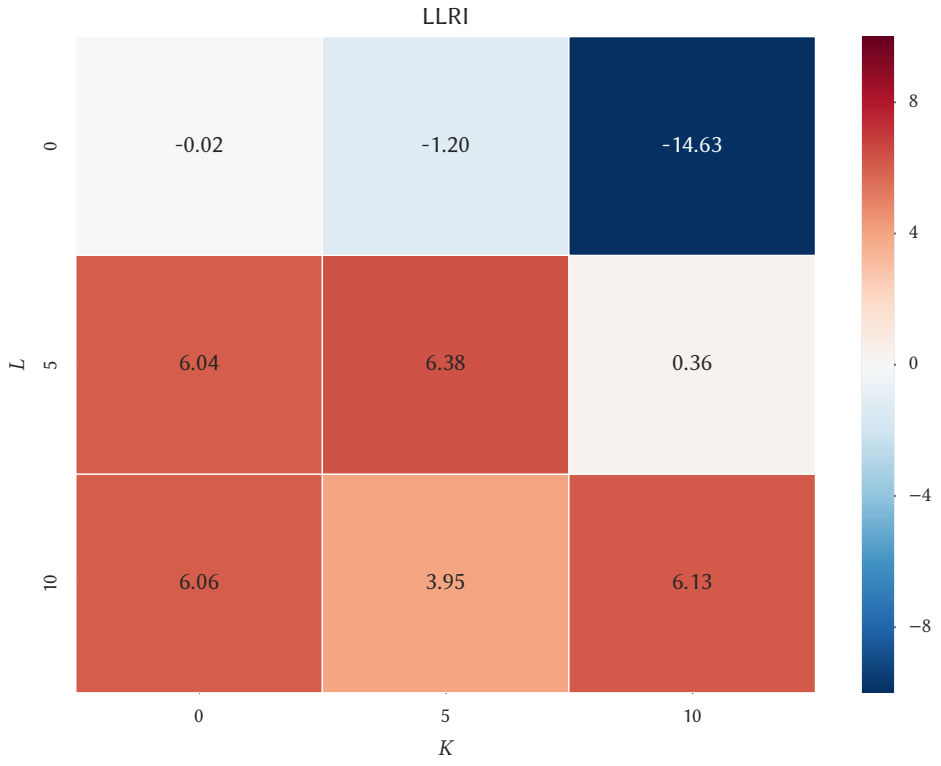


Figure 7.5: Log-likelihood relative improvement (LLRI) for FLDC models on the small dataset.

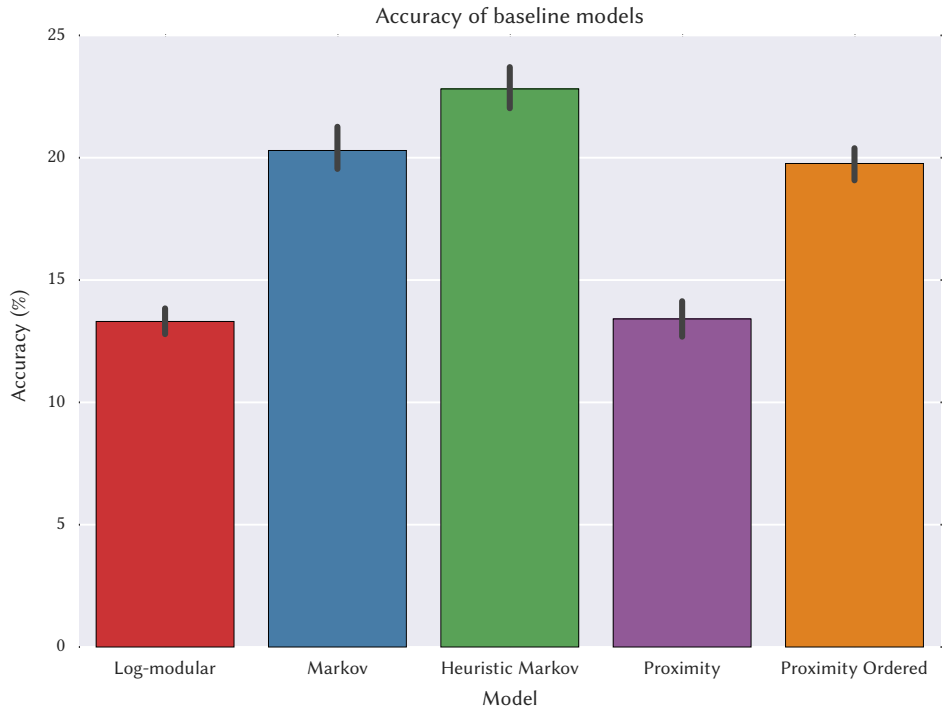


Figure 7.6: Accuracy of baseline models for the large dataset.

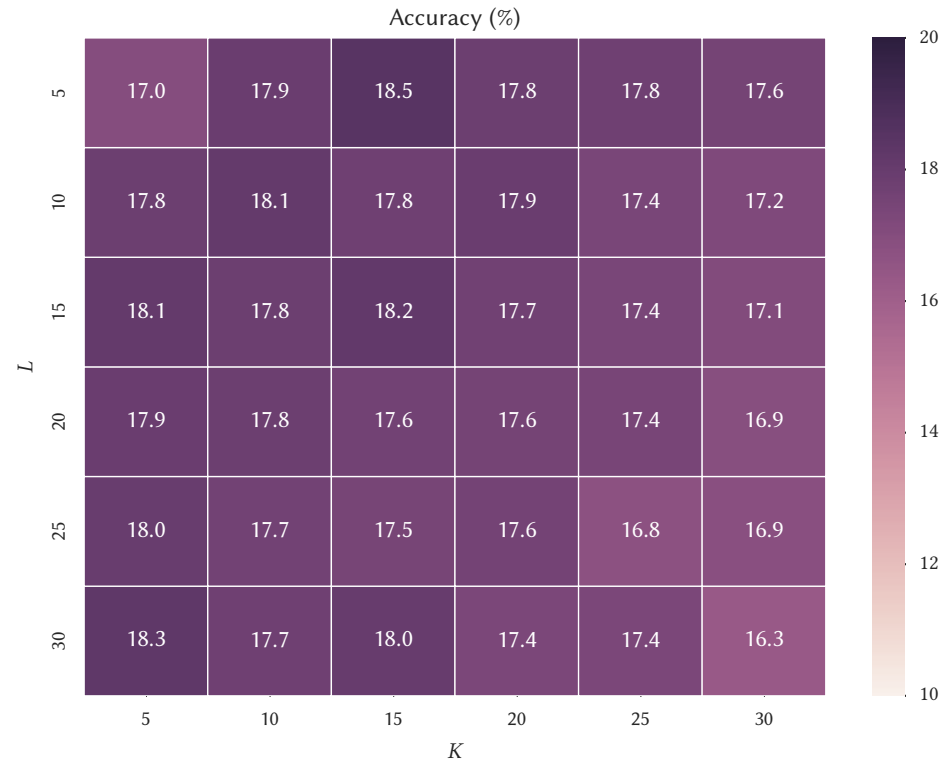


Figure 7.7: Accuracy of FLDC models with varying L and K values.

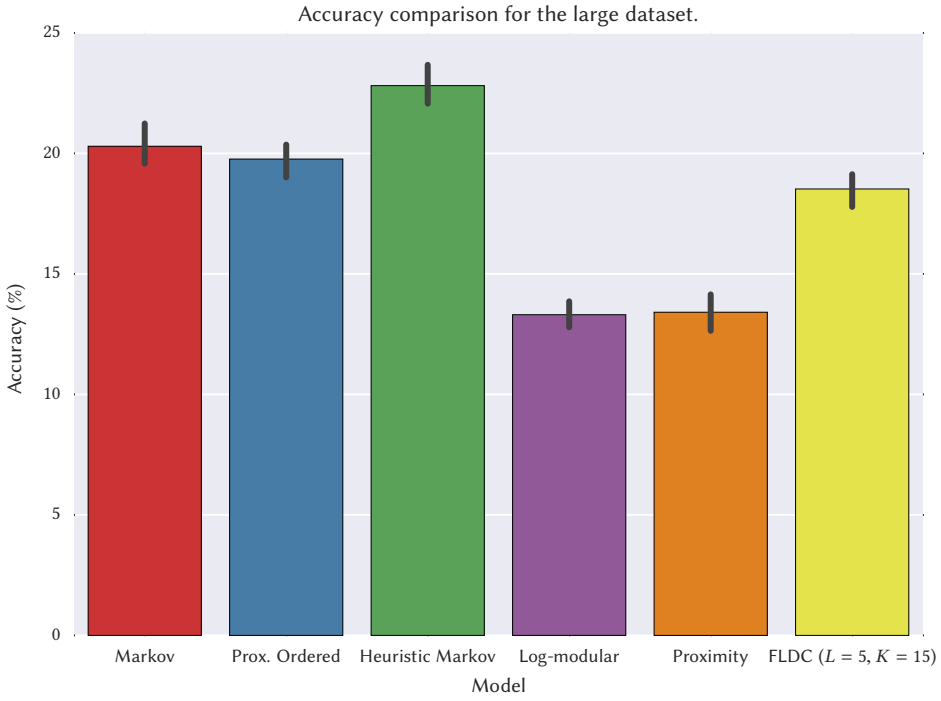


Figure 7.8: Comparison of baseline models to the best FLDC model for the large dataset.

set, i.e. they model probabilities other paths, while the others only model membership of the elements in a set. This extra information helps these models perform the prediction task more accurately.
Additional experiments with features were performed, aiming to enrich the model with information such as number of photos taken in a location and number of users. However, this did not show improvement in comparison to the FLDC model.

Chapter 8

Conclusion

Bibliography

Amir, R. (2005). Supermodularity and complementarity in economics: an elementary survey. *Southern Economic Journal*, pages 636–660.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics*, pages 177–186.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer.

Bruno, W. J., Rota, G.-C., and Torney, D. C. (1999). Probability set functions. *Annals of Combinatorics*, 3(1):13–25.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619.

Darken, C. and Moody, J. (1990). Note on learning rate schedules for stochastic optimization. In *Neural Information Processing Systems*, pages 832–838.

Darken, C. and Moody, J. (1992). Towards faster stochastic gradient search. In *Neural Information Processing Systems*, pages 1009–1016.

Djolonga, J. and Krause, A. (2014). From MAP to marginals: Variational inference in bayesian submodular models. In *Neural Information Processing Systems (NIPS)*.

Djolonga, J. and Krause, A. (2015). Scalable variational inference in log-supermodular models. In *International Conference on Machine Learning (ICML)*.

Donaire, J. A., Camprubí, R., and Galí, N. (2014). Tourist clusters from Flickr travel photography. *Tourism Management Perspectives*, 11:26–33.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

Jerrum, M. and Sinclair, A. (1990). Polynomial-time approximation algorithms for the ising model. In *Automata, Languages and Programming*, pages 462–475. Springer Berlin Heidelberg.

Kleinberg, J., Huttenlocher, D., Backstrom, L., and Crandall, D. (2009). Mapping the World's Photos. In *Proceedings of the 18th International Conference on World Wide Web*, pages 761–770.

Krause, A. and Golovin, D. (2014). Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, chapter 3. Cambridge University Press.

Kurashima, T., Iwata, T., Irie, G., and Fujimura, K. (2010). Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 579–588.

Levin, D. A., Peres, Y., and Wilmer, E. L. (2008). *Markov chains and mixing times*. American Mathematical Society.

Shahaf, D., Guestrin, C., and Horvitz, E. (2012). Trains of thought: Generating information maps. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 899–908, New York, NY, USA. ACM.

Tschiatschek, S., Djolonga, J., and Krause, A. (2016). Learning probabilistic submodular diversity models via noise contrastive estimation. In *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Yan, R., Wan, X., Otterbacher, J., Kong, L., Li, X., and Zhang, Y. (2011). Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 745–754, New York, NY, USA. ACM.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 116–123, New York, NY, USA. ACM.