



Department of Computer Science

# On Tour: Harnessing Social Tourism Data for City and Point-of-Interest Recommendation

Thomas David Bewley

---

A dissertation submitted to the University of Bristol in accordance with the requirements of  
the degree of Master of Science in the Faculty of Engineering

---

September 2019 | CSMSC-19



0000058853

# **Declaration:**

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Thomas David Bewley, September 2019

## Executive Summary

Tourism is a favoured activity for millions worldwide, with cities being a popular destination type. For a group of travellers with divergent preferences, the choice between many candidate cities and within-city points of interest (POIs) may be daunting. Recommender systems can support such decision making, informed by knowledge of tourists' preferences and previous travel history. In this project, I develop a variety of data-driven models for the ranking of candidate city destinations and POIs. The models are implemented with a novel dataset of 1.3 million POI visits across 200 cities, which I construct from the *Flickr YFCC100M* dataset of photo metadata in concert with *OpenStreetMap*.

I propose three models for city recommendation based on collaborative filtering, content-based relation of candidate cities to those already visited, and matching of cities to tourist's preference distributions. I also investigate two methods for aggregating the group's diverse preferences. For POI recommendation I consider only a single target tourist, and frame the problem as that of ranking POIs to visit at a designated time during an ongoing city holiday. I develop six numerical features to quantify the suitability of candidate POIs based on overall popularity, the target tourist's known preferences, contextual factors (location, time and date) and historic correlations with previously-visited POIs. I investigate three methods for mapping features into recommendation scores: simple summation, linear regression and a feedforward neural network.

In cross-validation experiments I assess the models' ability to predict tourists' true travel decisions. For city recommendation, the collaborative filtering model performs strongest by a wide margin, and performance further improves as the group size is increased. For POI recommendation, even the simple summation of features proves highly effective, though small gains can be attained by using a neural network for scoring. On both problems, all of my models outperform a variety of more naïve baselines.

The key contributions of this project are as follows:

- **Concept:** A recommender system that uses the same social dataset, and many of the same derived features, to perform both city- and POI-level recommendation.
- **Dataset:** A novel dataset of travel histories synthesised from *Flickr* and *OpenStreetMap* data, larger than any similar set that could be found in existing work.
- **Models:** Several simple collaborative filtering and content-based approaches to the city recommendation problem. For POI recommendation, a selection of preference- and context-sensitive features for quantifying POI suitability, which can be mapped into a single score for ranking.
- **Results:** Consistent baseline-beating performance on cross-validation experiments.

The complete working directory for this project can be found [here](#) (expires 05/12/19), and a *GitHub* repository containing only the essential data and scripts is available [here](#).

## Acknowledgements

I would like to thank my supervisors Iván Palomares Carrascosa and Ercan Ezin for their support and guidance throughout this project, as well as Rui Ponte Costa for his feedback on an earlier draft of this thesis, and numerous fellow students and friends, whose helpful contributions to discussions of my ideas have helped to clarify my thinking. Extra special thanks also go to my family and girlfriend Tash, all of whom have shown boundless care and understanding, and may now be wary of ever holidaying again for fear of unsolicited point-of-interest recommendations.

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
1.1	Aims and Objectives . . . . .	9
<b>2</b>	<b>Background and Context</b>	<b>10</b>
2.0.1	Recommender System Basics . . . . .	10
2.0.2	Types of Recommender System . . . . .	11
2.0.3	Recommendation for Groups . . . . .	13
2.0.4	Recommendation for Tourism . . . . .	14
2.1	Flickr Data for Tourism Recommendation . . . . .	16
2.2	Summary of Implications . . . . .	18
<b>3</b>	<b>Overview of Recommendation Models</b>	<b>19</b>
3.1	City Recommendation . . . . .	19
3.1.1	Notation . . . . .	19
3.1.2	Problem Definition . . . . .	21
3.1.3	Ranking by Tourist-Tourist Similarity . . . . .	21
3.1.4	Ranking by City-City Similarity . . . . .	22
3.1.5	Ranking by Tourist-City Similarity . . . . .	23
3.1.6	Extension to Multi-Tourist Groups . . . . .	23
3.2	Point-of-interest Recommendation . . . . .	24
3.2.1	Notation . . . . .	25
3.2.2	Problem Definition . . . . .	26
3.2.3	Model Components . . . . .	27
3.2.4	Overall Popularity Feature . . . . .	27
3.2.5	Preference-aware Feature . . . . .	28
3.2.6	Context-aware Features . . . . .	28
3.2.7	History-aware Feature . . . . .	31
3.2.8	Scoring Methods . . . . .	32
3.2.9	Extension to Multi-Tourist Groups . . . . .	33

<b>4 Creation of Travel Histories Dataset</b>	<b>34</b>
4.1 <i>YFCC100M</i> and <i>OpenStreetMap</i> . . . . .	34
4.2 Implementation Notes . . . . .	35
4.3 Data Preprocessing . . . . .	35
4.4 Visit Creation and Labelling . . . . .	36
4.4.1 Grouping Photos into Visits . . . . .	36
4.4.2 Obtaining POIs from <i>OpenStreetMap</i> . . . . .	37
4.4.3 Labelling Visits with POIs . . . . .	38
4.5 Label Bootstrapping via Word Likelihood Ratios . . . . .	39
4.6 Final Dataset . . . . .	41
4.7 Data Quality Assessment . . . . .	43
4.7.1 Manual Inspection . . . . .	43
4.7.2 Statistical Properties . . . . .	45
4.7.3 Comparison to Existing Dataset . . . . .	50
<b>5 Application of Models to Dataset</b>	<b>52</b>
5.1 Implementation Notes . . . . .	52
5.2 Construction of Data Structures . . . . .	52
5.2.1 City Recommendation . . . . .	52
5.2.2 POI Recommendation . . . . .	54
5.2.3 Algorithmisation of Models . . . . .	55
5.2.4 City Recommendation . . . . .	55
5.2.5 POI Recommendation . . . . .	58
<b>6 Evaluation and Optimisation Methods</b>	<b>61</b>
6.1 Evaluation Methods . . . . .	61
6.1.1 City Recommendation . . . . .	61
6.1.2 POI Recommendation . . . . .	62
6.2 Optimisation of Machine Learning Models for POI Scoring . . . . .	64
<b>7 Results and Discussion</b>	<b>67</b>
7.1 City Recommendation . . . . .	67
7.1.1 Single-Tourist Results . . . . .	67
7.1.2 Group Results . . . . .	71
7.1.3 Summary of Findings . . . . .	76
7.2 POI Recommendation . . . . .	77
7.2.1 Optimisation Process . . . . .	77
7.2.2 Performance Comparison . . . . .	80
7.2.3 Correlation Analysis . . . . .	81
7.2.4 Performance Patterns . . . . .	83

7.2.5	Summary of Findings . . . . .	85
<b>8</b>	<b>Conclusion</b>	<b>87</b>
8.1	Summary of Outcomes . . . . .	87
8.2	Summary of Contributions . . . . .	89
8.3	Areas for Improvement and Future Work . . . . .	89

# Chapter 1

## Introduction

Tourism is a popular activity for many millions of people worldwide. In 2017, 1.33 billion international tourist visits were recorded, generating around US\$1,340 billion in receipts [1]. Over half of these visits took place in Europe, where city breaks are a common form of holiday, especially in ancient cultural centres such as London, Paris, Rome and Barcelona. For a group of prospective travellers, the challenge of choosing satisfactorily between the many potential city destinations can be a daunting one, particularly when their opinions and preferences diverge. In addition, when the group arrives in a new city, they face countless decisions about which points-of-interest (POIs) to visit, from bars and cafés to museums and natural sights, and in what order.

Modern computational technologies, enabled by the vast quantity of data available on the World Wide Web, are capable of assisting travellers in these decision making processes. In the existing literature, a range of *recommender systems* have been developed to suggest both high-level destinations and specific POIs to individuals and groups, informed their explicitly- or implicitly-defined preferences and the experiences of past visitors. An increasing proportion of these operate by exploiting trends in large social datasets, such as those provided by the *Flickr* photo sharing platform. However, the reported results have been limited to only a small number of target cities, and have concerned only POI recommendation alone. No attempt has yet been made to perform both city- and POI-level recommendation using the same underlying dataset. In addition, relatively little effort has been dedicated to developing robust techniques for inferring accurate travel histories from the raw social data, and the accuracy of the derived datasets has not been questioned or investigated. In this project, a variety of recommender systems are designed, built and evaluated that build upon the successes of past tourism recommendation tools, and address each of the above previously-underexplored areas.

## 1.1 Aims and Objectives

The overall aim of this project is to develop a set of recommendation tools to assist groups of travellers in the planning of city holidays, using both personalised data and population-wide statistics from a novel dataset of travel histories. This dataset is constructed by combining data from *OpenStreetMap* with that from *YFCC100M*, the world’s largest publicly-available media collection, which features accurate time and location tags for tens of millions of photos uploaded to the *Flickr* photo sharing platform. Two specific problems are addressed: city recommendation for groups of travellers, and within-city POI recommendation for a single traveller. In both cases, the output is a ranking of all available options, from most- to least-recommended.

This thesis is organised to correspond with the following list of project objectives:

- Understand a broad range of techniques for both individual and group recommendation, especially those previously applied in the tourism context.
- Formalise the city recommendation problem for both the single-tourist and group case, and develop a selection of simple models for solving it using both collaborative filtering and content-based techniques.
- Formalise the POI recommendation problem for the single-tourist case, and develop a single hybrid model for solving it, that extracts a number of context-sensitive features of a candidate POI and aggregates them into an overall score.
- Design and implement a pipeline for augmenting the *YFCC100M* dataset with information from *OpenStreetMap* to create a dataset of POI-level travel histories, and thoroughly verify the accuracy of this dataset.
- Implement all models for use specifically with the travel histories dataset. For the POI recommendation model, this involves training a number of machine learning models.
- Evaluate all models by cross-validating against unseen travel experiences from the dataset, with the aim of predicting tourists’ true decisions. Identify the best-performing variants and investigate areas of strength and weakness.

The added value of this project is twofold. Firstly, the novel two-part system (both city and POI recommendation) lays the foundations for an end-to-end solution for travel planning, from initial destination selection to flexible, real-time decision making during the trip itself. Secondly, and independently of the success of the recommender models themselves, the derived *Flickr-OpenStreetMap* dataset of travel histories is larger than any produced as part of prior work, and has the potential to provide a new depth of insight into real-world tourism patterns in future.

# Chapter 2

## Background and Context

This project concerns the development of both group and single-user recommender systems for application in the tourism domain, which harness data from social media to infer both individual preferences and population-wide trends. In this chapter, I present an overview of the core concepts underlying the approach, and existing work published in this space.

### 2.0.1 Recommender System Basics

A recommender system is a decision support system that assists a user in the selection of objects, actions or experiences (generically called *items*) from a large space of possibilities, usually by inferring something about their relevant preferences or needs [2]. Because the user must engage with the system to provide such information, and almost always has the final say on whether real-world action is taken as a result of the items selected, the recommendation process can be viewed as a collaborative effort between user and software [3]. The final output of the system may be a single recommended item, a shortlist, or a complete ranking of all items from most- to least-recommended. The latter output is the most general, since the others can readily be derived from it. Recommender systems are accepted as a mature and effective technology in many industries, and are employed by technology companies to suggest music [4], videos [5] and products [6] to users browsing online platforms, curate social network news feeds [7] and assist in romantic matchmaking [8], among many other applications. Their importance is destined to only grow in an age of increasing information overload, as a solution to the ‘paradox of choice’ problem.

A core conceptual component of most personalised recommender systems, which may or may not be instantiated as a tangible data structure, is the *user-item ratings matrix*  $m$ , which has a row for each user  $u$  and a column for each item  $i$  [2]. In the most general terms, the value  $m_{u,i}$  reflects the utility of  $i$  for  $u$ . Depending on the kind of recommendation algorithm, the matrix is used in various ways, and in concert with various other sources of information, to inform the choice of items to recommend. As users participate in the

system, consuming or engaging with some of the available items, they may voluntarily offer feedback on their experiences by providing scores or reviews – in this case, ratings are said to be obtained *explicitly* and can be entered directly into the matrix. However, in many contexts users may not be willing or able to provide explicit feedback, so ratings must be inferred from *implicit* data such as the number of discrete interactions with an item (e.g. mouse clicks) or the time spent viewing it [9]. The construction of reliable ratings from implicit data is generally far more difficult, and the producers of advanced recommender systems, such as those at the heart of modern social networks, are typically incentivised to collect ever more data about their customers to inform highly complex preference models. Another key challenge for recommender systems is the *cold-start problem*, which arises whenever a new user joins the system and there is a scarcity of information about their preferences [10]. Many of the most innovative developments in recommendation algorithms seek to mitigate this problem.

### 2.0.2 Types of Recommender System

Recommendation algorithms can be categorised into a number of basic types, each of which views the task of selecting items to recommend from a unique perspective and harnesses a particular set of information about the populations of users and items.

#### Collaborative Filtering

Collaborative filtering seeks to simulate the effect of word-of-mouth promotion within a population of users [3]. By default, according to the preceding definition, a user-item ratings matrix only contains non-zero values for the small fraction of items that each user has *already* interacted with, and is entirely uninformative with respect to any unseen items. Choosing between these unseen items is usually the target of recommendation, and to enable this the remaining cells in the matrix must be populated. A collaborative filtering algorithm solves this problem by identifying other users in the population with a similar set of existing ratings to the target user (often called the *neighbourhood*), and applying the assumption that their preferences will continue to align in future. As a basic example: if two users *Alice* and *Bob* have both given item *x* a high rating and item *y* a low rating, and *Alice* has also given a third item *z* a high rating, then a collaborative filtering algorithm would assume that *Bob* will do the same, and in turn may be likely to recommend *z* to *Bob* in future. Many valid metrics exist for quantifying the similarity between two users' sets of ratings, with the Pearson correlation coefficient being the most widely used [2]. The choice of neighbourhood size is also an important consideration. Too small, and the outcome may be too sparse or statistically insignificant; too large, and the degree of personalisation and relevance is reduced.

## **Content-based Recommendation**

Whereas collaborative filtering requires no semantic knowledge about items, content-based recommendation works by assigning keywords to items and modelling each user's preferences with respect to those keywords [11]. In their purest form, content-based methods do not incorporate any consideration of the actions of other users within the system. By relating new items to previously-consumed ones in terms of their keywords, predictions can be made about the user's likely ratings of them. A major challenge of content-based recommendation is that users still rate on an item-by-item basis, and it can be difficult to disentangle preferences over keywords from the highly specific features of individual items, especially when the keywords themselves are ambiguous or subjective. In addition, for a large population of items it may be very labour-intensive to manually label every item with its associated set of keywords.

## **Constraint-based recommendation**

Constraint-based recommendation requires an even deeper semantic knowledge about the set of items, with each being assigned a number of properties and parameters. During the search, enquiry or browsing process, the user may manually specify their requirements (e.g. price, category, features) which are represented as constraints to screen out some proportion of the available items [12]. Alternatively, the requirements themselves might be inferred from the user's past behaviour, although this is a risky strategy unless a high degree of certainty exists. Those items that remain after filtering are ranked by some measure of utility or relevance, which could be derived through collaborative or content-based means. If no items are able to satisfy all the requirements, the challenging problem arises of how best to relax the specified constraints without sacrificing the spirit of the user's query; there is rarely a single solution [3].

## **Critiquing-based recommendation**

Critiquing-based recommendation frames the item selection process as a cyclic interaction between the system and the user. The recommender proposes a sequence of items, and the user either accepts an item or scores it according to the perceived value of some attribute (e.g. attractiveness, comfort, bulkiness) [13]. Whether a lower or higher attribute value is better is context-dependent; sometimes the goal may be to match a certain target value. Such user feedback guides the recommender in its subsequent search through the space of items, until an item is accepted. This dynamic can be thought of as a two-player game of imperfect information, in which the ultimate aim is to maximise the expected rating of the finally-recommended item and minimise the number of cycles required to get there. Only the user has direct access to their preference model; only the recommender system has direct access to the complete population of items. Critiquing-based approaches are

useful in domains where users are non-experts who can pass judgement on items when they see them, but are not capable of reliably specifying their requirements *a priori*, as required in constraint-based recommendation [14].

### Context-aware recommendation

Context-aware recommendation is somewhat akin to the content-based approach in that it assigns keywords, categories or properties, except that here those labels are applied to the user and the particularities of their present situation. For example, the system may categorise users according to demographic factors, their recent activity on the platform, or environmental data such as location, weather, and time-of-day [15]. Such information can be used to add constraint to a collaborative filtering algorithm, or form the basis of hand-crafted rules (e.g. *women in their 30s shop for clothes at weekends*). A disadvantage of naïve context-aware recommendation is that it may entrench unhelpful stereotypes, and ineffectively cater to individuals who do not conform to trends in the wider population [16]. Rather than enforcing hard thresholds and filters based on contextual factors, it is often more appropriate to apply a weighting operation, raising or lowering ratings incrementally.

### Hybrid recommendation

Hybrid recommendation is the catch-all term for approaches that combine several of the basic recommendation types in order to attain improved accuracy or generalisation performance [17]. Separate recommendation modules, which take different views on the available data, may be designed and assembled into a cascade pipeline or have their recommendations combined in a weighted manner. Alternatively, a fully-mixed approach may be used, in which the different techniques are blended to yield a single recommendation whose origins cannot be so easily disentangled [3]. With the proliferation of large, diverse datasets and powerful computing hardware, the majority of recommender systems deployed in industry harness some degree of hybridisation [18].

#### 2.0.3 Recommendation for Groups

The philosophical and practical problem of effective decision making within inhomogeneous groups has been a focus of scholarly inquiry for millennia, with ramifications throughout society. In the political and business spheres, many decision making protocols have been crafted, from a wide array of voting-based methods to the unanimity-focused debates of the European Council and the endless complexities of bureaucratic government. In computer science, a range of algorithms have been developed to support the decision making of groups. Of particular relevance to this project are decision contexts where the choice is between a large but finite number of options, over which each group

member has a set of preferences. This is the domain of group recommender systems, of which travel and tourism is one of the most natural applications.

At one level, group recommender systems have much in common with their single-user counterparts. The relationship between individual users and items, as quantified by implicit or explicit preferences, persists, as does the ultimate aim to maximise satisfaction or utility. Additional complexity arises from the fact that in groups, multiple sets of preferences exist, which are very likely to be in conflict [3]. In this context there is generally no single optimal utility function, and assumptions must be made about how best to reconcile any conflicts and give fair priority to all individuals.

At a high-level, group recommender systems follow two alternative strategies for arriving at a single decision. Firstly, the system may compute separate item rating scores for each individual in isolation, then combine the scores into a single set of values (*aggregation-of-scores*). Alternatively, preference and demographic information about all members may be aggregated into a *group profile* which is treated as a single user, and processed as normal to select a single outcome (*aggregation-of-preferences*) [3]. One or the other of these approaches may yield superior results, depending on the size and homogeneity of the group, and the nature of the available options. There are also many valid mathematical operations that could be used to aggregate the scores or preferences. These vary from a simple averaging procedure to more sophisticated mechanisms of voting or simulated bargaining. In the *least misery* method, the objective of aggregation is to yield a recommendation that maximises the smallest satisfaction value across the group members [19].

#### 2.0.4 Recommendation for Tourism

Tourism has long been a popular application domain for recommendation technology. A review article dating back to 2002 [20] presents two successful systems that had been deployed for real-world use: TripleHop’s TRIPMATCHER [21] and VacationCoach’s ME-PRINT. Both systems focus exclusively on aiding the selection of high-level destinations rather than specific POIs; the author claims that this level of recommendation affords the simplest direct comparison between options. Both systems also follow a broadly content-based approach, aiming to mimic the interaction with a human travel agent through a sequence of targeted questions to elicit users’ preferences. They also sort tourists into categories to further tailor recommendations, with ME-PRINT requiring the user to self-identify with one of several labels, and TRIPMATCHER applying collaborative filtering to users’ history of interactions on the site. The author warns that a recommender system relying entirely on collaborative filtering cannot work in the tourism domain, since the space of travel experiences is so vast that any user-item ratings matrix becomes too sparse to be usable. This view is supported by a more recent survey from 2014 [22], which

notes that the majority of successful recommenders employ hybrid mechanisms. It also heralds the widespread ownership of mobile devices as a major opportunity to provide time- and location-sensitive recommendations to tourists, greatly enhancing relevance. A trend towards agent-based recommendation architectures is also highlighted. Here users, travel providers and even candidate destinations are explicitly modelled as agents with goals and a degree of decision-making autonomy.

One such agent-based system, called TURIST@ [23], tackles the problem of within-city POI recommendation. The authors describe a complex architecture featuring one agent per activity type (e.g. sights, sports, theatre), one per user, and one central broker. The system employs a hybrid of content-based methods (comparing feature vectors for user profiles to those for candidate POIs) and collaborative filtering (clustering using the CLUSDM algorithm) to synthesise recommendations. User profiles are initialised via a questionnaire and demographic data to mitigate the cold-start problem. Another ambitious tourism recommender, called SAMAP [24], augments suggested itineraries with transport links and dining spots. The system comprises three agents, handling sequential stages of the recommendation pipeline. The first models user preferences, the second uses collaborative filtering to generate a list of suitable POIs, and the third assembles a subset of this list into an itinerary, compatible with opening times and transportation. The system uses logical induction to compare desired and actual attributes and return results that satisfy all constraints. In addition to the agent-based formulation, a staged item-selection pipeline is also a common feature of tourism recommenders. ABiPRS [25] seeks to rank a list of POIs by their expected relevance, but first applies a series of hard filters based on geographic proximity, time-of-day and overall score on review websites; this greatly reduces the computational intensiveness of the ranking operation. The tool also employs an ensemble of ranking models which, when evaluated using data from both *TripAdvisor* and *Yelp*, is found to improve robustness over any single method alone.

In each of the above cases, the objective is to recommend a full itinerary of POIs between fixed start and end points – this is commonly known as the *orienteering problem*. Other systems focus on recommending only the *next POI to visit*, given a tourist’s current location. Many successfully employ collaborative filtering for this task, drawing similarities between users based on their existing histories [26], potentially operating on the basis of POI categories as well as individual location [27] or incorporating temporal information to ensure that recommendations are appropriate for the time-of-day [28]. Others use more generic machine learning techniques such as Markov models, gradient-boosted regression trees and support vector machines [29]. These also may take temporal, and even seasonal, factors into account [30]. The majority of these systems output just a single recommendation or *top-k* items. While easy for the end user to digest, this limits the amount of information available to them and partially curtails their agency.

Over the past decade, the majority of innovative work around tourism recommenda-

tion has operated at the level of within-city POI recommendation, but several works have applied similar techniques in different contexts. For example, the PERSQ algorithm [31] uses Monte Carlo tree search to build amusement park ride itineraries, with the goal of minimising queuing time while maximising enjoyment. Other systems recommend high-level destinations (e.g. cities), for example by identifying scenery from a database that is similar to a user-provided photo or set of keywords [32], or by engaging the user in a comprehensive survey of landscape and cultural preferences [33]. In the latter work, recommendation results are presented not as a list or ranking, but as a cartogram with localised colouring that represents the suitability of each region. A third destination-level system [34] seeks to bypass extensive questioning by asking users merely to place themselves into one of several ‘personality categories’, which in turn can be matched up with stereotypical travel preferences. It is demonstrated that many of the key differences between individuals can be successfully captured by this approach.

Several previous works have investigated the problem of tourism recommendation for heterogeneous groups. Some follow traditional content-based approaches, matching POIs to explicitly-given individual preferences before computing group ratings via a weighted average [35], while others provide a platform for discussion and debate with the aim of converging towards agreement [36]. The debate may be mediated by a central ‘negotiation agent’ which iteratively proposes candidate destinations in response to expressed preferences [37] or critiques of previous proposals [38].

## 2.1 Flickr Data for Tourism Recommendation

The rise of social networks and GPS-enabled devices over the past ten years has provided a vast new data source of time- and location-tagged content which can be used to infer the movements of tourists around the world. This idea has been explored numerous times in the academic literature; in nearly all these cases the *Flickr* photo sharing platform has been used as the primary data source. Many millions of posts on this service are fully visible under a Creative Commons license, and come with a variety of surrounding metadata such as user tags and descriptions. 2015 saw the release of the *Flickr YFCC100M* dataset [39], which contains metadata for 100 million such posts, and has made ambitious research in this space significantly easier. The aforementioned work on the PERSQ algorithm [31] harnesses *YFCC100M*, from which photo locations and timestamps are extracted to infer the time real tourists spent in the vicinity of each theme park ride. Queuing times are estimated by subtracting ride durations found on *Wikipedia*. The dataset is also used to estimate ride popularity, under the assumption that users upload more photographs of places they enjoy more.

Elsewhere, the dataset has been used to inform recommendation of touristic routes between specified start and end points within the city of Beijing [40]. Here, the data

are preprocessed to create a ground-truth bank of itineraries based on the temporal order and location of uploaded photos. These are used for collaborative filtering along with the user’s own location history, to construct routes inspired by those of similar tourists. The system is evaluated against the dataset itself (cross-validation) via the metric of *coincidence rate* (how frequently predicted routes match the ground-truth data). It is found to outperform twelve classical methods on this metric, though given the model is developed specifically for this single-city data source, the risk of overfitting should be taken seriously. In [41], a travel histories dataset of eight cities is derived from *YFCC100M*, and results for four are presented. Again the objective is to recommend POI sequences that maximise enjoyment given fixed start and end points, but here visit and travel times are also considered. The task is cast as an integer programming problem, and cross-validation is used to evaluate performance via metrics such as F1-score. In a later work [42], the authors extend the model to the group recommendation problem: groups are assembled via clustering based on the cosine similarity of their preference vectors and a group profile is computed by a simple averaging operation. In these works, a number of naïve baselines are developed and used as points of comparison, such as ranking based on overall popularity or proximity to the last-visited POI.

A similar set of baselines are adopted in [43], where a system is developed to suggest travel routes within four Greek cities. Here, a large amount of attention is paid to the method of reconstructing POI visit histories from the Flickr dataset. Location-tagged photos are clustered into high-density ‘areas of interest’, a set of popular keywords are extracted from user-provided titles, descriptions and tags, and these keywords are related to profiles of nearby POIs taken from the *OpenStreetMap* mapping service. This approach allows large-scale assignment of POIs to photos with minimal manual input. In an even more ambitious work concerning nine popular cities [44], POI labelling is assisted by visual feature matching (via SIFT descriptors) across the uploaded images themselves, and textual tags are instead used to infer enjoyment via sentiment analysis. However, it should be acknowledged that none of the cited works include any analysis of the quality of their Flickr-derived datasets, and effectively assume their accuracy when using them to inform recommendation.

Each of the above *Flickr*-based systems has been developed to solve POI recommendation problems (either orienteering or single-next-POI) within a small number of cities. No work that could be found that harnesses the large bodies of location-tagged data to infer preferences and enable collaborative filtering for destination-level recommendation, or for POI recommendation within any more than ten target cities. In this respect, it appears that the full potential of this global data source is yet to be realised.

## 2.2 Summary of Implications

The range of possible approaches to recommendation problems is vast, and only becomes more diverse when recommending to diverse groups of users. It is difficult to determine *a priori* which approach will yield the strongest performance, so it is beneficial to consider a variety of options or build a hybrid system that utilises several.

In the domain of tourism, most novel developments in data-driven recommendation have been for within-city POI recommendation, usually framed in terms of the orienteering problem. City-level recommenders appear to demonstrate less innovation, with most still relying on explicitly-given preferences. There is little evidence of cross-pollination of ideas between the two domains, and I found no work that seeks to perform both city- and POI-level recommendation using the same underlying dataset.

In recent years, *Flickr* has become a very popular source of data for travel recommendation, but remarkably little space in publications has been dedicated to the exact methods used to infer tourists' visitation histories from their photo uploads, and there is no evidence of systematic evaluation of the accuracy of the derived datasets. In addition, the scope of these data-driven projects has been limited, with results reported for only a small number of cities (typically between 4 and 10). This begs the question as to whether the developed techniques may have been overfitted to their narrow applications. Cross-validation is a standard method for the offline evaluation of POI recommendation models, with a number of simple baselines (e.g. overall popularity, proximity) being common points of comparison.

# Chapter 3

## Overview of Recommendation Models

This chapter mathematically formalises the two problems that I seek to address in this project: city recommendation for groups of tourists and within-city POI recommendation for a single tourist. Rather than framing each problem as the presentation of a small number of ‘best’ options to the target tourist or group, I am interested in computing a complete ranking, from most- to least-recommended, of a finite set of options. After introducing each problem, I present various models for solving it, that are intended to be generic enough to be applied in a variety of environments. Subsequent chapters concern the application of these models to a specific dataset of real-world travel histories.

### 3.1 City Recommendation

The first problem addressed in this project is that of recommending new cities for groups of tourists to visit. Recommendations are based on the set of cities that each tourist has already visited, and their preferences for various categories of point-of-interest. Figure 3.1 presents the key conceptual elements of the problem.

#### 3.1.1 Notation

Throughout this section and the rest of the thesis, I employ the following notation:

- $\mathbb{T} = \{t_1, \dots, t_k\}$  is a set of  $k$  individual tourists.
- $\mathbb{C} = (c_1, \dots, c_l)$  is a tuple of  $l$  cities.  $\text{set}(\mathbb{C})$  returns the set of its elements.
- $\mathbb{X} = (x_1, \dots, x_m)$  is a tuple of  $m$  POI categories.  $\text{set}(\mathbb{X})$  returns the set of its elements.
- $\mathbb{V}^t = \{c_1, \dots, c_a\}$  and  $\mathbb{U}^t = \{c_1, \dots, c_b\}$  are the sets of tourist  $t$ ’s visited and unvisited cities respectively, such that  $\mathbb{V}^t \cup \mathbb{U}^t = \text{set}(\mathbb{C})$  and  $\mathbb{V}^t \cap \mathbb{U}^t = \emptyset$ .

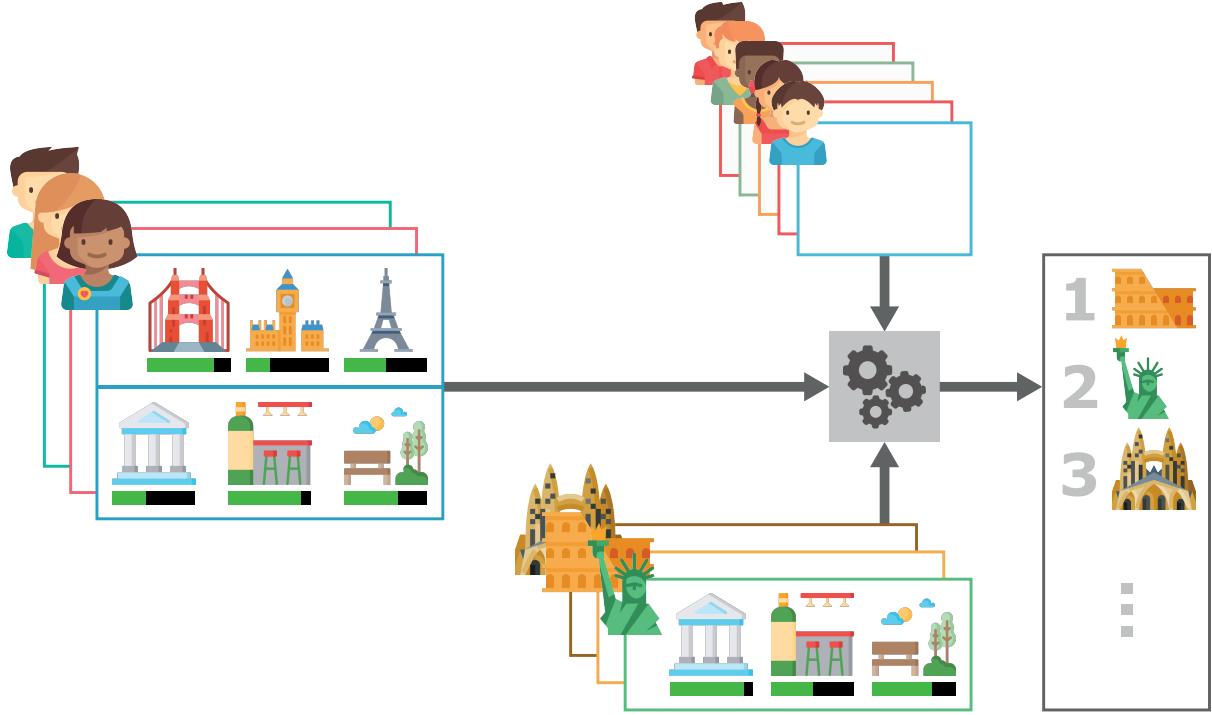


Figure 3.1: Diagram illustrating the key elements of the city recommendation problem. Each member of the tourist group [left] has a set of previously-visited cities, and a measure of enjoyment for each. They also have preferences over POI categories such as architectural landmarks, bars and parks. The same information is also available for a wider population of tourists [top]. The character of each city [bottom] can be described by the relative importance of each POI category within it. For example, a city with a high importance value for landmarks (e.g. New York) is one in which tourists are likely to visit specifically to experience this category of POI. In this chapter I propose three recommendation models, which each take a different perspective on the available data to score and rank all unvisited cities from most- to least-recommended [right].

- $\mathbf{e}^t = [\mathbf{e}_1^t, \dots, \mathbf{e}_l^t]$  is a vector of tourist  $t$ 's past enjoyment of each city, ordered to correspond with  $\mathbb{C}$ .  $\forall i \in \{1..l\} \mathbf{e}_i^t \in \mathbb{R}_{\geq 0}$  and  $\sum_{i=1}^l \mathbf{e}_i^t = 1$ . Note that unvisited cities are included, but with enjoyment values of 0.
- $\mathbf{p}^t = [\mathbf{p}_1^t, \dots, \mathbf{p}_m^t]$  is a vector of tourist  $t$ 's preference for each POI category, ordered to correspond with  $\mathbb{X}$ .  $\forall i \in \{1..m\} \mathbf{p}_i^t \in \mathbb{R}_{\geq 0}$  and  $\sum_{i=1}^m \mathbf{p}_i^t = 1$ .
- $\mathbf{i}^c = [\mathbf{i}_1^c, \dots, \mathbf{i}_m^c]$  is a vector representing the relative importance of POI categories in city  $c$ , ordered to correspond with  $\mathbb{X}$ .  $\forall i \in \{1..m\} \mathbf{i}_i^c \in \mathbb{R}_{\geq 0}$  and  $\sum_{i=1}^m \mathbf{i}_i^c = 1$ .
- $\text{enj}^t(c) \rightarrow \mathbb{R}_{\geq 0}$  is a latent function that returns tourist  $t$ 's *potential enjoyment* for each  $c \in \mathbb{U}^t$ , were they to visit that city.

### 3.1.2 Problem Definition

In the single-tourist case, the objective of city recommendation is to produce a ranking of cities  $\mathbb{R}^t = (c_1, \dots, c_b)$ , containing all the elements in  $\mathbb{U}^t$ . An optimal ranking,  $\hat{\mathbb{R}}^t$ , is ordered such that  $\forall i \in \{1 \dots b - 1\}$ ,  $\text{enj}^t(\hat{\mathbb{R}}_i^t) \geq \text{enj}^t(\hat{\mathbb{R}}_{i+1}^t)$ . However, since the function  $\text{enj}^t$  is unobservable by the recommender system, optimality is unlikely to be practically attainable. A more relaxed performance metric is provided by the *normalised discounted cumulative gain* function nDCG [45], which operates on the premise that a highly relevant item (in this case, a city with high potential enjoyment) that appears low in the ranking should be penalised in logarithmic proportion to its position. In this particular context, nDCG is defined as

$$\text{nDCG}(\mathbb{R}^t) = \frac{\text{DCG}(\mathbb{R}^t)}{\text{DCG}(\hat{\mathbb{R}}^t)} \quad (3.1)$$

where DCG is the *discounted cumulative gain* function

$$\text{DCG}(\mathbb{R}^t) = \sum_{i=1}^b \frac{2^{\text{enj}^t(\mathbb{R}_i^t)} - 1}{\log_2(i + 1)} \quad (3.2)$$

This definition assumes that while  $\text{enj}^t$  is hidden from the recommender system, it is accessible during evaluation.  $\text{nDCG}(\mathbb{R}^t)$  always lies in  $[0, 1]$ , equalling 1 if and only if the city ranking is optimal,  $\mathbb{R}^t = \hat{\mathbb{R}}^t$ .

The following subsections present three generic recommendation models for producing a city ranking  $\mathbb{R}^t$  for a single tourist  $t$ , given  $\mathbb{V}^t$ ,  $\mathbf{e}^t$ ,  $\mathbf{p}^t$  and  $\mathbf{i}^c$  for populations of tourists  $\mathbb{T}$ , cities  $\mathbb{C}$  and POI categories  $\mathbb{X}$ . These models are then extended to work with multi-tourist groups. Later chapters describe the implementation of the models using real-world data.

### 3.1.3 Ranking by Tourist-Tourist Similarity

The first model produces city rankings through conventional collaborative filtering across the population of tourists. I measure the similarity  $\text{sim}(t, t')$  of two tourists  $t$  and  $t'$  in terms of their city enjoyment distributions  $\mathbf{e}^t$  and  $\mathbf{e}^{t'}$ , using the rationale that tourists who have visited and enjoyed several similar cities in the past are likely to continue to have similar tastes in the future.

The pairwise similarity between two continuous-valued probability distributions can be quantified in numerous ways, including cosine similarity, Pearson correlation and Manhattan distance [46]. For this application, as well as several others throughout the models in this chapter, I employ the non-parametric *Jensen-Shannon divergence* (JSD). It is based on the Kullback-Leibler divergence, with the additional advantages of being symmetric and always having a finite value. Its square root – the *Jensen-Shannon distance* – satisfies all the requirements of a distance measure [47]. The JSD between two

distributions  $\mathbf{x}$  and  $\mathbf{y}$  is defined as

$$\text{JSD}(\mathbf{x}||\mathbf{y}) = \frac{1}{2}D(\mathbf{x}||\mathbf{z}) + \frac{1}{2}D(\mathbf{y}||\mathbf{z}) \quad (3.3)$$

where  $\mathbf{z} = \frac{1}{2}(\mathbf{x} + \mathbf{y})$  and  $D$  is the Kullback-Leibler divergence:

$$D(\mathbf{x}||\mathbf{y}) = \int_{\infty}^{\infty} \mathbf{x}(a) \log \left( \frac{\mathbf{x}(a)}{\mathbf{y}(a)} \right) da \quad (3.4)$$

The first step for producing a recommendation ranking for a single tourist  $t$  is to compute the similarity value with each other tourist  $t'$ , which I define as 1 minus the Jensen-Shannon distance between their city enjoyment distributions:

$$\text{sim}(t, t') = 1 - \sqrt{\text{JSD}(\mathbf{e}^t || \mathbf{e}^{t'})} \quad (3.5)$$

A *neighbourhood*  $\mathbb{N}^t$  of the  $n$  highest-similarity tourists is assembled using this metric.  $\forall i : \mathbb{C}_i \in \mathbb{U}^t$ , the recommendation score  $S_{TT}^t(\mathbb{C}_i)$  is defined as

$$S_{TT}^t(\mathbb{C}_i) = \frac{1}{n} \sum_{t' \in \mathbb{N}^t} \mathbf{e}_i^t \cdot \text{sim}(t, t') \quad (3.6)$$

A city ranking  $\mathbb{R}_{TT}^t$  is constructed by ordering the cities in  $\mathbb{U}^t$  by their  $S_{TT}^t$  scores.

### 3.1.4 Ranking by City-City Similarity

The second model produces city rankings by drawing similarities between the cities in  $\mathbb{V}^t$  and those in  $\mathbb{U}^t$ , and works on the assumption that tourists tend to visit cities with features in common. I measure the similarity  $\text{sim}(c, c')$  of two cities  $c$  and  $c'$  in terms of their POI category importance distributions  $\mathbf{i}^c$  and  $\mathbf{i}^{c'}$ . The result is that, for example, two cities with many popular parks and restaurants but few museums would be measured as highly similar.

Again, I define city similarity in terms of Jensen-Shannon distance. Similarity values are computed between each unvisited city  $c \in \mathbb{U}^t$  and each visited city  $c' \in \mathbb{V}^t$ :

$$\text{sim}(c, c') = 1 - \sqrt{\text{JSD}(\mathbf{i}^c || \mathbf{i}^{c'})} \quad (3.7)$$

$\forall c \in \mathbb{U}^t$ , the recommendation score  $S_{CC}^t(c)$  is defined as

$$S_{CC}^t(c) = \frac{1}{a} \sum_{i : \mathbb{C}_i \in \mathbb{V}^t} \mathbf{e}_i^t \cdot \text{sim}(c, \mathbb{C}_i) \quad (3.8)$$

where  $a$  is the cardinality of  $\mathbb{V}^t$ . A city ranking  $\mathbb{R}_{CC}^t$  is constructed by ordering the cities in  $\mathbb{U}^t$  by their  $S_{CC}^t$  scores.

### 3.1.5 Ranking by Tourist-City Similarity

The third and final model produces recommendations by matching the tourist's POI category preferences with the POI category importance distribution for each city in  $\mathbb{U}^t$ . This model is closest to the traditional notion of a content-based recommender system, and operates on the assumption that, for example, a tourist with a known preference for visiting art galleries should be recommended cities where this kind of attraction is popular.

The Jensen-Shannon distance is again used to define the similarity metric. Here, similarity values are computed between the tourist's POI category preferences  $\mathbf{p}^t$  and the POI category importance distribution  $\mathbf{i}^c$  for each unvisited city  $c \in \mathbb{U}^t$ :

$$\text{sim}(t, c) = 1 - \sqrt{\text{JSD}(\mathbf{p}^t || \mathbf{i}^c)} \quad (3.9)$$

In this model, no further computation is needed.  $\forall c \in \mathbb{U}^t$ , the recommendation score from tourist-city similarity  $S_{TC}^t(c)$  can be simply defined as

$$S_{TC}^t(c) = \text{sim}(t, c) \quad (3.10)$$

A city ranking  $\mathbb{R}_{TC}^t$  is constructed by ordering the cities in  $\mathbb{U}^t$  by their  $S_{TC}^t$  scores.

### 3.1.6 Extension to Multi-Tourist Groups

In the case of producing city recommendations for a group, defined as a set of  $g$  tourists  $\mathbb{G}$  where  $1 < g \ll k$ , the problem definition must be adapted slightly. Here, a city ranking  $\mathbb{R}^{\mathbb{G}}$  should contain all the elements in  $\mathbb{U}^{\mathbb{G}}$ , where

$$\mathbb{U}^{\mathbb{G}} = \bigcap_{t \in \mathbb{G}} \mathbb{U}^t \quad (3.11)$$

The quality of a ranking can be quantified as the average nDCG value across the tourists in the group.

I propose two approaches to extending the three recommender models to multi-tourist groups, one based on *aggregation-of-scores* (*AoS*) and the other based on *aggregation-of-preferences* (*AoP*).

#### AoS Approach

In this aggregation approach, an independent recommendation score  $S_{XX}^t(c)$  (where  $XX$  stands for any of the three model types: *TT*, *CC* or *TC*) is generated for each tourist  $t \in \mathbb{G}$  and each city  $c \in \mathbb{U}^{\mathbb{G}}$ , before being combined. There are several viable combination methods, of which I explore two in this project: computing the *sum* of the scores and

taking the *maximum* value from across the group. In these two cases respectively, the group recommendation score  $S_X^G(c)$  is defined as

$$S_{XX}^G(c) = \sum_{t \in \mathbb{G}} S_{XX}^t(c) \quad \text{or} \quad S_{XX}^G(c) = \max_{t \in \mathbb{G}} (S_{XX}^t(c)) \quad (3.12)$$

As in the single-tourist models, the city ranking  $\mathbb{R}_{XX}^G$  is constructed by ordering the cities in  $\mathbb{U}^G$  by their  $S_{XX}^G$  scores.

### AoP Approach

In this aggregation approach, a single ranking is generated using a *group profile*, consisting of an aggregated visited cities set  $\mathbb{V}^G$ , city enjoyment distribution  $\mathbf{e}^G$  and POI category preference distribution  $\mathbf{p}^G$ .  $\mathbb{V}^G$  is simply the union of the visited cities for the tourists in the group:

$$\mathbb{V}^G = \bigcup_{t \in \mathbb{G}} \mathbb{V}^t \quad (3.13)$$

while for the two distribution vectors, there are again several ways in which they could reasonably be combined. Once again opting to consider sum- and maximum-based aggregation, I define  $\mathbf{e}^G$  as

$$\mathbf{e}^G = \frac{\sum_{t \in \mathbb{G}} \mathbf{e}^t}{\sum_{i=1}^l \sum_{t \in \mathbb{G}} \mathbf{e}_i^t} \quad \text{or} \quad \mathbf{e}^G = \frac{\text{ewmax}_{t \in \mathbb{G}} (\mathbf{e}^t)}{\sum_{i=1}^l \max_{t \in \mathbb{G}} (\mathbf{e}_i^t)} \quad (3.14)$$

where `ewmax` returns the elementwise maximum of a set of vectors. It is important that the aggregated vector be renormalised to sum to 1, so that it remains a probability distribution. Similarly, I define  $\mathbf{p}^G$  as

$$\mathbf{p}^G = \frac{\sum_{t \in \mathbb{G}} \mathbf{p}^t}{\sum_{i=1}^m \sum_{t \in \mathbb{G}} \mathbf{p}_i^t} \quad \text{or} \quad \mathbf{p}^G = \frac{\text{ewmax}_{t \in \mathbb{G}} (\mathbf{p}^t)}{\sum_{i=1}^m \max_{t \in \mathbb{G}} (\mathbf{p}_i^t)} \quad (3.15)$$

Once the elements of the group profile have been computed, they can be used directly as inputs for the single-tourist recommender models, meaning the group profile is effectively treated as an individual tourist.

## 3.2 Point-of-interest Recommendation

The second problem addressed in this project is that of recommending points-of-interest to visit during an ongoing trip to a specific city. In this problem, only a single target tourist is considered. In addition to the tourist's preferences, the suitability of recommendations is highly sensitive to contextual factors (the tourist's current location, as well as the time

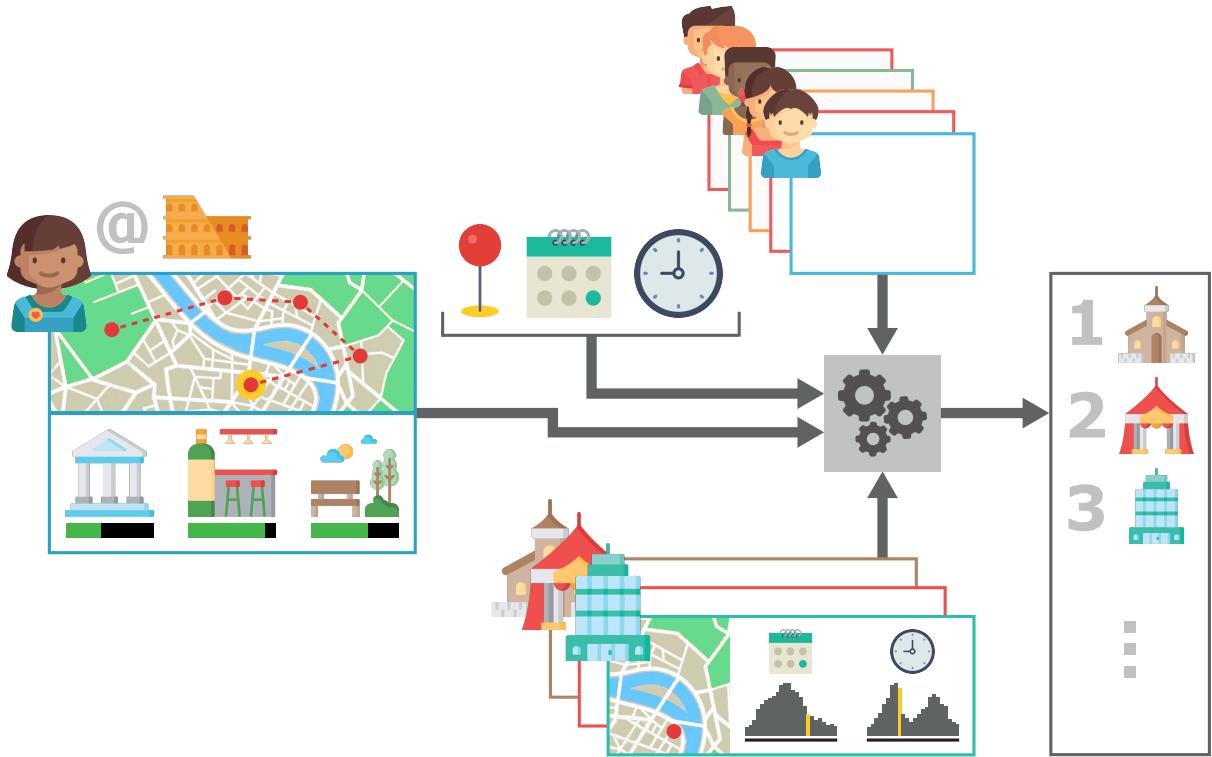


Figure 3.2: Diagram illustrating the key elements of the POI recommendation problem. The target tourist [left] is assumed to have arrived in the city and visited at least one POI already. Their history of visits is available, complete with time, date and location information, alongside their POI category preference values as in the city recommendation problem. The same information is also available for a wider population of tourists [top-right]. Each POI [bottom] is a member of a category, and can be described by its location, as well as distributions describing how its popularity varies as a function of time-of-day and date. This can be related to the current time, date and location of the tourist [top-left]. In this chapter I propose a hybrid recommendation model that incorporates a variety of contextual and personalised factors to score and rank all POIs in the city from most- to least-recommended [right].

and date) and the set of POIs that have already been visited. Figure 3.2 presents the key conceptual elements of the problem.

### 3.2.1 Notation

In addition to all notation introduced in the previous section, the following is required for discussion of the point-of-interest recommendation problem:

- $\mathbb{P}^c = \{p_1, \dots, p_{o^c}\}$  is a set of  $o^c$  POIs in city  $c$ .
- $\text{cat} : \mathbb{P}^c \rightarrow \text{set}(\mathbb{X})$  is a function that maps a POI  $p \in \mathbb{P}^c$  to a category  $x \in \text{set}(\mathbb{X})$ .
- $\text{loc} : \mathbb{P}^c \rightarrow \mathbb{R}^2$  is a function that maps a POI to a [latitude, longitude] vector.

- $\mathbb{H}^{t,c} = (v_1, \dots, v_{q^{t,c}})$  is a tuple representing the history of  $q^{t,c}$  POI visits already made by tourist  $t$  in city  $c$ . It is ordered chronologically, with the first element representing the earliest visit. If  $t$  has never visited  $c$ ,  $\mathbb{H}^{t,c} = ()$ , the empty tuple.
- $\text{poi} : \mathbb{H}^{t,c} \rightarrow \mathbb{P}^c$  is a function that maps a visit  $v \in \mathbb{H}^{t,c}$  to a POI  $p \in \mathbb{P}^c$ .
- $\text{startTime} : \mathbb{H}^{t,c} \rightarrow \text{timestamp}$  and  $\text{endTime} : \mathbb{H}^{t,c} \rightarrow \text{timestamp}$  are functions that map a visit  $v \in \mathbb{H}^{t,c}$  to timestamp objects representing the start and end of that visit.
- $\text{month} : \text{timestamp} \rightarrow \{1..12\}$  is a function that maps a timestamp to an integer. If the corresponding visit is situated in the Northern Hemisphere, i.e.  $\theta \geq 0$  where  $[\theta, \phi] = \text{loc}(\text{poi}(v))$ , this integer represents its calendar month of occurrence (Jan:1, Feb:2, ... Dec:12). However, if  $\theta < 0$ , the number is shifted by 6 to reflect the different pattern of seasons (Jan:7, Feb:8, ... Dec:6).
- $\text{hour} : \text{timestamp} \rightarrow \{0..23\}$  is a function that maps a timestamp to an integer representing the hour of the day. Times are rounded down, e.g. 15:53 → 15.
- $\text{enj}^{t,\text{now}} : p \rightarrow \mathbb{R}_{\geq 0}$  is a latent function that returns tourist  $t$ 's *potential enjoyment* for each  $p \in \mathbb{P}^c$ , conditioned on the timestamp  $\text{now}$  of the hypothetical visit.

### 3.2.2 Problem Definition

In the single-tourist case, the objective of POI recommendation within a given city  $c$ , and at a target timestamp  $\text{now}$ , is to produce a ranking of POIs  $\mathbb{R}^t = (p_1, \dots, p_j)$ , containing each element  $p \in \mathbb{P}^c$ . In this project, I focus specifically on the problem of recommending a POI to visit shortly after the *most recent visit* in the tourist's history, thereby mimicking the scenario of exploring a city in an unstructured manner and deciding where to go next on-the-fly. In including all elements of  $\mathbb{P}^c$  in the ranking, this problem implicitly permits the recommendation of POIs that the tourist has already visited. There are realistic scenarios in which this may be appropriate, such as visiting a landmark at both night and day, and eating more than once at a highly-enjoyed restaurant.

An optimal ranking,  $\hat{\mathbb{R}}^t$ , is ordered such that  $\forall i \in \{1 \dots j - 1\}$ ,  $\text{enj}^{t,\text{now}}(\hat{\mathbb{R}}_i^t) \geq \text{enj}^{t,\text{now}}(\hat{\mathbb{R}}_{i+1}^t)$ . Just as in the city recommendation problem, the function  $\text{enj}^{t,\text{now}}$  is unobservable by the recommender system, so optimality is practically unattainable. Again, the nDCG function provides a more relaxed performance metric, and can be applied in analogous fashion to that described in section 3.1.2<sup>1</sup>.

---

<sup>1</sup>When comparing recommendation performance between two cities  $c$  and  $c'$ , one must acknowledge that in general they will have different numbers of POIs, i.e.  $|\mathbb{P}^c| \neq |\mathbb{P}^{c'}|$ . During evaluation of the recommender model created in this project, I generally discuss ranks in terms of *proportions* of the number of available POIs.

The following subsections present a generic recommendation model for producing a POI ranking  $\mathbb{R}^t$  for a single tourist  $t$  in a city  $c$  at timestamp  $now$ , given  $\mathbb{P}^c$ ,  $\mathbb{H}^{t,c}$  and  $\mathbf{p}^t$  for populations of tourists  $\mathbb{T}$ , cities  $\mathbb{C}$  and POI categories  $\mathbb{X}$ . Later chapters describe the implementation of the model using real-world data.

### 3.2.3 Model Components

In contrast to the city recommender system, I integrate various approaches to scoring candidate POIs into a single hybrid model. Each approach yields a numerical feature, which is fed into an aggregation algorithm that outputs recommendation scores. For each tourist  $t$ , visit time  $now$  and POI  $p$ , I synthesise features representing the following:

- $p$ 's overall popularity;
- $t$ 's level of preference for  $\text{cat}(p)$ ;
- The proximity of  $p$  to  $t$ 's current location;
- The appropriateness of  $p$  given  $\text{month}(now)$  and  $\text{hour}(now)$ ;
- Population-wide correlations between the visitation of  $p$  and the POIs that  $t$  has already been to.

The character of each feature is modulated by hyperparameters. In this project, I consider three techniques for aggregating the features into final recommendation scores. The first is a simple sum of the (normalised) feature values, while the others are machine learning models: linear regression and a feedforward neural network.

### 3.2.4 Overall Popularity Feature

The first feature used in the model represents each POI's overall popularity amongst the population of tourists. For a POI  $p$ , I define the visitor count  $\text{vis}(p)$  as the number of *unique* tourists that have visited it (i.e. repeat visits by the same tourist are ignored):

$$\text{vis}(p) = |\{t : (\exists v \in \mathbb{H}^{t,c} : \text{poi}(v) = p)\}| \quad (3.16)$$

To create the popularity feature  $\text{fPop}(p)$ , I apply a function that compresses the value of  $\text{vis}(p)$  into the range  $[0, 1]$ . Preprocessing the feature in this way reduces the extreme differential between the most popular POIs and the least popular, which otherwise risks virtually eliminating the chance of the latter being highly ranked.  $\text{fPop}(p)$  is defined as

$$\text{fPop}(p) = 1 - 2^{\frac{-\text{vis}(p)}{\alpha}} \quad (3.17)$$

where  $\alpha > 0$  is a hyperparameter that dictates the steepness of the function. The equation is set up so that  $\alpha$  equals the number of unique visitors that makes  $\text{fPop}(p) = 0.5$ .

### 3.2.5 Preference-aware Feature

The second consideration that the model takes into account is the tourist's preferences over POI categories, as represented by the vector  $\mathbf{p}^t$  which was introduced in section 3.1.1. For a tourist  $t$  and POI  $p$ , I define the category preference weight  $\text{fCat}_{\mathbf{p}^t}(p)$  to equal the corresponding category preference value:

$$\text{fCat}_{\mathbf{p}^t}(p) = \mathbf{p}_i^t \quad (3.18)$$

where  $i$  is the index of  $\text{cat}(p)$  in the tuple of categories  $\mathbb{X}$ .

If, for example, the scoring model consisted only of the straight product of  $\text{fPop}$  and  $\text{fCat}_{\mathbf{p}^t}$ , each POI would be given a recommendation score proportional to its visitor count (with activation function applied), scaled by  $t$ 's preference for its category, and ranked accordingly. As an example: the Natural History Museum in London is extremely popular, but if the tourist had low preference for museums, this POI would appear lower in their personalised ranking than otherwise.

### 3.2.6 Context-aware Features

The model also accounts for the following factors that are agnostic to the preferences of the individual tourist, and are instead functions of their particular spatio-temporal context.

#### Proximity to Current Location

Tourists typically wish to minimise the time and effort spent moving between locations and experiences of interest, and as such will tend to prioritise easily-accessible POIs when choosing where to visit next. The model estimates these factors through the proxy of physical distance. Since in this project I focus on the problem of recommending POIs to go to soon after the tourist's most recent visit  $v = \mathbb{H}_{q^{t,c}}^{t,c}$ , I assume that the [latitude, longitude] of this visit  $[\theta_v, \phi_v] = \text{loc}(\text{poi}(v))$  is the tourist's current location. The Euclidean distance in metres from here to another POI  $p$ , whose location is  $[\theta_p, \phi_p] = \text{loc}(p)$ , can be approximated by

$$\text{dist}(v, p) = R \sqrt{(\theta_p - \theta_v)^2 + \cos^2 \theta (\phi_p - \phi_v)^2} \quad \text{where} \quad \theta = \frac{\theta_v + \theta_p}{2} \quad (3.19)$$

where  $R = 6.371 \times 10^6$ m is the average radius of the earth. This equation embodies a number of simplifying assumptions, not least that of the perfect sphericity of the planet, but given that all distances of interest are intra-city (thus at the scale of several thousand metres at most) the effect of its imprecision is negligibly small. I define the proximity

feature  $\text{fProx}_t(p)$  as

$$\text{fProx}(p) = 1 - 2^{\frac{\text{dist}(v,p)}{\beta}} \quad (3.20)$$

where  $\beta > 0$  is a hyperparameter that dictates the rate with which the feature value drops off with distance. The equation is set up so that  $\beta$  equals the distance to  $p$  in metres that makes  $\text{fProx}(p) = 0.5$ .

### Time-of-day Appropriateness

Another factor that mediates the suitability of a given POI recommendation is the time-of-day of the intended visit. Clearly, a visit to a park is more suited to 10am than 10pm, whereas quite the opposite is true for a bar or nightclub, and a restaurant is likely to be more relevant around common mealtimes than at other points in the day. Such knowledge could be hard-coded into an expert system module, but instead I choose to derive it statistically.

I initially quantify temporal trends on a category-wide basis, by assembling a histogram of visit counts starting at each hour of the day. This histogram considers visits in *all cities* in  $\mathbb{C}$ , not just the current location of the target tourist. For a category  $x$  and start hour  $h \in \{0..23\}$ , the histogram value  $\text{chHist}_h(x)$  is defined as

$$\text{chHist}_h(x) = \sum_{c \in \mathbb{C}} \sum_{t \in \mathbb{T}} \sum_{v \in \mathbb{H}^{t,c}} \delta_x(\text{cat}(\text{poi}(v))) \cdot \delta_h(\text{hour}(\text{startTime}(v))) \quad (3.21)$$

where  $\delta_x(y)$  is defined as

$$\delta_x y = \begin{cases} 1 & \text{if } y = x \\ 0 & \text{otherwise} \end{cases} \quad (3.22)$$

This value captures high-level information about the global visitation of each category, but misses any POI-specific deviations from the norm. Examples of this phenomenon might include a fountain that plays host to a late-night light show, or a pub with a beer garden that makes it unusually suited to daytime drinking. To capture this, a histogram can be assembled for each POI  $p$  in isolation. For each start hour  $h$ , the value  $\text{phHist}_h(p)$  is defined as

$$\text{phHist}_h(p) = \sum_{t \in \mathbb{T}} \sum_{v \in \mathbb{H}^{t,c}} \delta_p(\text{poi}(v)) \cdot \delta_h(\text{hour}(\text{startTime}(v))) \quad (3.23)$$

It may seem at first glance that equation 3.23 always yields more meaningful results than equation 3.21 for the purposes of recommendation, but this may not be the case if visits to  $p$  are sparse, thereby giving an unrepresentative sample. The following equation for the time-sensitive feature  $\text{fTime}_h(p)$  of a POI  $p$  balances the histogram for its category

against the sparser but more specific histogram for  $p$  itself, which gains weight as the number of visits to  $p$  increases:

$$\text{fTime}_h(p) = \left[ 2^\varepsilon \cdot \frac{\text{chHist}_h(\text{cat}(p))}{\frac{1}{24} \sum_{i=0}^{23} \text{chHist}_i(\text{cat}(p))} \right] + \left[ (1 - 2^\varepsilon) \cdot \frac{\text{phHist}_h(p)}{\frac{1}{24} \sum_{i=0}^{23} \text{phHist}_i(p)} \right] \quad (3.24)$$

where

$$\varepsilon = \frac{-\sum_{t \in \mathbb{T}} \sum_{i=1}^{q^{t,c}} \delta_p(\text{poi}(\mathbb{H}_i^{t,c}))}{\gamma}$$

The equation is set up so that the single hyperparameter  $\gamma > 0$  represents the number of visits that  $p$  must have for its specific histogram to be weighted *equally* to the category histogram.

### Time-of-year Appropriateness

In a similar vein, the date of the intended visit also influences how POIs should be scored. Places such as beaches, water parks and sports stadia all have significant seasonality, a fact which should be reflected in the final ranking. My approach to this factor mirrors almost exactly the time-of-day weighting described above: both category-wide and POI-specific histograms are assembled and balanced according to the POI's visit count. For a category  $x$  and month  $m \in \{1..12\}$ , the histogram value  $\text{cmHist}_m(x)$  is defined as<sup>2</sup>

$$\text{cmHist}_m(x) = \sum_{c \in \mathbb{C}} \sum_{t \in \mathbb{T}} \sum_{v \in \mathbb{H}^{t,c}} \delta_x(\text{cat}(\text{poi}(v))) \cdot \delta_m(\text{month}(\text{startTime}(v))) \quad (3.25)$$

The POI-specific histogram value  $\text{pmHist}_m(p)$  is defined as

$$\text{pmHist}_h(p) = \sum_{t \in \mathbb{T}} \sum_{v \in \mathbb{H}^{t,c}} \delta_p(\text{poi}(v)) \cdot \delta_m(\text{month}(\text{startTime}(v))) \quad (3.26)$$

The final date-sensitive feature  $\text{fDate}_m(p)$  is defined as

$$\text{fDate}_m(p) = \left[ 2^\varepsilon \cdot \frac{\text{cmHist}_m(\text{cat}(p))}{\frac{1}{12} \sum_{i=1}^{12} \text{cmHist}_i(\text{cat}(p))} \right] + \left[ (1 - 2^\varepsilon) \cdot \frac{\text{pmHist}_h(p)}{\frac{1}{12} \sum_{i=1}^{12} \text{pmHist}_i(p)} \right] \quad (3.27)$$

and the definition of  $\varepsilon$  is the same as in equation 3.24 (with a common  $\gamma$  hyperparameter).

---

<sup>2</sup>It is worth reiterating here that  $\text{month}(v)$  is defined such that the month numbers for Southern Hemisphere POIs are shifted by 6. This accounts for the flip in seasonality south of the equator.

### 3.2.7 History-aware Feature

The final factor that the model considers is the coincidence of POIs in tourists' histories. For example, it may be that tourists who visit a certain quiet park are also likely to visit a particular café, due not to physical proximity, but to its similarly peaceful ambience. Having measured all such coincidences, the model can combine them with the target tourist's visit history and boost the recommendation scores of POIs which commonly appear alongside those already visited. In building this part of the model, I make the assumption that visits that are closer in time should contribute more to the measure of coincidence. A natural extension of this means that more recent visits in the target tourist's history also contribute more to the final recommendation.

For two POIs  $p$  and  $p'$ , the *coincidence score*  $\text{coinc}(p, p')$  is defined as

$$\text{coinc}(p, p') = \frac{1}{\text{vis}(p')} \sum_{t : (\exists v \in \mathbb{H}^{t,c} : \text{poi}(v) = p) \wedge (\exists v \in \mathbb{H}^{t,c} : \text{poi}(v) = p')} \zeta + (1 - \zeta) \cdot 2^{\frac{-\Delta_t(v, v')}{\kappa}} \quad (3.28)$$

where  $\Delta_t(v, v')$  is the duration in hours by which the visit to  $p$  (denoted  $v$ ) and the visit to  $p'$  (denoted  $v'$ ) are separated in  $\mathbb{H}^{t,c}$ . This is either  $\text{startTime}(v') - \text{endTime}(v)$  or  $\text{startTime}(v) - \text{endTime}(v')$  depending on which was visited first. If one or more of  $p$  and  $p'$  appears multiple times in  $\mathbb{H}^{t,c}$ , the single shortest separation time is used.

The expression inside the summation in equation 3.28 applies a weight to each coincidence that decays exponentially with  $\Delta_t(p, p')$ . The two hyperparameters  $\zeta \in [0, 1]$  and  $\kappa > 0$  dictate the asymptotic value and the rate of decay respectively. Setting  $\zeta = 1$  effectively ignores the time difference, regardless how long, whereas setting  $\zeta = 0$  and  $\kappa$  to a small value means that only very temporally proximal visits contribute to the coincidence score. The  $\text{coinc}$  function is *not* symmetric ( $\text{coinc}(p, p') \neq \text{coinc}(p', p)$ ) because the equation is normalised by the number of visitors to  $p'$  as defined in equation 3.16.

Let  $\mathbb{A}^{t,c}$  be the set of POIs that tourist  $t$  has visited in city  $c$ :  $\mathbb{A}^{t,c} = \{\text{poi}(v) : v \in \mathbb{H}^{t,c}\}$ . The history-sensitive feature  $\text{fHist}_{\mathbb{H}^{t,c}}(p)$  for a POI  $p$  is defined as a function of its coincidence score with respect to each POI  $p' \in \mathbb{A}^{t,c}$ , scaled by the time elapsed between  $t$ 's most recent visit to  $p'$  (denoted  $v'$ ) and *now*:

$$\text{fHist}_{\mathbb{H}^{t,c}}(p) = \sum_{p' \in \mathbb{A}^{t,c}} \text{coinc}(p, p') \cdot \left[ \zeta + (1 - \zeta) \cdot 2^{\frac{-\Delta_t(v', \text{now})}{\kappa}} \right] \quad (3.29)$$

where the hyperparameters  $\zeta$  and  $\kappa$  are shared with equation 3.28. The summation operation means that tourists with more unique POIs in their visit histories generally produce higher history-sensitive feature values. This is appropriate since longer histories provide more information about a tourist's habits and preferences.

### 3.2.8 Scoring Methods

Given the six features introduced above, a single recommendation score must be computed for each POI. In this project, I compare three alternative methods for performing the features-to-score mapping operation. Prior to passing the features into each, I combine them into a *feature vector*  $\mathbf{v}^{t,p}$ , where

$$\mathbf{v}^{t,p} = [\text{fPop}(p), \text{fCat}_{\mathbf{p}^t}(p), \text{fProx}_{\mathbb{H}^{t,c}}(p), \text{fTime}_{\text{hour}(now)}(p), \text{fDate}_{\text{month}(now)}(p), \text{fHist}_{\mathbb{H}^{t,c}}(p)] \quad (3.30)$$

Rather than using the raw feature values, I perform a *z-normalisation* operation to bring them into a common scale. Given a large set of recommendation scenarios  $\mathbb{S}$ , each concerning a tourist  $t$  in city  $c$  at time  $now$ , and with history  $\mathbb{H}^{t,c}$  and preferences  $\mathbf{p}^t$ , the global mean  $\mu_i$  and standard deviation  $\sigma_i$  of element number  $i$  of the feature vector can be calculated as

$$\mu_i = \frac{1}{|\mathbb{S}|} \sum_{(t,c,now) \in \mathbb{S}} \sum_{p \in \mathbb{P}^c} \mathbf{v}_i^{t,p} \quad \text{and} \quad \sigma_i = \sqrt{\frac{1}{|\mathbb{S}|} \sum_{(t,c,now) \in \mathbb{S}} \sum_{p \in \mathbb{P}^c} (\mathbf{v}_i^{t,p} - \mu_i)^2} \quad (3.31)$$

respectively. Each element of each POI's feature vector, in each scenario in  $\mathbb{S}$ , can then z-normalised to yield a new vector  $\bar{\mathbf{v}}^{t,p}$  as follows:

$$\bar{\mathbf{v}}_i^{t,p} = \frac{\mathbf{v}_i^{t,p} - \mu_i}{\sigma_i} \quad (3.32)$$

The three scoring methods explored in this project are

- *Sum*: Simply summing all the elements of the z-normalised feature vector  $\bar{\mathbf{v}}^{t,p}$  to yield the score  $S_{Sum}^t(p)$ , i.e.

$$S_{Sum}^t(p) = \sum_{i=1}^6 \bar{\mathbf{v}}_i^{t,p} \quad (3.33)$$

- *Lin*: Computing a weighted linear sum of the elements of  $\bar{\mathbf{v}}^{t,p}$  to yield the score  $S_{Lin}^t(p)$ , i.e.

$$S_{Lin}^t(p) = \sum_{i=1}^6 \mathbf{w}_i \cdot \bar{\mathbf{v}}_i^{t,p} \quad (3.34)$$

where  $\mathbf{w}$  is a vector of *weights*, which must be optimised to minimise a particular error measure. My choice of error measure and training procedure for this project is described in section 6.2.

- *NN*: Feeding  $\bar{\mathbf{v}}^{t,p}$  into the input layer of a feedforward neural network with one or more hidden layers and a single output neuron whose activation is taken as the

score  $S_{NN}^t(p)$ . In this project, I consider a variety of network architectures with one, two or three hidden layers and use logistic activation functions. The neural network training procedure is also described in section 6.2.

Regardless of which scoring technique is used, the POI ranking  $\mathbb{R}^t$  is constructed by ordering the POIs in  $\mathbb{P}^c$  by their  $S^t(p)$  scores.

### 3.2.9 Extension to Multi-Tourist Groups

The POI recommendation model outlined in this section is designed to work with a single target tourist rather than a group. The model could be extended to work with groups in a roughly similar manner to the city recommender, through either *AoS-* or *AoP-based* methods. However, project time constraints, and foreseeable difficulties with adapting my chosen evaluation methods to the group recommendation setting, mean that I have left this extension as future work.

# Chapter 4

## Creation of Travel Histories Dataset

In this project, I implement the recommendation models outlined in the previous chapter to work specifically with a novel dataset of real-world travel histories. In this respect, the real tourists whose activity is documented in the dataset serve as the user base for the recommender systems. This chapter describes the method by which I synthesised the dataset from published social media data, augmented by information from an open-source mapping service.

### 4.1 *YFCC100M* and *OpenStreetMap*

*YFCC100M* [39] is the largest publicly-available media collection in the world, consisting of detailed metadata for 100 million posts (99.2 still images and 0.8 million videos) uploaded to the *Yahoo!*-owned *Flickr* photo sharing platform between 2004 and 2014 [39]. Types of available metadata include unique photo and user IDs; the time, date, longitude and latitude of capture; and a user-assigned title, description and list of tags. All of these are actively utilised in this project. Alongside the core dataset, several expansion packs are available, including one called *Places* which labels many of the photos with unique WOEID (Where On Earth IDentifier) tags<sup>1</sup>. These tags use *Yahoo!*'s internal database of geolocation data to provide place name estimates with a hierarchy of precision, even down to individual POIs in some cases. For example, the tag

```
Taka Taka:POI, Bristol:Town, Avon:County  
United Kingdom:Country, Europe/London:Timezone
```

refers to a particularly delectable late night sandwich shop in the city of Bristol.

*OpenStreetMap* (OSM) [48] is a crowdsourced world map, built by millions of volunteers who collect data on the locations of real-world entities (such as buildings, roads, businesses, public amenities and natural structures), often through manual survey. The

---

<sup>1</sup>The WOEID protocol now appears to be defunct and is no longer used by *Yahoo!*, but the tags nonetheless contain the information required for my purposes.

map database can be freely edited in the same manner as *Wikipedia* and, like the popular online encyclopaedia, compares well in terms of accuracy with proprietary alternatives such as *Google Maps*. As an open-source project, OSM has a vibrant community of developers and many APIs for querying various aspects of the database. As outlined below, I used one such API to obtain detailed records of POIs in cities around the world.

## 4.2 Implementation Notes

I implemented all processing stages described in this chapter using the Python programming language. Many of the more complex data wrangling operations were completed with the help of the `pandas` library [49], which provides a suite of fast and flexible tools for manipulating large tables. Since the volume of data in *YFCC100M* is large, the creation of the travel histories dataset was the most computationally-intensive part of the project. For this reason, I deployed the dataset and processing scripts on a high-performance Linux virtual machine instance (`n1-highcpu-4` machine type, 4 vCPUs, 3.6 GB memory) on the Google *Cloud Compute Engine*.

## 4.3 Data Preprocessing

After transferring the core *YFCC100M* dataset and Places expansion pack (two Bzip2-compressed files of size 14.6GB and 1.8GB respectively) from a dedicated Amazon S3 data bucket to the virtual machine instance, I used the `bz2` Python module to decompress the dataset files and read them in streaming format, one photo at a time<sup>2</sup>. To avoid needing to hold the entire dataset in memory at once, the script split it into 100 equal batches. The objective of this initial pass through the dataset was to divide it according to the city where each photo was taken, via the WOEID information in the Places expansion pack. Any photo that contained a WOEID tag with a precision level of `Town` or higher was appended to a data structure for the corresponding town (referred to as *city* throughout this thesis). Data for the top 200 most common cities were written out in `.csv` format for later use in this project. Each line in the 200 `.csv` files contained:

- The line number of the photo in the *YFCC100M* dataset;
- The unique identification number of the photo on the *Flickr* platform, allowing it to be viewed in a web browser if prefixed with `www.flickr.com/photo.gne?id=`;

---

<sup>2</sup>Both the core dataset and Places expansion pack are randomised with respect to date, location and originating user, but the randomised order is common between them. This makes relation of the two datasets trivial since, for example, line number 123 of the core dataset refers to the same photo as line 123 of the Places dataset.

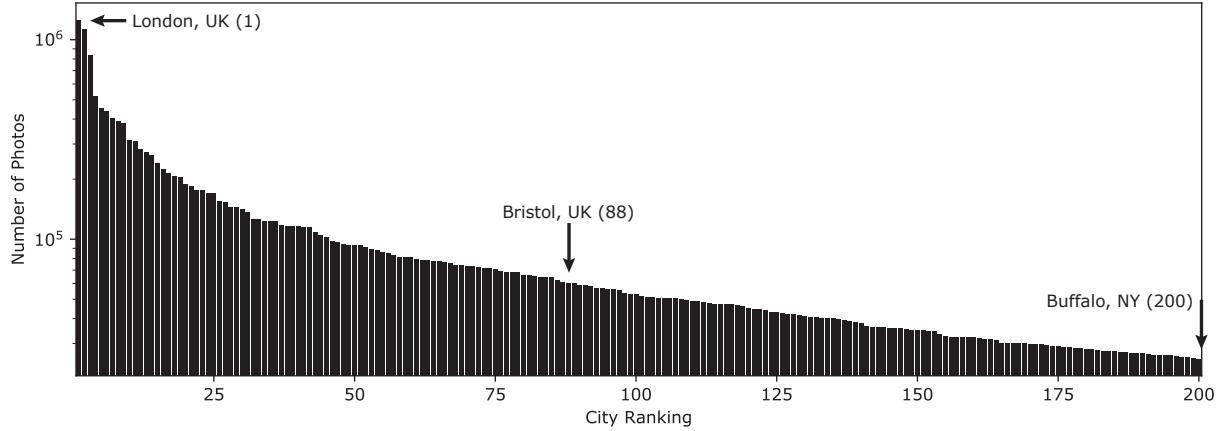


Figure 4.1: Distribution of photos taken in the 200 most popular cities in the *YFCC100M* dataset. Annotations indicate rankings of several of these cities.

- The unique identification string of the originating user, allowing their profile to be viewed in a web browser if prefixed with `www.flickr.com/photos/`;
- The time and date of capture in `dd/mm/yyyy HH:MM:SS` format;
- The longitude and latitude of capture, alongside a precision measure ranging from 1 (world-level) to 16 (street-level);
- If available, the `POI` section of the WOEID tag, which would later be used to assist the labelling of POI visits;
- The user-assigned title, description and list of tags for the photo;

Out of the full dataset of 100 million photos, 19.5 million were identified as having been taken in the top 200 cities. Fig. 4.1 shows the distribution of photo counts.

## 4.4 Visit Creation and Labelling

The next step was to use the information stored in the `.csv` files to accurately estimate the POI at which each photo was taken. Specifically, I used the longitude and latitude coordinates, the user-provided title, description and tags, and (initially) the name given in the `POI` section of the WOEID tag, if this was available.

### 4.4.1 Grouping Photos into Visits

From initial exploration of the dataset, including viewing the original images, my first observation was that many users had uploaded multiple consecutive photos at the same location, which should be treated as a single visit. I addressed this situation by processing

Table 4.1: **Top:** Relevant data for three photos taken by a user within a 40 minute period in the city of Bristol. **Bottom:** Single visit constructed from the three photos.

Time	Long	Lat	WOEID POI	Title	Description	Tags
01/07/19 12:00	-2.606800	51.454078	Brandon Hill Park	Tower	It's quite tall	tower, tall, bristol
01/07/19 12:15	-2.606887	51.454085		Plaque	Says some- thing about John Cabot	plaque, metal, folly
01/07/19 12:40	-2.606729	51.453963		Bristol View		brandon, hill, view

Start	End	Long	Lat	#	Words
01/07/19 12:00	01/07/19 12:40	-2.606805	51.454375	3	{brandon, hill, park, tower, it's, quite, tall, bristol, plaque, says, something, about, john, cabot, metal, folly, view}

the photos for each user independently: ordering by capture date, then iterating through from earliest to latest. The very first photo initiated the first visit. If any subsequent photo met both of the following conditions:

- Capture time within 1 hour of the *last* photo in the current visit;
- Coordinates within 10 metres of the *first* photo in the current visit, as computed using the Euclidean distance equation presented in equation 3.19;

it was appended to the current visit. Otherwise, a new visit was initiated. For each visit  $v$ , the number of constituent photos was stored, alongside the first and last capture time and mean coordinates. Also stored was a set of *visit words*  $W_v$ . I assembled  $W_v$  by concatenating the lists of words in all photo titles, descriptions and user tags, and, if available, the POI name from the WOEID tag. Duplicate words were discarded.

Table 4.1 illustrates the result of grouping three photos into a single visit.

#### 4.4.2 Obtaining POIs from *OpenStreetMap*

I used the `Overpass` API for OSM [50] to download a list of POIs within each city. In the OSM database, each entity is assigned a characteristic pair of coordinates, as well as a number of additional properties which define whether it represents an amenity (e.g. shop,

restaurant, cinema), building, road, natural space, landmark, or one of many other kinds of real-world object. Using the API’s query protocol, I retrieved only the entities that would be relevant to tourists, namely those with one or more of the following properties:

- **amenity** (excluding supermarkets, community centres and several others);
- **natural** (excluding individual trees or unnamed bodies of water);
- **building** (excluding residential buildings, schools, car parks and several others);
- **tourism** (excluding hotels and tourist information points);
- **historic**.

I arrived at this combination of properties after a process of experimentation, and found that it successfully retrieved the vast majority of entities that could reasonably be called POIs, with minimal extraneous additions. These properties also allowed each downloaded POI to be automatically assigned to a *category*, which would be crucial information for several subsystems of the recommendation models.

Among the other properties provided for each OSM entity are a name (often in several languages) and more specialist details such as the cuisine for a restaurant or architect for a building. Similar to the construction of visit words from user-provided titles and descriptions, I used such fields to assemble a set of *POI words*  $W^p$  for each POI  $p$ . As an example, Cabot Tower in the city of Bristol yielded the set  $W^p = \{\text{cabot, tower, william, venn, gough, folly, attraction}\}$ . The sets of visit words and POI words were the essential information for assigning POIs to user visits, a process I refer to as *labelling*.

#### 4.4.3 Labelling Visits with POIs

For a given visit  $v$ , labelling proceeded as follows:

1. Assemble a shortlist  $P^v$  of all the POIs within a 250 metre radius of its coordinates (the mean coordinates of the constituent photos).
2. For each POI  $p \in P^v$ , compute the intersection of  $W^v$  and  $W^p$ . Denote the resultant (in most cases, empty) set the *evidence*  $E^{v,p}$ .
3. Compute a match score  $S^{v,p}$  as a function of both the evidence and the distance between the coordinates of the visit and the POI:

$$S^{v,p} = \begin{cases} \frac{|E^{v,p}|}{d(v,p)^{0.75}} & \text{if } |E^{v,p}| \geq 2, \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

where  $|E^{v,p}|$  is the cardinality of  $E^{v,p}$ ,  $d(v,p)$  is the Euclidean distance between  $v$  and  $p$  and the empirically-chosen constant 0.75 discounts  $p$  with distance. This scoring function requires that  $|E^{v,p}| \geq 2$  for the POI to be considered, i.e.  $W^v$  and  $W^p$  must have at least two words in common.

4. If at least one POI in  $P^v$  has a match score of greater than 0, assign the single highest-scoring one to the visit. Otherwise, leave the visit unlabelled.

Starting with the 19.5 million photos taken in the top 200 cities, the visit creation and labelling process yielded 4.4 million visits, 1,431,234 of which were labelled. However, manual inspection of the labelled dataset revealed that roughly one third of POI assignments were incorrect, which I found could largely be attributed to the visit words originating from *Yahoo!*'s own WOEID tags. While I had expected this information to assist the labelling process, it was often running counter to the far more reliable content in users' own titles, descriptions and tags. In the original *YFCC100M*, the method by which POI-level WOEID tags are assigned is entirely opaque, and my findings here indicate that its accuracy is poor.

For this reason, I removed all WOEID tag information from the visit words and repeated the labelling process. In the language of binary classification, this change had the effect of greatly increasing the POI assignment precision, at the cost of lower recall: the total number of labelled visits was reduced to 977,745.

## 4.5 Label Bootstrapping via Word Likelihood Ratios

The operations outlined in the previous section yielded a labelled visit dataset that was of high quality, but utilised only a small fraction of the photos taken in each city. For this reason, I devised a *bootstrapping* approach that makes use of information from already-labelled visits to label additional visits.

Returning once more to the example of Cabot Tower in Bristol, the specific visit shown in Table 4.1, and the aforementioned POI words `{cabot, tower, william, venn, gough, folly, attraction}` from OSM: the evidence  $E^{v,p}$  here is `{cabot, tower, folly}`. Given this evidence, and the close proximity of the visit coordinates to those of the Cabot Tower entity in OSM, the visit would be labelled with this POI over any other. Crucially, those visit words that *do not* form part of the evidence (`{brandon, hill, park...}`) should not be ignored, since they may indicate other descriptive words and features that pertain to the POI, but are not on the OSM database. While one labelled visit from a single user provides little benefit, a statistical analysis of many visits could be a robust indicator of the most relevant descriptive words for each POI, which in turn can be used to identify additional visits which may have taken place there. With these

observations as motivation, I implemented the following bootstrapping algorithm that operated on city-level datasets:

1. Load a pre-existing dataset of visits for a city, a fraction of which will already be labelled with POIs by the method described in section 4.4.3.
2. Iterate through every visit in the city, *both labelled and unlabelled*, and maintain a running count of the appearances of all elements in the sets of visit words.
3. Once all visits have been seen, discard any words with fewer than 3 appearances. Divide the count values by the total number of visits to obtain a set of city-wide word frequencies  $F^*$ .
4. For each POI  $p$ , iterate through only those visits that have been labelled with  $p$  and similarly compute word counts and frequencies. Again discard words with fewer than 3 appearances. If a single user has visited  $p$  more than once, also discard any repeated words to avoid a potential source of bias. This yields a set of POI-specific word frequencies  $F^p$ .
5. Divide each frequency value in  $F^p$  by the corresponding value in  $F^*$  to give a *likelihood ratio* that indicates how much more likely the word is given  $p$ , versus in the city as a whole. Discard any words with a likelihood ratio of less than 10, yielding a set of (word, ratio) pairs  $R^p$ . Table 4.2 shows the highest-ratio words in  $R^p$  for three arbitrary POIs in the dataset.
6. Iterate through the unlabelled visits. For each visit  $v$ , assemble a shortlist of nearby POIs  $P^v$ , as during the primary labelling process.
7. For each POI  $p \in P^v$ , sum the likelihood ratios in  $R^p$  for the elements in  $W^v$ , skipping any words that do not appear at all. Denote this sum the *ratio sum*  $r^{v,p}$ .
8. Compute a bootstrap match score  $S_b^{v,p}$  as a function of both the ratio sum and the distance between the coordinates of the visit and the POI:

$$S_b^{v,p} = \frac{r^{v,p}}{d(v, p)^{0.1}} \quad (4.2)$$

where  $d(v, p)$  is the Euclidean distance between  $v$  and  $p$  and the empirically-chosen constant 0.1 discounts  $p$  with distance (far less harshly than in primary labelling).

9. If at least one POI in  $P^v$  has a bootstrap match score of at least 30, assign the single highest-scoring one to the visit. Otherwise, leave the visit unlabelled.

Table 4.2: Words with the highest likelihood ratios for three POIs in the dataset.

POI ( $p$ )	City	Top five highest-ratio words in $R^p$ , with values shown.
SS Great Britain	Bristol	$\{(transatlantic, 70.2), (stern, 61.5), (steamship, 61.5), (propeller, 61.5), (screw, 61.5)\}$
Camp Nou	Barcelona	$\{(futebol, 221.5), (supercopa, 196.9), (bojan, 177.2), (marcador, 168.7), (lionel, 147.6)\}$
Deutsches Museum	Munich	$\{(astronaut, 229.6), (rockets, 229.6), (pendulum, 191.3), (foucault, 191.3), (physics, 143.5)\}$

I determined values for the various constants used in the algorithm through a process of experimentation; these values were seen to maximise the proportion of true positive labels while minimising false positives.

The high-ratio words for the exemplar POIs in Table 4.2 hint at the potential effectiveness of this bootstrapping algorithm with my dataset. Words with high likelihood ratios tended to align intuitively with how a human with good knowledge of a particular POI may describe it. There were also many interesting cases of high-ratio words that a human may not think to include. For sports stadia, the abbreviation `vs` had a consistently high ratio across many cities, and accordingly its presence in  $W_v$  for an unlabelled visit seen as a strong indicator that the user was indeed at a stadium. For zoos, the names of various wild animals (e.g. `cheetah`, `kangaroo`, `capybara`) were similarly useful. Bootstrapping also increased the degree of multi-lingual support; English translations of foreign POI names often had some of the highest likelihood ratios.

Overall, bootstrapping increased the number of labelled visits to 1,277,112, an improvement of 31% over the non-bootstrapped dataset.

## 4.6 Final Dataset

The final form of the travel histories dataset consists of 1.3 million POI-labelled visits made by 64,826 *Flickr* users in 200 cities across the world. Figure 4.2 shows the distribution of labelled visit counts across all cities. The degree of variability in per-city visit counts is significantly larger than for the photo counts shown in figure 4.1, varying from 109,615 for New York City to just 13 for Las Pavas in Colombia. That said, a significant majority of the cities (ranks 28 to 163) have between 1,000 and 10,000 visits.

The dataset is stored as a single 59MB JSON file with the following schema:

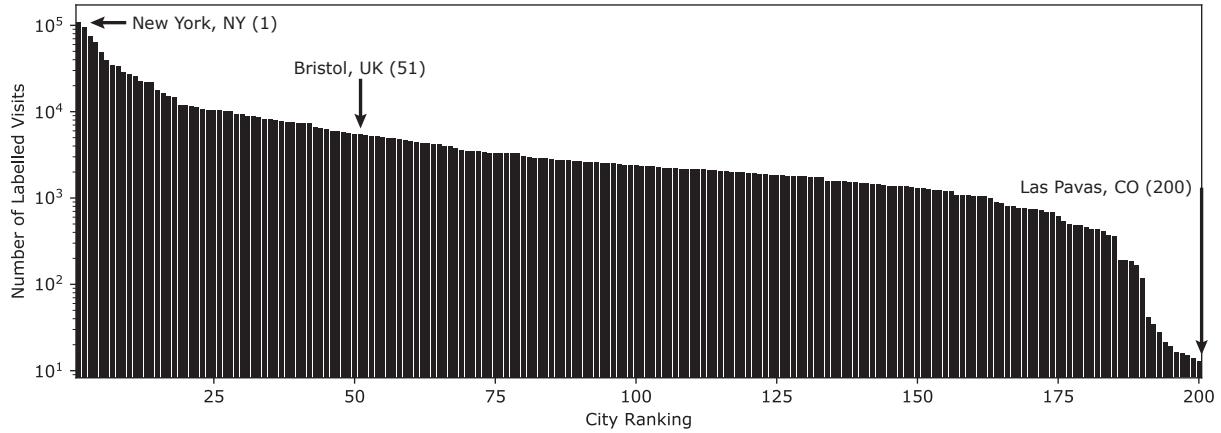


Figure 4.2: Distribution of labelled visits in the 200 most popular cities in the *YFCC100M* dataset. Annotations indicate rankings of several of these cities.

```
{
  'Users': {
    'User ID': {
      'City': {
        'Total Num Photos': __
        'Visits': [
          {'POI': __, 'Start': __, 'End': __, 'Num Photos': __},
          ...
        ]
      },
      ...
    }
  }
  'POI Categories' : {
    'POI': __
    ...
  }
  'POI Locations' : {
    'POI': [__, __],
    ...
  }
}
```

The core elements of the dataset are individual visits. Each visit entry contains the POI (as represented by the unique *OpenStreetMap* identifier), the start and end timestamps in `yyyy-mm-dd HH:MM:SS` format, and the number of photos taken. Visits

are categorised at the highest level by user, then by city, and placed in chronological order within each city. The dataset also includes:

- The total number of photos taken by each user in each city, including those in visits that have not been labelled. In rare cases, a user has non-zero photos in a city but zero labelled visits, in which case their `Visits` list is empty ([]).
- Separate subschemas containing the category name and coordinates ([latitude, longitude]) of each POI. These do not need to be sorted by city since each POI identifier is globally unique.

## 4.7 Data Quality Assessment

Before embarking on the implementation stage of the project, I sought to estimate the quality and cleanliness of the dataset through a variety of means, in order to demonstrate that it would be suitable for use within the city- and POI-level recommender systems.

### 4.7.1 Manual Inspection

The first and most straightforward mode of assessment was to sample a small number of labelled visits from the dataset, identify the photos from which each was assembled, find these photos on *Flickr*, and manually check whether the POI assigned during the labelling process matched the true location. For each of the 200 cities, I sampled one visit at random, since this would avoid biasing the assessment towards the most popular destinations (which tend to be in European or English-speaking countries). In a handful of cases, the photos were no longer available on *Flickr*, which generally indicated that the originating user had deleted their account. In such cases, I sampled another visit from the same city.

Table 4.3 shows five of the visits labelled assessed. The visits in Bologna, Düsseldorf and Santiago are definitively correct, and it is worth noting how the bootstrapping algorithm has been used in the first two of these cases to identify a POI based on an informal English-language description (*two leaning towers*) or the reason for which it is known (as a building [*gebäude*] designed by famed *architect*, *Frank Gehry*). The visit in Singapore has been incorrectly labelled, since the photo shown was taken on a road with the same name as the assigned park. The visit in Bristol is somewhat ambiguous: the cottage pictured is a former constituent of the Blaise Castle Estate, but is not the castle itself.

Table 4.4 summarises the statistical results of my survey of 200 labelled visits. In cases where the label was not definitively correct, I attempted to classify the nature of the error or uncertainty. Three classes of erroneous cases emerged. In the first, the true and assigned POI were near to each other, and both named after the same person,

Table 4.3: Five labelled visits from the dataset, including a selected photo from each that illustrates the location. The POI assigned during labelling is shown, alongside its category, as well as the set of words used as evidence. “BS” indicates that the visit was labelled in the bootstrapping stage rather than in the initial sweep through the dataset. The Y/N column shows whether the visit has been correctly labelled (Y = yes; N = no;  $\sim$ Y = ambiguous).

City	Assigned POI	Evidence	Example Photo	Y/N
Bologna	Asinelli Tower (tower)	{two, leaning, towers} (BS)		Y
Bristol	Blaise Castle (castle)	{blaise, castle, estate}		$\sim$ Y
Düsseldorf	Neuer Zollhof (building)	{frank, architect, gebäude} (BS)		Y
Santiago	Palacio de La Moneda (attraction)	{palacio, de, la, moneda}		Y
Singapore	Ann Siang Hill Park (park)	{ann, sing, hill}		N

Table 4.4: Classification of 200 labelled visits considered in manual assessment.

Classification	Count
Definitively correct	152
Ambiguous: assigned POI too specific	8
Ambiguous: POI not shown but mentioned	5
Incorrect: shared place/area name	4
Incorrect: inaccurate location	6
Incorrect: assorted unlucky keywords	11

organisation or area. In the second, the coordinates of the visit were inaccurate, so the true POI was outside of the search radius of the labelling algorithm. In the third, the set of visit words was simply ‘unlucky’ in that it contained more matches to the incorrect POI than the true one. There were also two kinds of ambiguous case: those in which the assigned POI represented only one specific part of the location that was actually visited (e.g. a single university building or monument in a town square), and those in which the originating *Flickr* user alluded to having visited the assigned POI in the photo title or description, but it was not visible in the photo itself. In the aggregate, this small-scale manual assessment suggests that the dataset boasts between 76% and 85% labelling precision, depending on how strictly a ‘correct’ label is defined.

### 4.7.2 Statistical Properties

It is unreasonable to expect that the integrity of a dataset of 1.3 million labelled visits can be fully assessed using a sample size of just 200. For that reason, I analysed the statistics of the dataset to determine whether it was consistent with expectations derived from general domain knowledge. This in turn would provide evidence against large-scale systematic errors. Specifically, my expectations were that:

- Different POI categories would be popular at different times of the day and year;
- POIs that are widely known as ‘top attractions’ would receive the most visits.

#### Diurnal and Seasonal Visitation Patterns

To assess agreement with the first expectation, I collected all labelled visits to a selection of popular POI categories for which pronounced diurnal or seasonal variations in popularity seemed likely (e.g. restaurants, parks, museums, bars and beaches) and constructed histograms of the hour or month component of the start time<sup>3</sup>. Figure 4.3 contains the diurnal visitation trends for twelve POI categories of interest, and figure 4.4 contains the seasonal trends for nine categories.

---

<sup>3</sup>For Southern Hemispheres, dates were shifted by 6 months as initially described in section 3.2.1.

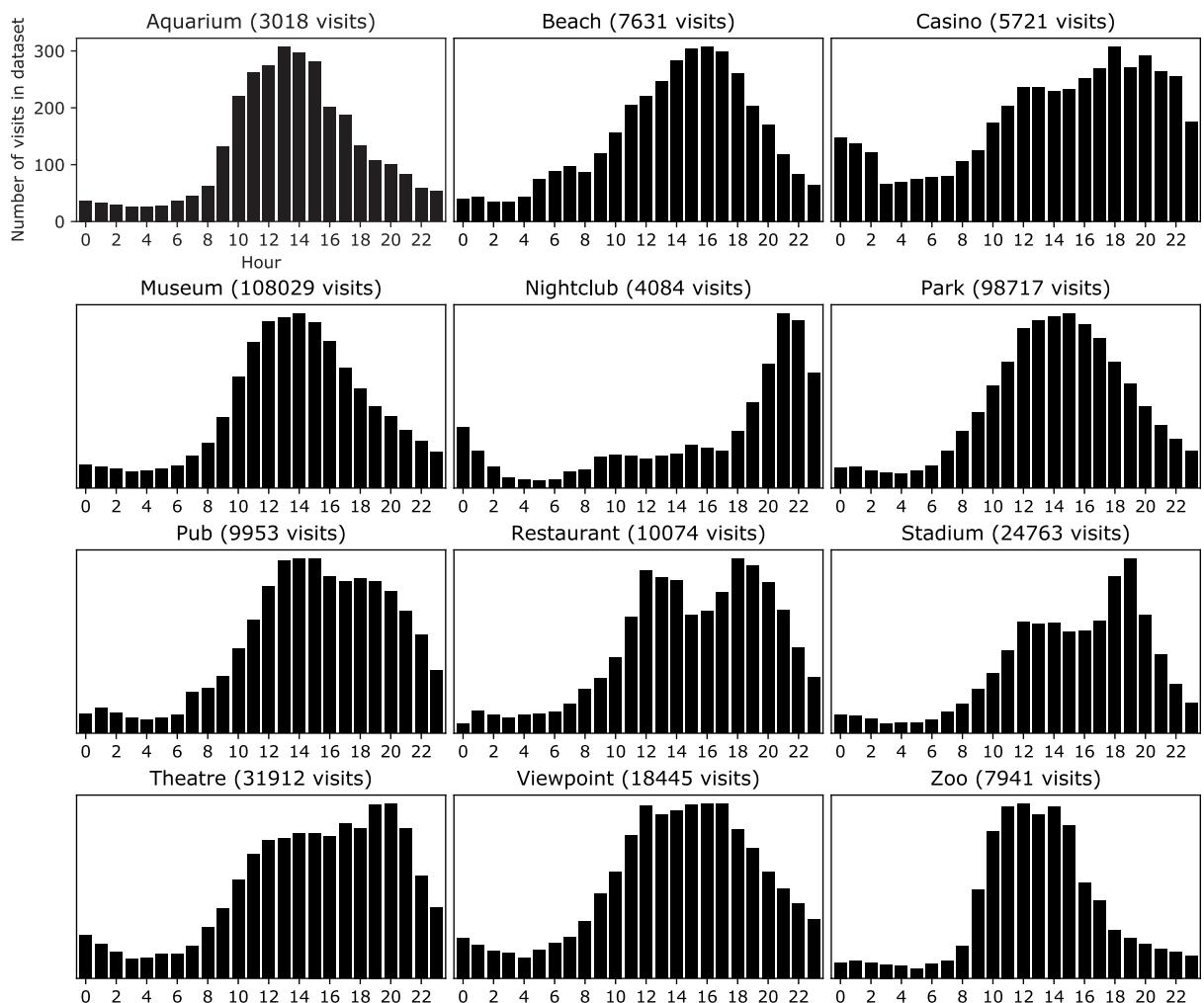


Figure 4.3: Distributions of the hour component of visit times for twelve POI categories.

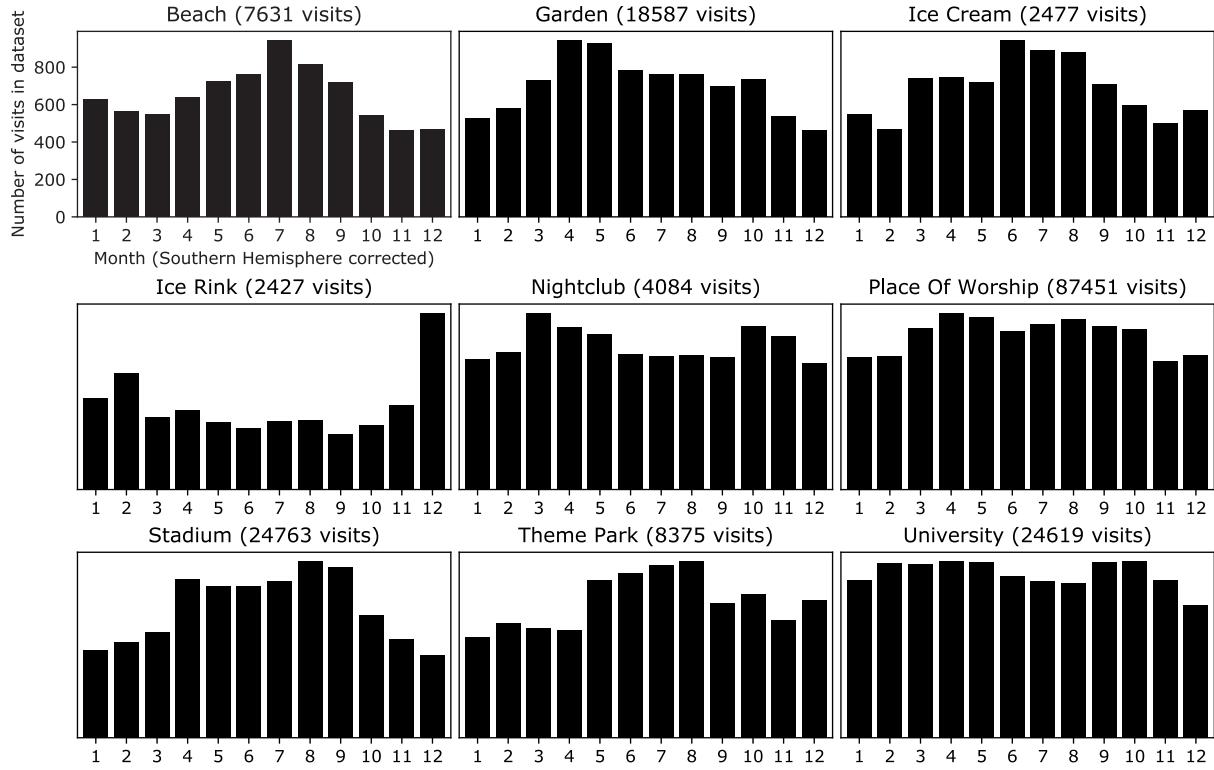


Figure 4.4: Distributions of the month component of visit times for nine POI categories.

Looking first at figure 4.3, the following trends are visible that align well with expectation. As outdoor public spaces, beaches and parks are visited throughout the day, with smoothly-varying distributions that peak in mid-afternoon. The former also shows a minor peak in the early morning, which may be due to visitors wishing to capture images of sunrise. In contrast, aquaria and zoos are attractions with clearly-defined daytime opening hours, and their distributions share sharp rises and falls. Restaurants show two clear peaks in attendance around lunch and dinner, while stadia have a sharp spike to match the common early-evening fixture time of sporting matches. A smaller spike is also visible around the most popular showing time of theatres. Pubs are visited during lunchtime and throughout the afternoon and evening. Their popularity drops off sharply around 9pm, which coincides with a sharp increase in the number of visits to nightclubs. Meanwhile, casinos are attended at all hours, with a surge in the late evening that is sustained until well past midnight. The fact that all categories, including overtly daytime ones, show non-negligible visitation in the small hours of the morning is probably attributable to cameras with miscalibrated timezones.

In general, the seasonal variations in visit frequency shown in figure 4.4 are less dramatic than the diurnal ones. However, more visits are recorded in summer months than winter months, aligning with the typical pattern of tourism. Summer peaks are particularly visible for beaches, ice cream sellers, and to a slightly lesser extent, theme parks. Ice rinks present a dramatic exception, being most popular in the middle of winter,

while gardens are most frequently attended around the springtime bloom. Stadia are more popular in summer, the height of the sporting calendar in many countries. Visits to places of worship are relatively constant throughout the year, with a small increase towards a maximum around Easter. Finally, the fact that both universities and nightclubs exhibit peaks during academic term times seems unlikely to be a coincidence.

## Top Attraction Agreement

In the second statistical assessment method, I compared an externally-sourced ranking of popular tourist sites (specifically, the ‘Top 10 Attractions’ section on *TripAdvisor.com* [51]) with the ranking of POI visits within the dataset in a selection of cities<sup>4</sup>. I used ten cities for this assessment, the results of which are shown in table 4.5.

Out of the 100 top attractions listed in the table, 36 also appear in the top 10 of their respective city rankings by number of labelled visits, and 52 appear in the top 20. These statistics suggest that renowned POIs do indeed receive amongst the highest visitation in the dataset, as expected. In 19 instances, no corresponding POI could be found for an attraction, most notably in the city of Osaka. This is largely due to the limitations of the *OpenStreetMap* service; not every attraction has a corresponding POI on the OSM database with the same name. In addition, several extremely famous POIs, such as the Musee de l’Orangerie in Paris and Imperial Palace in Vienna, appear surprisingly low down the popularity rankings. Such occasional omissions are likely unavoidable, since my dataset creation method relies on the photo-upload habits of a random selection of *Flickr* users, including the reliable mentioning of correct POIs in titles, descriptions or tags. In general, however, this assessment provided good evidence that the dataset features most popular POIs in appropriate quantities.

---

<sup>4</sup>While I used exactly the top 10 *TripAdvisor* attractions where possible, I skipped cases where the listed entries were ephemeral events (e.g. the Edinburgh Military Tattoo), loosely-defined areas (e.g. Budapest Old Town), or locations outside of the city proper.

Table 4.5: *TripAdvisor* top attraction lists for ten cities, with corresponding popularity ranks in the travel histories dataset. A dash indicates that no corresponding POI exists in the dataset.

<b>Chicago (1272 POIs)</b>		<b>Paris (2194 POIs)</b>		<b>Budapest (454 POIs)</b>		<b>Delhi (188 POIs)</b>		<b>Edinburgh (693 POIs)</b>	
Art Institute	2	Musee d'Orsay	12	Hungarian Parliament	5	Qutub Minar	11	Arthur's Seat	16
Millenium Park	4	Notre-Dame	3	Halaszbastya	-	Gurudwara Bangla Sahib	14	Royal Yacht Britannia	45
Museum of Science and Industry	22	The Louvre	2	St Stephen's Basilica	1	Swaminarayan Akshardham	115	National Museum of Scotland	8
360 Chicago	6	Saint-Chapelle	24	Buda Castle	30	Humayun's Tomb	7	Edinburgh Castle	3
Cloud Gate	1	Eiffel Tower	1	Shoes on the Danube	50	Lodhi Garden	36	Camera Obscura	21
Wrigley Field	7	Palais Garnier	17	Margaret Island	193	Chandni Chowk	12	Royal Mile	2
Magnificent Mile	-	Seine River	-	Gellert Hill	20	India Gate	5	Palace of Holyroodhouse	50
Chicago Riverwalk	20	Arc de Triomphe	6	Varhegy	182	Rashtrapati Bhavan	-	Calton Hill	6
Field Museum	9	Musee de l'Orangerie	137	Szechenyi Chain Bridge	2	Mughal Garden	-	Royal Botanic Garden	9
Chicago Cultural Center	25	Montmarte	25	Matthias Church	7	Hauz Khas Village	25	Forth Road Bridge	-
<b>Glasgow (400 POIs)</b>		<b>Osaka (115 POIs)</b>		<b>Perth (255 POIs)</b>		<b>Toronto (1120 POIs)</b>		<b>Vienna (623 POIs)</b>	
Kelvingrove Museum	1	Osaka Castle Park	5	Kings Park and Botanic Garden	21	Ripley's Aquarium of Canada	146	Schonbrunn Palace	15
Riverside Museum	14	Kuchu Teien Observatory	-	The Perth Mint	-	Royal Ontario Museum	11	Kunsthistorisches Museum	10
Glengoyne Distillery	53	Osaka Aquarium	1	The Bell Tower	5	CN Tower	2	Tiergarten Schoenbrunn	6
University of Glasgow	56	Kids Plaza	-	State War Memorial	7	St Lawrence Market	-	Imperial Palace	244
Celtic Park	26	National Bunraku Theater	-	Lotterywest Federation Walkway	100	Hockey Hall of Fame	16	Belvedere Museum	20
The Necropolis	-	Mint Museum	-	Art Gallery of Western Australia	4	Art Gallery of Ontario	9	Albertina	11
Tenents Wellpark Brewery	-	Nakanoshima Park	-	Queens Gardens	249	Steam Whistle Brewery	78	Natural History Museum	8
The Clydeside Distillery	-	Sumiyoshi Taisha Shrine	51	Scitech	-	High Park	18	Wiener Staatsoper	2
Glasgow Botanic Gardens	5	Tempozan Ferris Wheel	-	Hyde Park	11	Casa Loma	29	St. Stephen's Cathedral	134
Glasgow Science Centre	3	Kuromon Market	-	RAC Arena	148	Rogers Centre	3	Leopold Museum	53

### 4.7.3 Comparison to Existing Dataset

As part of one prior work identified in the literature review [41], Lim et al. assembled and published a dataset of travel histories based on *YFCC100M*. In this dataset, *Wikipedia* is used as the source of ground-truth POI names and locations instead of OSM. The labelling process is also simpler: photos are considered individually rather than grouped into visits, POIs are assigned based on physical proximity alone, and user-provided titles, descriptions and tags are not utilised. Finally, the dataset concerns just eight cities instead of 200. These cities are Budapest, Delhi, Edinburgh, Glasgow, Osaka, Perth, Toronto and Vienna; the latter eight shown in table 4.5. Table 4.6 compares the size of this dataset (**B**) to the corresponding subset of the dataset created in this project (**A**) in terms of the number of photos used to create it (i.e. those both inside and outside of labelled visits in the case of **A**), and the number of users, POIs and visits.

Table 4.6: Size comparison of travel histories dataset used in this project (**A**) with dataset created by Lim et al.

	# Photos		# Users		# POIs		# Visits	
	<b>A</b>	<b>B</b>	<b>A</b>	<b>B</b>	<b>A</b>	<b>B</b>	<b>A</b>	<b>B</b>
Budapest	93849	36000	868	935	454	39	3880	18513
Delhi	36016	13919	395	279	188	26	1521	3993
Edinburgh	144038	82060	1816	1454	693	29	12007	33944
Glasgow	68034	29019	747	601	400	29	3804	11434
Osaka	48711	392420	133	450	115	29	273	7747
Perth	64882	<i>Not stated</i>	252	159	255	25	922	3643
Toronto	281633	157505	2115	1395	1120	30	16681	39419
Vienna	169611	85149	1445	1155	623	29	7784	34515
<b>Total</b>	<b>906774</b>	<b>&gt;796072</b>	<b>7771</b>	<b>6428</b>	<b>3848</b>	<b>236</b>	<b>46872</b>	<b>153208</b>

Aside from containing a comparable number of users, **A** and **B** have quite dramatic statistical differences. In **A**, I consider more POIs (hundreds per city as opposed to roughly 30 of the most popular ones). In **B**, a higher proportion of photos are mapped into visits (18.7% instead of 5.2%). As a consequence, **B** has many more visits per POI on average. However, due to the less sophisticated labelling process used in **B**, there is a far greater risk of each visit being erroneous (i.e. the originating Flickr user did not actually visit the assigned POI).

Following the same method described in section 4.7.1, I manually assessed a selection of visits in **B** by viewing the corresponding photo on Flickr. For each of the eight cities I checked 25 visits, bringing the total to 200 (matching the previous assessment of my dataset **A**). Table 4.7 contains the results, with a slightly different categorisation of ambiguous cases. Without inside knowledge of the dataset-generation process, I was unable to diagnose and categorise the incorrect cases.

This assessment indicates that the reliability of POI assignment in **B** is significantly

Table 4.7: Classification of 200 visits in dataset created by Lim et al.

Classification	Budapest	Delhi	Edinburgh	Glasgow	Osaka	Perth	Toronto	Vienna	Total
Definitively correct	9	22	8	5	12	7	18	10	<b>91</b>
Ambiguous: assigned POI too specific	0	0	0	4	0	0	0	1	<b>5</b>
Ambiguous: POI nearby	2	0	5	0	3	2	2	0	<b>14</b>
Ambigious: no clear POI shown	4	0	1	0	2	2	0	0	<b>9</b>
Incorrect	10	3	11	16	8	14	5	14	<b>81</b>

below that of **A**. Depending on which ambiguous cases are considered acceptable, the precision lies between 46% and 60%, which is on the order of 20-30% lower. Therefore, despite the smaller number of per-city and per-POI visits, **A** certainly appears to be a higher quality dataset in terms of accuracy. It also has the additional benefit of scale, covering 200 cities instead of eight, providing greater opportunity for the quantification of global trends as explored in this project. I concluded the assessment process with confidence that my travel histories dataset was suitable for use with the city- and POI-level recommender models.

# Chapter 5

## Application of Models to Dataset

With both the structure of the recommendation models and the travel histories dataset in place, the next stage of the project was to manipulate the data into a form that could be fed into the various equations, and implement and optimise the models themselves.

### 5.1 Implementation Notes

All implementation work completed in this project was done in the Python programming language, with frequent use of the `pandas` library for data manipulation, as well as `numpy` for mathematical operations and `scipy` for statistical analysis. Rather than implementing the models with a view towards efficient end-user deployment, for example with the aid of a graphical user interface, I prioritised ease-of-evaluation on the static dataset of travel histories. For example, where I had conceived of multiple models or techniques for performing the same operation, I ran them all simultaneously and collected results from each. Since the travel histories dataset is far more compact than the original *YFCC100M* data from which it was derived (59MB compared with 16.4GB), I was able to deploy all necessary scripts on my personal computer, thereby accelerating development time.

### 5.2 Construction of Data Structures

#### 5.2.1 City Recommendation

Each of the abstract mathematical objects introduced in section 3.1.1 needed to be embodied as a Python data structure. I constructed these data structures from the JSON-format dataset as follows<sup>1</sup>:

---

<sup>1</sup>As latent objects, the potential enjoyment functions  $\text{enj}^t$  could not be derived directly from the dataset. Section 6.1.1 describes how I emulated the values for evaluation purposes.

- The set of tourists  $\mathbb{T} = \{t_1, \dots, t_k\}$  = the set of all `User ID` values, so that  $k = 64,826$ .
- The tuple of cities  $\mathbb{C} = (c_1, \dots, c_l)$  = the set of all `City` values, alphabetical order. Since the dataset contains 200 cities,  $l = 200$ .
- The tuple of POI categories  $\mathbb{X} = (x_1, \dots, x_m)$  = the set of values in `POI Categories` whose constituent POIs appear at least 500 times across the dataset, in alphabetical order. In the final dataset,  $m = 74$ .
- For a tourist  $t$ , the set of visited cities  $\mathbb{V}^t = \{c_1, \dots, c_a\}$  = the set of all `City` keys in the corresponding subschema of the dataset. Depending on the tourist, the cardinality varies from  $a = 1$  to  $a = 64$ .
- For a tourist  $t$ , the set of unvisited cities  $\mathbb{U}^t = \{c_1, \dots, c_b\} = \text{set}(\mathbb{C}) - \mathbb{V}^t$ .
- For a tourist  $t$ , the vector of city enjoyment values  $\mathbf{e}^t = [\mathbf{e}_1^t, \dots, \mathbf{e}_l^t]$  = a function of the total number of *photos* (not labelled visits) uploaded by that tourist in each city. My underlying assumption here was that tourists upload more photos of cities that they find more enjoyable and wish to remember more vividly. Rather than taking the photo count values directly, I defined  $\mathbf{e}^t$  as:

$$\mathbf{e}^t = \frac{\mathbf{a}^t}{\|\mathbf{a}^t\|} \quad \text{where} \quad \forall i \in [1..l] \quad \mathbf{a}_i^t = \sqrt{\frac{\left( \frac{\#_{\mathbb{C}_i}^t}{\sum_{j=1}^l \#_{\mathbb{C}_j}^t} \right)}{\left( \frac{\sum_{t' \in \mathbb{T}} \#_{\mathbb{C}_i}^{t'}}{\sum_{t' \in \mathbb{T}} \sum_{j=1}^l \#_{\mathbb{C}_j}^{t'}} \right)}} \quad (5.1)$$

and  $\#_{\mathbb{C}_i}^t$  is the number of photos uploaded by tourist  $t$  in city  $\mathbb{C}_i$ , i.e. the ‘Total Num Photos’ value in the corresponding subschema of the dataset. This equation computes the proportion of the tourist’s photos taken in each city (numerator), expressed as a ratio of the *global* proportion of photos in that city (denominator). Using the global proportions in this way avoids biasing towards the most popular cities. My rationale here was that many tourists have been to London, Paris or New York, so the fact that two individuals have been there says little about their similarity, whereas common visits to, for example, Bologna is a stronger indicator of common preferences. The square root is used to reduce the variability of these values – it seems unlikely that a tourist who has uploaded twice as many photos of a city enjoyed it twice as much – and the normalisation is important to ensure that  $\mathbf{e}^t$  is a valid probability distribution.

- For a tourist  $t$ , the vector of POI category preference values  $\mathbf{p}^t = [\mathbf{p}_1^t, \dots, \mathbf{p}_m^t]$  = a function of the total number of photos uploaded by that tourist as part of labelled

visits to each category in  $\mathbb{X}$  (i.e. only those with at least 500 visits in total). This rests on a similar assumption to the one used for  $\mathbf{e}^t$  and as such, I defined  $\mathbf{p}^t$  in a similar manner:

$$\mathbf{p}^t = \frac{\mathbf{b}^t}{\|\mathbf{b}^t\|_1} \quad \text{where} \quad \forall i \in [1..m] \quad \mathbf{b}_i^t = \frac{\left( \frac{\#_{\mathbb{X}_i}^t}{\sum_{j=1}^m \#_{\mathbb{X}_j}^t} \right)}{\left( \frac{\sum_{t' \in \mathbb{T}} \#_{\mathbb{X}_i}^{t'}}{\sum_{t' \in \mathbb{T}} \sum_{j=1}^m \#_{\mathbb{X}_j}^{t'}} \right)} \quad (5.2)$$

and  $\#_{\mathbb{X}_i}^t$  is the total number of photos uploaded by tourist  $t$  within visits to category  $\mathbb{X}_i$ . My rationale here was again largely the same as that for equation 5.1: common visits to less popular POI categories such as ice rinks or casinos are more informative than common visits to museums or places of worship. I opted to exclude the square root in this equation, because there was less extreme variability in per-category photo counts compared with per-city.

- For a city  $c$ , the vector of POI category importance values  $\mathbf{i}^c = [\mathbf{i}_1^c, \dots, \mathbf{i}_m^c]$  = a function of the total number of photos uploaded in that city as part of labelled visits to each category in  $\mathbb{X}$ . Mirroring the definition of  $\mathbf{p}^t$ , I defined  $\mathbf{i}^c$  as:

$$\mathbf{i}^c = \frac{\mathbf{c}^c}{\|\mathbf{c}^c\|_1} \quad \text{where} \quad \forall i \in [1..m] \quad \mathbf{c}_i^c = \frac{\left( \frac{\#_{\mathbb{X}_i}^c}{\sum_{j=1}^m \#_{\mathbb{X}_j}^c} \right)}{\left( \frac{\sum_{c' \in \mathbb{C}} \#_{\mathbb{X}_i}^{c'}}{\sum_{c' \in \mathbb{C}} \sum_{j=1}^m \#_{\mathbb{X}_j}^{c'}} \right)} \quad (5.3)$$

and  $\#_{\mathbb{X}_i}^c$  is the total number of photos uploaded in city  $c$  within visits to category  $\mathbb{X}_i$ . Yet again, the use of ratios with respect to global proportions is important here: many cities have museums, but relatively few have popular beaches, so it makes sense to boost the importance of the latter.

### 5.2.2 POI Recommendation

In addition to some of those data listed above, the abstract mathematical objects introduced in section 3.2.1 also needed to be embodied as Python data structures. I constructed these data structures from the JSON-format dataset as follows<sup>2</sup>:

- For a city  $c$ , the set of POIs  $\mathbb{P}^c = \{p_1, \dots, p_{o^c}\}$  = the set of POI values that appear within `Visits` lists for that city.
- The `POI → category` mapping function `cat` = looking up the POI in the `POI Categories` subschema of the dataset.

---

<sup>2</sup>As in the city recommendation problem, the potential enjoyment functions  $\text{enj}^{t, now}$  could not be derived directly from the dataset. Section 6.1.2 describes how I emulated the values for evaluation purposes.

- The `POI` → location mapping function `loc` = looking up the `POI` in the `POI Locations` subschema of the dataset.
- For a tourist  $t$  and city  $c$ , the tuple of POI visits  $\mathbb{H}^{t,c} = (v_1, \dots, v_{q^{t,c}}) =$  the corresponding `Visits` list. If  $c$  does not appear as an entry in  $t$ 's subschema, or if the `Visits` list is empty,  $\mathbb{H}^{t,c} = ()$ .

The `poi`, `startTime` and `endTime` functions required no explicit definition since this information is contained within the `Visits` lists themselves. For month and hour, I parsed the required values from the full timestamps using the `datetime` Python library, and made the requisite 6-month date shifts for Southern Hemisphere POIs.

### 5.2.3 Algorithmisation of Models

### 5.2.4 City Recommendation

The construction of the required data structures was the most challenging task faced during implementation of the city recommender models, and the rest mostly involved substituting the values into the various equations presented in section 3.1. Algorithm 1 outlines the complete operation of the Python script I wrote to generate city rankings for a tourist group<sup>3</sup>  $\mathbb{G}$ . It returns tuples representing rankings of all unvisited cities in order of recommendation score according to each of the three recommender models, with each of the two group aggregation methods.

To further illustrate the operation of the recommender algorithm, the following is an example of an arbitrary group of real tourists taken from the dataset, and the various city rankings output by the system. The group consists of three tourists, with *Flickr* photo upload histories as shown in table 5.1.

Table 5.2 contains the top 5 cities in the rankings created for this group by the system. For the tourist-tourist model (*TT*) the neighbourhood size  $n$  was set to 50, and for the group profile vectors and city scores sum-based aggregation was used throughout.

In all of the rankings, historic northern European cities appear near the top, with Cologne occupying the number 1 spot in the tourist-tourist and tourist-city rankings, and Salzburg placing first in the city-city rankings. German cities are particularly prevalent, a good result given that all three tourists have visited at least one place in Germany in the past. The recommendations also appear reasonable with respect to the tourists' POI category visitation: many of the highly-ranked cities are famed for their historic architecture including city gates; beer gardens are extremely prevalent in Germany and neighbouring countries; both Salzburg and Cologne are riverside cities with tourist ferries; and Cologne is home to an internationally-renowned zoo. While such observations are

---

<sup>3</sup>The same script also works in the single-tourist case. If only one `User` ID is specified, the steps pertaining to the group recommendation problem are skipped.

---

**Algorithm 1:** City recommendation for a multi-tourist group

---

**inputs** : the JSON-format dataset file; a set of `User` ID values  $\mathbb{G}$  to define the tourists in the group  
**outputs:** city rankings from each of the three recommender models, with each of the two group aggregation methods if  $|\mathbb{G}| > 1$

```
// Assemble data structures
 $\mathbb{T} \leftarrow$  set of all User ID values in the dataset;
 $\mathbb{C} \leftarrow$  tuple of unique City values, alphabetical order;
 $\mathbb{X} \leftarrow$  tuple of unique POI Category values with  $\geq 500$  visits, alphabetical order;
for  $t \in \mathbb{T}$  do
     $\mathbf{e}^t \leftarrow$  vector derived from per-city photo counts for  $t$  (equation 5.1)
for  $t \in \mathbb{G}$  do
     $\mathbb{V}^t \leftarrow$  set of all cities with at least 1 photo by  $t$ ;
     $\mathbb{U}^t \leftarrow \mathbb{C} - \mathbb{V}^t$ ;
     $\mathbf{p}^t \leftarrow$  vector derived from per-category photo counts for  $t$  (equation 5.2)
for  $c \in \mathbb{C}$  do
     $\mathbf{i}^c \leftarrow$  vector derived from per-category photo counts for  $c$  (equation 5.3)

// Run recommender models for each tourist in the group
for  $t \in \mathbb{G}$  do
    for  $t' \in \mathbb{T} - \mathbb{G}$  do
         $\text{sim}(t, t') \leftarrow$  function of  $\text{JSD}(\mathbf{e}^t || \mathbf{e}^{t'})$  (equation 3.5)
         $\mathbb{N}^t \leftarrow$  set of  $n$  most similar tourists to  $t$ ;
        for  $i : \mathbb{C}_i \in \mathbb{U}^t$  do
            for  $c' \in \mathbb{V}^t$  do
                 $\text{sim}(\mathbb{C}_i, c') \leftarrow$  function of  $\text{JSD}(\mathbf{i}^{\mathbb{C}_i} || \mathbf{i}^{c'})$  (equation 3.7)
                 $\text{sim}(t, \mathbb{C}_i) \leftarrow$  function of  $\text{JSD}(\mathbf{p}^t || \mathbf{i}^{\mathbb{C}_i})$  (equation 3.9);
                 $S_{TT}^t(\mathbb{C}_i) \leftarrow$  function of  $[\mathbf{e}_i^{t'} \text{ and } \text{sim}(t, t') \forall t' \in \mathbb{N}^t]$  (equation 3.6);
                 $S_{CC}^t(\mathbb{C}_i) \leftarrow$  function of  $\mathbf{e}_i^t$  and  $[\text{sim}(\mathbb{C}_i, c') \forall c' \in \mathbb{V}^t]$  (equation 3.8);
                 $S_{TC}^t(\mathbb{C}_i) \leftarrow \text{sim}(t, \mathbb{C}_i)$  (equation 3.10)
             $\mathbb{R}_{TT}^t, \mathbb{R}_{CC}^t$  and  $\mathbb{R}_{TC}^t \leftarrow U^t$  ordered by  $S_{TT}^t, S_{CC}^t$  and  $S_{TC}^t$  respectively
```

*continued overleaf...*

---

---

```

if  $|\mathbb{G}| > 1$  then
    // Assemble components of group profile
     $\mathbb{V}^{\mathbb{G}} \leftarrow$  union of  $\mathbb{V}^t$  sets  $\forall t \in \mathbb{G}$  (equation 3.13);
     $\mathbb{U}^{\mathbb{G}} \leftarrow$  intersection of  $\mathbb{U}^t$  sets  $\forall t \in \mathbb{G}$  (equation 3.11);
     $\mathbf{e}^{\mathbb{G}} \leftarrow$  sum/max aggregation of  $[\mathbf{e}^t \forall t \in \mathbb{G}]$  (equation 3.14);
     $\mathbf{p}^{\mathbb{G}} \leftarrow$  sum/max aggregation of  $[\mathbf{p}^t \forall t \in \mathbb{G}]$  (equation 3.15);

    // Run recommender models for group profile (AoP)
    Same method as for individual tourists; yields  $\mathbb{R}_{TTP}^{\mathbb{G}}$ ,  $\mathbb{R}_{CCP}^{\mathbb{G}}$  and  $\mathbb{R}_{TCP}^{\mathbb{G}}$  ;

    // Combine individual town rankings into group rankings (AoS)
    for  $c \in \mathbb{U}^{\mathbb{G}}$  do
         $S_{TTR}^{\mathbb{G}}(c) \leftarrow$  sum/max aggregation of  $[S_{TT}^t(c) \forall t \in \mathbb{G}]$  (equation 3.12);
         $S_{CCR}^{\mathbb{G}}(c) \leftarrow$  sum/max aggregation of  $[S_{CC}^t(c) \forall t \in \mathbb{G}]$  (equation 3.12);
         $S_{TCR}^{\mathbb{G}}(c) \leftarrow$  sum/max aggregation of  $[S_{TC}^t(c) \forall t \in \mathbb{G}]$  (equation 3.12)
         $\mathbb{R}_{TTR}^{\mathbb{G}}$ ,  $\mathbb{R}_{CCR}^{\mathbb{G}}$  and  $\mathbb{R}_{TCR}^{\mathbb{G}} \leftarrow \mathbb{U}^{\mathbb{G}}$  ordered by  $S_{TTR}^{\mathbb{G}}$ ,  $S_{CCR}^{\mathbb{G}}$  and  $S_{TCR}^{\mathbb{G}}$  respectively;
    return  $\mathbb{R}_{TTP}^{\mathbb{G}}$ ,  $\mathbb{R}_{TTR}^{\mathbb{G}}$ ,  $\mathbb{R}_{CCP}^{\mathbb{G}}$ ,  $\mathbb{R}_{CCR}^{\mathbb{G}}$ ,  $\mathbb{R}_{TCP}^{\mathbb{G}}$ ,  $\mathbb{R}_{TCR}^{\mathbb{G}}$ 

else
    return  $\mathbb{R}_{TT}^t$ ,  $\mathbb{R}_{GG}^t$  and  $\mathbb{R}_{TG}^t$  where  $\mathbb{G} = \{t\}$ 

```

---

Table 5.1: Summary of upload histories for three tourists in exemplar group. Top 3 POI categories shown for each tourist.

	Tourist A	Tourist B	Tourist C
Total # Photos	250	236	679
# Photos by City	Paris: 66 Berlin: 60 Duesseldorf: 49 Munich: 35 San Francisco: 25 Frankfurt: 15	Brussels: 41 Duesseldorf: 43 Florence: 15 Lisbon: 33 Paris: 85 Rome: 19	Berlin: 70 Dresden: 64 Duesseldorf: 42 Hamburg: 77 Helsinki: 21 Munich: 241 St. Petersburg: 85 Vienna: 79
# Photos by POI Category	Heritage: 23 Beer Garden: 14 Wood: 12 ...	Wood: 51 Heritage: 36 Park: 20 ...	City Gate: 67 Ferry Terminal: 55 Zoo: 46 ...

Table 5.2: Top 5 cities in rankings for the group generated by each model, with each aggregation method.

$\mathbb{R}_{TTP}^G$	$\mathbb{R}_{CCP}^G$	$\mathbb{R}_{TCP}^G$
1. Cologne 2. Moscow 3. Stuttgart 4. Salzburg 5. Venice	1. Salzburg 2. Krakow 3. Stuttgart 4. Barcelona 5. Antwerp	1. Cologne 2. Hamburg 3. Birmingham 4. Dresden 5. Houston
$\mathbb{R}_{TTR}^G$	$\mathbb{R}_{CCR}^G$	$\mathbb{R}_{TCR}^G$
1. Cologne 2. Amsterdam 3. London 4. Stuttgart 5. Salzburg	1. Salzburg 2. Krakow 3. Stuttgart 4. Barcelona 5. Antwerp	1. Cologne 2. Moscow 3. Houston 4. Dublin 5. Manchester

promising, this discussion is far from a rigorous performance evaluation, which is reserved for subsequent chapters.

### 5.2.5 POI Recommendation

As mentioned in section 3.2.9 I chose to implement the POI recommendation model for a single tourist  $t$ , due to both time constraints and difficulties with adapting my evaluation method to groups. As with the city recommender, I implemented the equations in section 3.2 in a single Python script, whose operation is summarised in algorithm 2. It returns a single tuple representing a ranking of all POIs in the target city  $c$  in order of recommendation score according to the model.

To further illustrate the operation of the recommender algorithm, the following is an example of an arbitrary tourist from the dataset who has visited London on two separate occasions, and a ranking of POIs to visit next (i.e. in continuation of the second of these trips) output by the recommender system. The relevant aspects of the tourist’s *Flickr* photo upload history are shown in table 5.3.

Table 5.4 contains the top 10 POIs in the ranking created for this tourist by the system, assuming a visit at 17:30 on the 6th of April 2012, around an hour after the most recent visit in their history. Since I have not yet described the method used for training the machine learning-based scoring techniques (this is presented in section 6.2), the scores were computed by the simple summing method *Sum*. When calculating the features themselves, the following hyperparameter values were used:

$$\alpha = 100 \quad \beta = 2000 \quad \gamma = 100 \quad \zeta = 0.1 \quad \kappa = 24$$

Without going into too much detail about the reasonableness of these results: most of the POIs shown are popular tourist attractions in close proximity to the Tate Modern

---

**Algorithm 2:** POI recommendation for a single tourist

---

**inputs** : the JSON-format dataset file; a single `User` ID value  $t$ , `City` value  $c$ , and timestamp  $now$  to define the recommendation scenario

**outputs:** a POI ranking from the hybrid model

```

// Assemble data structures
 $\mathbb{T}$  ← set of all User ID values in the dataset;
 $\mathbb{C}$  ← tuple of unique City values, alphabetical order;
 $\mathbb{X}$  ← tuple of unique POI Category values with  $\geq 500$  visits, alphabetical order;
for  $t' \in \mathbb{T}$  do
  for  $c' \in \mathbb{C}$  do
     $\mathbb{H}^{t',c'} \leftarrow$  tuple equal to the corresponding Visits list
 $\mathbb{P}^c \leftarrow$  set of POI values that appear within Visits lists for  $c$ ;
 $\mathbf{p}^t \leftarrow$  vector derived from per-category photo counts for  $t$  (equation 5.2);
for  $x \in \mathbb{X}$  do
   $\text{chHist}_{\text{hour}(now)}(x) \leftarrow x$ 's visit count for hour( $now$ ) (equation 3.21);
   $\text{cmHist}_{\text{month}(now)}(x) \leftarrow x$ 's visit count for month( $now$ ) (equation 3.25);

// Compute recommendation score for each POI in the city
for  $p \in \mathbb{P}^c$  do
  // Overall popularity feature
   $fPop(p) \leftarrow$  function of unique visitors to  $p$  (equation 3.17);

  // Preference-aware feature
   $fCat_{\mathbf{p}^t}(p) \leftarrow$  corresponding element of  $\mathbf{p}^t$  (equation 3.18);

  // Context-aware features
   $fProx_{\mathbb{H}^{t,c}}(p) \leftarrow$  function of distance to  $\text{loc}(p)$  (equation 3.20);
   $\text{phHist}_{\text{hour}(now)}(p) \leftarrow p$ 's visit count for hour( $now$ ) (equation 3.23);
   $fTime_{\text{hour}(now)}(p) \leftarrow$  function of time histogram values (equation 3.24);
   $\text{pmHist}_{\text{month}(now)}(p) \leftarrow p$ 's visit count for month( $now$ ) (equation 3.26);
   $fDate_{\text{month}(now)}(p) \leftarrow$  function of date histogram values (equation 3.27);

  // History-aware features
   $fHist_{\mathbb{H}^{t,c}}(p) \leftarrow 0$ ;
  for  $p' \in \{\text{poi}(v) : v \in \mathbb{H}^{t,c}\}$  do
     $\text{coinc}(p, p') \leftarrow$  separation-weighted coincidence count (equation 3.28);
     $fHist_{\mathbb{H}^{t,c}}(p) \leftarrow fHist_{\mathbb{H}^{t,c}}(p) + \text{coinc}(p, p')$  weighted by time elapsed since most recent visit (equation 3.29)

  // Scoring and ranking
   $\bar{\mathbf{v}}^{t,p} \leftarrow$  z-normalised vector of all six features (equation 3.32);
   $S^t(p) \leftarrow$  function of  $\bar{\mathbf{v}}^{t,p}$ ; sum or machine learning model (section 3.2.8)
 $\mathbb{R}^t \leftarrow \mathbb{P}^c$  ordered by  $S^t$ ;
return  $\mathbb{R}^t$ 

```

---

Table 5.3: Summary of upload history for exemplar tourist. Top 3 POI categories shown.

Previous Visits in London	POI Name	Date	Start Time	End Time	# Photos
	Natural History Museum	2009-11-05	14:42:30	16:36:46	5
	Old Street Records	2009-11-07	19:11:07	19:11:07	1
	South Kensington Farmers Market	2009-11-08	10:42:52	11:18:20	3
	Natural History Museum	2012-04-05	12:28:23	12:28:23	1
	Columbia Road Flower Market	2012-04-05	14:33:33	14:50:53	2
	Tate Modern	2012-04-06	14:17:01	16:26:56	6
Other Cities Visited	Berlin (9 visits)	Stockholm (4 visits)	Paris (2 visits)		
# Photos by POI Category	Attraction: 24	Arts Centre: 13	Museum: 7	...	

Table 5.4: Top 10 POIs in ranking for the tourist generated the model, with *Sum* scoring method. Features z-normalised using statistics from a large body of other recommendation scenarios. All values given to 2 significant figures.

POI Name	Z-normalised Feature Values						Score
	fPop	fCat	fProx	fTime	fDate	fHist	
1. Millennium Bridge	2.3	-0.068	1.9	1.2	0.064	0.46	5.9
2. Bank of England	0.69	-0.068	1.1	1.3	2.0	-0.25	4.8
3. St Paul's Churchyard	2.5	-0.068	1.4	0.28	0.77	-0.093	4.7
4. South Bank Book Market	2.3	0.00060	1.0	1.5	0.036	-0.27	4.6
5. Shakespeare's Globe	0.93	-0.068	2.0	0.41	0.10	0.75	4.1
6. OXO Tower Wharf	0.59	-0.068	1.5	1.1	0.25	0.50	3.9
7. Tate Modern Café	-0.97	-0.068	2.1	-0.048	1.5	1.3	3.8
8. George Inn	-0.97	-0.068	1.4	-0.010	3.6	-0.23	3.6
9. Tower Bridge	2.5	-0.019	0.6	0.91	0.11	-0.48	3.6
10. Postman's Park	-1.07	-0.050	1.1	0.48	3.55	-0.44	3.5

(the tourist’s current location), that seem suitable places to visit in an early springtime evening – especially the two bridges and cathedral courtyard, which could provide attractive ‘golden hour’ views of some of London’s most famous landmarks. As with the city recommendation problem, a quantitative performance evaluation is contained in the following chapters.

# Chapter 6

## Evaluation and Optimisation Methods

The implementation work described thus far yielded systems that could generate rankings of cities and POIs to visit for tourists in the travel histories dataset. However, in the absence of external information there is no way to evaluate the quality of these rankings, and in the case of the POI recommender, no source of supervision to train the machine learning models used to map features into scores. This chapter describes how I address these issues. My general approach is one of cross-validation: hiding parts of the dataset from the recommender models, and recasting the recommendation problem as one of predicting the missing data.

### 6.1 Evaluation Methods

#### 6.1.1 City Recommendation

##### Single-Tourist Case

As per the problem definition in section 3.1.2, the aim of single-tourist city recommendation is to produce a city ranking  $\mathbb{R}^t$  that maximises  $\text{nDCG}(\mathbb{R}^t)$ . However, since the travel histories dataset contains no data about unvisited cities, the potential enjoyment function  $\text{enj}^t$  cannot be obtained directly. For evaluation purposes, my solution is to remove one city (denoted the *forgotten city*  $c^*$ ) entirely from the target tourist’s history, effectively moving it from the visited set  $\mathbb{V}^t$  to the unvisited set  $\mathbb{U}^t$  and deleting any associated POI visits. I then define  $\text{enj}^t$  as follows:

$$\forall c \in \mathbb{U}^t \quad \text{enj}^t(c) = \begin{cases} 1 & \text{if } c = c^*, \\ 0 & \text{otherwise} \end{cases} \quad (6.1)$$

This extremely simple definition in turn reduces the nDCG equation to

$$\text{nDCG}(\mathbb{R}^t) = \frac{1}{\log_2(r^* + 1)} \quad (6.2)$$

where  $r^*$  is the position of  $c^*$  in the ranking (1 being the top position). Maximising this metric is exactly equivalent to moving  $c^*$  to the top of the ranking. In other words, an ideal solution to the city recommendation problem with the travel histories dataset would produce a ranking that has the forgotten city in first place, thereby predicting the true travel decision of the target tourist.

Given that the rank of the forgotten city is more intuitively comprehensible than the nDCG, and the two are functionally equivalent in this context, the results in the next chapter use  $r^*$  as their primary performance metric.

### Multi-Tourist Group Case

For the group recommendation problem, I constrain the set of *valid* tourist groups to those in which all members have at least one city in common. Formally, a group  $\mathbb{G}$  is valid if

$$\mathbb{I}^{\mathbb{G}} = \bigcap_{t \in \mathbb{G}} \mathbb{V}^t \neq \emptyset \quad (6.3)$$

The forgotten city  $c^*$  is chosen at random from  $\mathbb{I}^{\mathbb{G}}$  and removed from the histories of *all* tourists in the group. Rankings can then be evaluated in exactly the same way as in the single-tourist cases: the higher the ranking of the forgotten city, the better.

### Baseline Rankings

Alongside a cross-comparison of the three recommendation models and two group aggregation methods, I baseline performance against the most naïve recommender possible that makes use of the dataset: one that simply ranks the cities in  $\mathbb{U}^t$  (or  $\mathbb{U}^{\mathbb{G}}$ ) by the number of photos uploaded in each. To the extent that any given model consistently outperforms this naïve approach, it must successfully be harnessing information that is specific to the target tourist or group. As a final point of comparison, I also generate entirely random rankings, against which all other methods can be compared.

#### 6.1.2 POI Recommendation

As with the city recommendation problem, the aim of POI recommendation is to produce a POI ranking  $\mathbb{R}^t$  that maximises  $\text{nDCG}(\mathbb{R}^t)$ , but the potential enjoyment function  $\text{enj}^{t,\text{now}}$  cannot be obtained directly. My evaluation solution in this case is to *cut* the target tourist  $t$ 's history in one of their visited cities  $c$  at a specified location, and delete

all visits after that point. The tourist's history in all other cities is left unaltered<sup>1</sup>. The choice of city and cut location  $i$  (which specifies the index of the immediately preceding visit) must satisfy the following criteria:

- The city has at least 10 POIs which have each received at least 20 unique visitors:  $|\bar{\mathbb{P}}^c| \geq 10$  where  $|\bar{\mathbb{P}}^c| = \{p \in \mathbb{P}^c : \text{vis}(p) \geq 20\}$ .
- The tourist has at least two visits in  $c$ , and the cut is after the first visit but before the last:  $1 \leq i < q^{t,c}$  where  $q^{t,c} = |\mathbb{H}^{t,c}| \geq 2$ .
- After cutting, the number of remaining visits in the tourist's history across all cities is at least 20:  $\left(i + \sum_{c' \in \mathbb{V}^t: c' \neq c} |\mathbb{H}^{t,c'}|\right) \geq 20$ .
- The two visits which lie either side of the cut are separated in time by at most 8 hours:  $\text{startTime}(\mathbb{H}_{i+1}^{t,c}) - \text{endTime}(\mathbb{H}_i^{t,c}) \leq 8h$ .
- The POI of the visit immediately after the cut has itself received at least 20 unique visitors:  $\text{poi}(\mathbb{H}_{i+1}^{t,c}) \in \bar{\mathbb{P}}^c$ .

A recommendation scenario is fully specified by the the tourist  $t$ , city  $c$  and valid cut location  $i$ . Rather than considering all POIs in the target city for recommendation, I limit the options to those with at least 20 unique visitors (i.e. the elements of  $\bar{\mathbb{P}}^c$ ). The POI associated with the visit immediately after the cut is denoted the *forgotten POI*  $p^*$ . From this point, I define  $\text{enj}^{t,\text{now}}$  as

$$\forall p \in \bar{\mathbb{P}}^c \quad \text{enj}^{t,\text{now}}(p) = \begin{cases} 1 & \text{if } p = p^*, \\ 0 & \text{otherwise} \end{cases} \quad (6.4)$$

which, as in the city recommendation problem, reduces the nDCG equation to

$$\text{nDCG}(\mathbb{R}^t) = \frac{1}{\log_2(r^* + 1)} \quad (6.5)$$

where  $r^*$  is the position of  $p^*$  in the ranking (1 being the top position). Maximising this metric is exactly equivalent to moving  $p^*$  to the top of the ranking. In other words, an ideal solution to the POI recommendation problem with the travel histories dataset would produce a ranking that has the forgotten POI in first place, thereby predicting the true travel decision of the target tourist. Following the rationale expounded for the city recommendation problem, I again use  $r^*$  as the basis of my performance metric. However, comparison of performance between cities in terms of  $r^*$  alone could be misleading, since they are very likely to have different numbers of POIs. For this reason, I define a general

---

<sup>1</sup>This is true even if the tourist has visited another city after the date at which the cut is made. This is somewhat artificial as future travel information would not be available to a recommender system in a real-world application, but given that it is available in the dataset, I can see no important reason why it should not be used.

performance metric  $Q$  as the complement of  $r^*$  as a *fraction* of the number of candidate POIs within the target city  $c$ :

$$Q = 1 - (r^*/|\mathbb{R}^c|) \quad (6.6)$$

### Baseline Rankings

Alongside a cross-comparison of the three methods for mapping features to scores, I evaluate ranking performance relative to a number of simple baselines. In [41], the authors use three such baselines when evaluating their POI recommender:

- Ranking by overall popularity;
- Ranking by proximity to current location;
- Random ranking.

I partially adopt this approach, but acknowledge that my definitions of the features fPop and fProx mean that they are monotonic in the first two of these measures respectively, so ranking by the features themselves produces identical results. Further, I bring in the other four individual features (fCat, fTime, fDate, fHist) as additional baselines, bringing the total to seven (six features plus random). This approach is particularly meaningful, since it allows individual evaluation of the efficacy of the features, and of the methods used to combine them.

## 6.2 Optimisation of Machine Learning Models for POI Scoring

The definition of a quantitative performance metric ( $r^*$ ) provided the means for training the two machine learning models for mapping POI features into recommendation scores (*Lin* and *NN*). This part of the project constituted a *learning-to-rank* problem, albeit one with weak supervision. In a fully-specified learning-to-rank problem, each item has a unique relevance score, and the objective of training is to produce rankings that are ordered by relevance as far as possible (as quantified by metrics such as nDCG). In this context, the equivalent of a relevance score is  $\text{enj}^t$ , which is zero for all POIs other than  $p^*$ , meaning the notion of a ‘good’ ranking is drastically under-constrained. For this reason, rather than seeking to guide the learning process towards a fixed, idealised end state, as in traditional supervised learning, I framed it procedurally as the act of *iteratively moving  $p^*$  higher in the rankings* across a large set of recommendation scenarios. For both *Lin* and *NN*, the training process proceeded as follows:

1. Sweep through the entire travel histories dataset and identify all recommendation scenarios that meet the criteria given in section 6.1.2. For each (tourist, city) pair, keep only the last qualifying scenario (i.e. the one with the highest  $i$ ).
2. For each scenario  $(t, c, i)$ , assemble feature vectors for all POIs in  $\bar{\mathbb{P}}^c$ . Store these alongside the name of the forgotten POI  $p^*$ .
3. Randomly partition the scenarios into training, validation and test sets with a 60:20:20 split. Perform z-normalisation on each set according to the method described in section 3.2.8.
4. Randomly initialise the weights of the learning model.
5. For one scenario  $(t, c, i)$  in the training set, iterate through every POI  $p$  and feed the corresponding feature vector  $\bar{\mathbf{v}}^{t,p}$  into the machine learning model. Use the output value as the POI's score  $S^t(p)$ .
6. Assemble the ranking  $\mathbb{R}^t$  by ordering the POIs by score. Determine  $r^*$ , the rank of the forgotten POI.
7. If  $r^* = 1$ , skip to step 10. Otherwise, define the *error*  $e$  for this scenario as the difference between the score assigned to  $p^*$  and the score of the POI ranked  $\omega$  places higher in the ranking:

$$e = S^t(\mathbb{R}_{r^*-\omega}^t) - S^t(p^*) \quad (6.7)$$

If  $\omega = 1$ , the error is defined with respect to the POI ranked immediately above  $p^*$ . If  $\omega = r^* - 1$ , it is defined relative to the top-ranked POI. A third option is for  $\omega$  to be sampled randomly from  $\{1 .. r^* - 1\}$  for each scenario. The influence of this hyperparameter is assessed in the following chapter.

8. Use  $e$  to inform *two* weight updates by gradient descent. In the first, use the feature vector for  $p^*$  and  $-e$  as the error. In the second, use the vector for  $p' = \mathbb{R}_{r^*-\omega}^t$  and  $e$  as the error. For the neural network model  $NN$ , use the backpropagation algorithm to determine the required weight updates for neurons in the hidden layer(s). The intended effect of these weight updates is to move  $p^*$  higher in the ranking, and the POI currently placed in position  $r^* - \omega$  lower in the ranking.
9. Perform two secondary weight updates with the aim of spreading the recommendation scores across the range  $[0, 1]$ . In the first, define the error as the difference between the score of the top-ranked POI and 1 ( $e = S^t(\mathbb{R}_1^t) - 1$ ) and use the vector as the input. In the second, do the same with the bottom-ranked POI, and the difference between its score and 0. The purpose of this step is to non-aggressively incentivise the model to produce scores of reasonable and well-differentiated magnitudes, which aids the stability of the learning process.

10. Complete steps 5-6 for one scenario in the validation set.
11. Repeat steps 5-10 for all other scenarios in the training and validation sets. Once the end of either set has been reached, randomly shuffle it and begin again from the start.
12. Throughout training, maintain a 500-sample moving average of  $Q$  as defined in equation 6.6. Denote this value  $Q_{500}$ . Evaluate the change in  $Q_{500}$  every 50 samples. Stop training if it becomes negative, and revert the model's weights to their state as of the previous evaluation.

An important influence on training performance was the learning rate, commonly denoted  $\eta$ . For the linear model *Lin*, I found that for the primary weight updates (aimed at moving  $p^*$  up the ranking),  $\eta_{prim} = 0.0001$  yielded fast and stable learning. For the secondary updates (aimed at spreading the scores across  $[0, 1]$ ), I used  $\eta_{sec} = 0.000005$ . As a more complex models with more parameters, neural networks commonly demand more aggressive learning rates, and also tend to suffer from a slower start to learning. For this reason, I used a dynamic learning rate that varied as a function of  $Q_{500}$ . Strong performance was obtained by setting the primary learning rate  $\eta_{prim}$  to

$$\eta_{prim} = 1.0 + 0.999 \cdot (1 - 2^{(Q_{500})^{1/4}}) \quad (6.8)$$

and secondary learning rate  $\eta_{sec}$  to  $0.02 \times \eta_{prim}$ .

The results of applying this training process to both the linear model *Lin* and neural network model *NN* (with various hidden layer topologies) are presented in the following chapter. Their post-training performance is compared to the simple summation model *Sum*, as well as the single-feature and random baselines introduced in section 6.1.2.

# Chapter 7

## Results and Discussion

This chapter presents the results of applying the city- and POI-level recommender models to the two recommendation problems with the travel histories dataset. Where I have presented multiple model types or variants, these are systematically compared in terms of the performance metrics outlined in the previous chapter. Further analysis seeks to identify the drivers of strong and weak recommendation performance, and highlight avenues for further improvement.

### 7.1 City Recommendation

#### 7.1.1 Single-Tourist Results

When assessing the performance of the three city recommendation models ( $TT$ ,  $CC$  and  $TC$ ), an important decision must be made regarding the generation of test cases. The two options are:

- Randomly sample *tourists* from the dataset, then select a city to forget for each;
- Randomly sample *cities* to forget from the list of 200, then select a tourist who has visited each one to use as the recommendation target.

These approaches produce test sets with markedly different statistics. The former mirrors the environment of a real-world recommender system, but its set of forgotten cities is inevitably biased towards major destinations such as London, New York and Paris, simply because many tourists have been to these places. The latter avoids this issue by sampling cities with uniform probability. While this approach is more artificial, it more thoroughly tests the models' ability to recommend less popular forgotten cities, and is arguably a more demanding assessment of personalised recommendation quality. I initially explored both approaches, generating two 5,000-sample test sets which I denote *tourist-first* and *city-first* respectively.

Figure 7.1 contains box plots summarising the distribution of  $r^*$  (position of forgotten city in ranking) across the test set for all three models<sup>1</sup>, as well as for the naïve and random baseline methods described in section 6.1.1, on both test sets. The difference between the two sampling methods is visually evident: all three models perform better on the tourist-first test set. The efficacy of the naïve ranking method differs most, varying from better than both  $CC$  and  $TC$  with tourist-first sampling to scarcely barely better than random with city-first sampling. This effectively demonstrates how recommending based on population-wide statistics can give good performance when the popularity of options is highly unbalanced, despite being entirely ignorant of individual preferences.

There are valid reasons why I might have chosen either test set for further analysis. However, the fundamental aim of this project is to build recommender systems capable of inferring individual preferences and generating personalised recommendations that are appropriate, *regardless how atypical those preferences are*. I believe a city-first test set better captures the spirit of this objective, since it is not biased towards the most common tourist destinations, and cannot be ‘gamed’ by a non-personalised method such as the naïve baseline. For this reason, all subsequent analysis in this section uses test sets generated using city-first sampling.

Focusing now on the box plots for the city-first test set in figure 7.1: while all three models outperform both baselines in the aggregate, the model based on tourist-tourist similarity ( $TT$ ) is markedly stronger. For this model, the  $r^*$  quartiles lie at ranks 2, 8 and 23. Differentiating between the two other models is more difficult:  $CC$  has its quartiles at 24, 61 and 110, compared with 21, 60 and 113 for  $TC$ . By one popular informal test: the notches of the two box plots overlap, so the difference between the medians is not statistically significant [52].

A more direct comparison to the naïve baseline is shown in figures 7.2 and 7.3. For each scatter point in the former, the abscissa represents the rank of the forgotten city output by one of the three recommender models on a particular test case, and the ordinate represents the rank according to the baseline. Points that lie to the left of the diagonal lines represent cases where the models outperform the baseline. The latter figure represents the same data as probability distributions over the *difference* in  $r^*$  between each model and the baseline on a case-by-case basis. Here, negative values indicate performance that exceeds the baseline. The dominance of  $TT$  is further evidenced here: out of the 5,000 test cases, this model ranks the forgotten city at least as high as the baseline 4386 times (87.7%), and on average ranks it 69 places higher. Interestingly, figure 7.2 demonstrates that the model’s performance remains constant all the way down to the least popular of the 200 cities (i.e. those with the lowest baseline ranks). This suggests that the dataset is large enough to provide informative neighbourhoods even for tourists with relatively unusual travel histories.

---

<sup>1</sup>For the  $TT$  model, a neighbourhood size of  $n = 50$  was used.

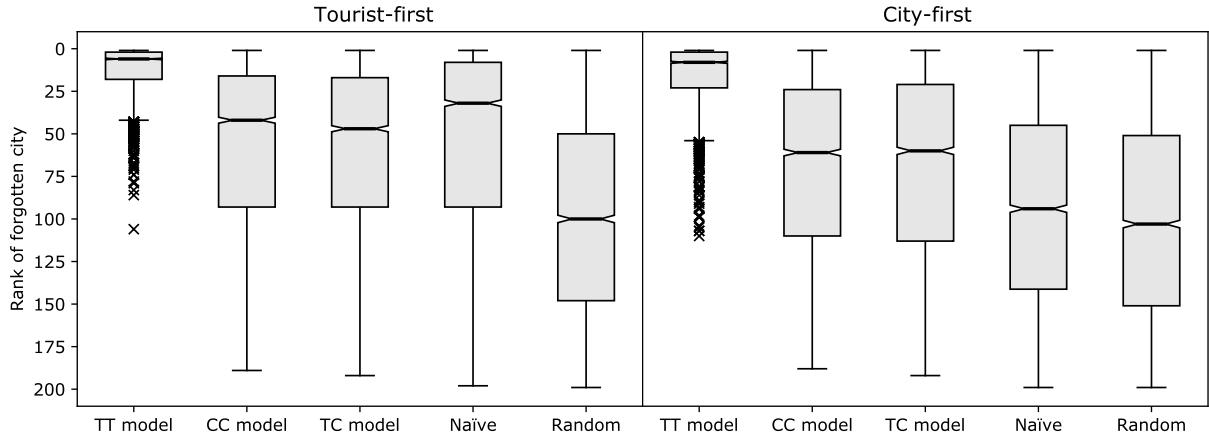


Figure 7.1: Box plots showing the distribution of  $r^*$  values for the three city recommender models and two baseline methods across 5,000 test cases, with both tourist-first and city-first sampling. In each plot, the horizontal line connecting the notches indicates the median, the top and bottom of the box indicate the 1st and 3rd quartiles, and the whiskers indicate the farthest data point within  $1.5 \times$  the interquartile range from the median. Crosses represent outliers.

*CC* outperforms the baseline in 3100 of the test cases (62.0%), with an average advantage of 24 places, and for *TC* the values are 3040 (60.8%) and 24 places. These two models are again almost inseparable in terms of performance. They can both clearly capture some discernible signal in the data to use for personalised recommendation, but the results are significantly less impressive than *TT*. It is a natural next step to investigate what might be causing the lack of consistency in performance.

Figures 7.4 and 7.5 attempt to uncover possible causal factors in the  $r^*$  values for the *CC* and *TC* models respectively. In figure 7.4, which concerns *CC*, the bar charts categorise the 5,000-sample test set according to the number of labelled visits made both in the forgotten city, and in the other cities that the target tourist has visited. The former appears to correlate positively with performance: the more visits have been made in a forgotten city, the more likely it is to be seen high up the ranking. This may be because more labelled visits enable better estimation of the city's POI category importance vector  $\mathbf{i}^{c^*}$ , thus a greater ability to compute meaningful city-city similarity values. No such relationship seems to exist with respect to the tourist's other visited cities, and if anything there is a slight negative trend. While too much should not be read into this counter-intuitive result, a possible explanation is that many less commonly-visited cities are known for just one or two famous POIs (e.g. the cathedral and Roman baths of Bath), so visitors there are likely to hold a specific interest in those POI categories in particular.

In figure 7.5, the performance of *TC* is presented as a function of labelled visits in the forgotten city, and also of labelled visits made by the target tourist themselves.

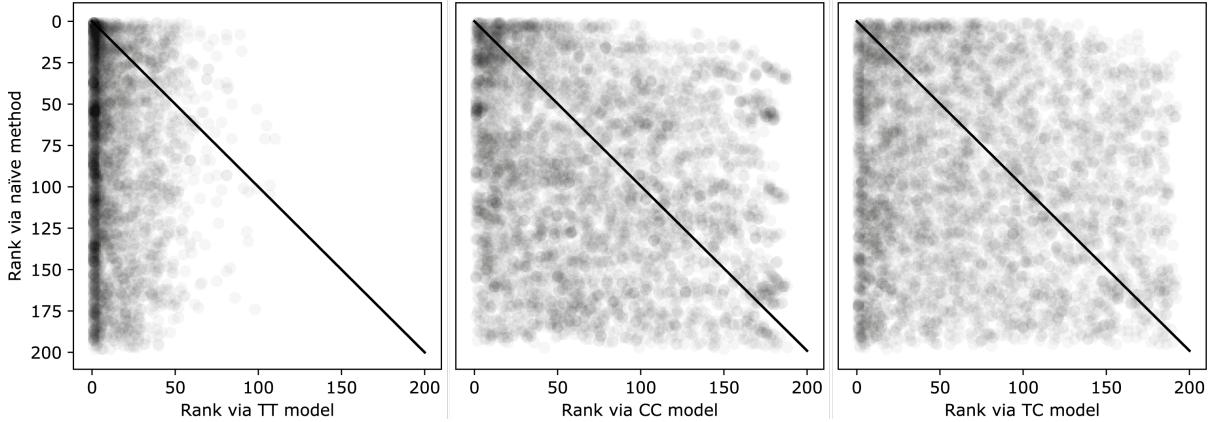


Figure 7.2: Scatter plots relating  $r^*$  from each model to the value from the naïve baseline method for 5,000 test cases (city-first sampling). The diagonal lines indicate thresholds of equal performance; any point to the left of a line is a case where the model ranks the forgotten city higher than the baseline.

The former again has a positive correlation, reinforcing the previous point about better estimation of the  $\mathbf{i}^{c^*}$  vector. The data around the tourist's visit count is inconclusive for middling values, but the trend again seems to be positive, and tourists with 50 to 200 labelled visits should expect to see the forgotten city appear around 10 places higher than those with fewer than 5 visits. None of the charts in figures 7.4 and 7.5 reveal trends strong enough to be incontrovertible. This in itself suggests that the *CC* and *TC* models are fundamentally weaker across-the-board than the *TT* model with respect to my chosen performance metric.

Shifting attention to the high-performing *TT* model, I investigated the effect of two factors on recommendation performance: the number of cities that the target tourist has already visited, and the neighbourhood size parameter  $n$ . Figure 7.6 plots the former against  $r^*$  for the city-first test set, with a trend line fitted using least squares regression. While it would be ill-advised to draw a definite conclusion from this graph since points for high city counts are sparse and the general variance is high ( $R^2 = 0.078$ ), it seems that as the number of cities a tourist has visited increases (especially beyond 10), the recommendation quality *decreases*, or at the very least becomes less consistent. This runs somewhat counter to intuition: one might expect that access to more information about the subject of a recommender system should improve performance. One plausible cause of this is that exceptionally well-travelled tourists are less likely to adhere to a consistent and predictable pattern than their less prolific (more selective) counterparts, perhaps because their movements are dictated by work or hobbies rather than the destinations themselves.

Figure 7.7 contains box plots showing the distribution of  $r^*$  for the *TT* model with a variety of values for the neighbourhood size  $n$ . For each  $n$  value, 500 city-first test

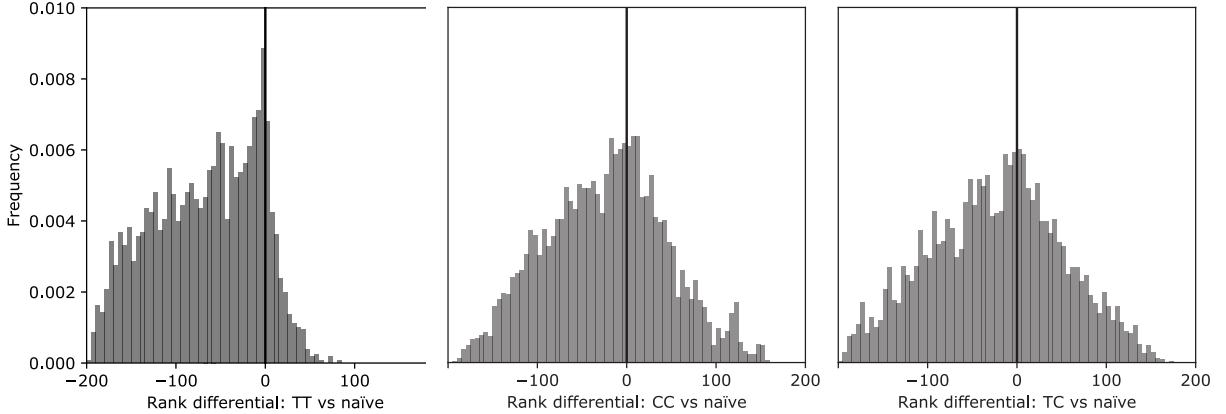


Figure 7.3: Probability distributions over the difference between  $r^*$  from each model and the value from the naïve baseline method for 5,000 test cases (city-first sampling). The vertical lines indicate thresholds of equal performance; probability mass to the left of a line corresponds to the model ranking the forgotten city higher than the baseline.

samples were used. The tightening of the box towards the upper end of the ranking as  $n$  is decreased indicates that smaller, more exclusive neighbourhoods generally produce rankings that place the forgotten city higher (i.e. small neighbourhoods give better-tailored recommendations). Crucially, however, for  $n$  values of 25 and below, cases appear in which the forgotten city is placed at the bottom of the ranking. This happens because in any small neighbourhood, there is a non-negligible chance that *none* of the constituent tourists have visited a given city and it is given a score of zero<sup>2</sup>. For  $n = 25$ , 12 such cases occur. For  $n = 10$ , this increases to 46 and for  $n = 5$ , the number is 93. There is thus a tradeoff between the *specificity* of the neighbourhood (small  $n$ ), which increases the likelihood of the forgotten city appearing at the very top of the ranking, and its *completeness* (large  $n$ ), which minimises the risk of it not appearing at all. In practical applications, it would be valid to experiment with small neighbourhoods, which may in fact lead to the greatest user satisfaction. However, according to the measure of recommendation performance used in this project – reliably-high ranking of the forgotten city – a neighbourhood of  $n = 50$  produces the strongest results of the options tried.

### 7.1.2 Group Results

Having analysed the performance of the city recommender models in the single-tourist case, I move on to the problem of recommending to multi-tourist groups. In addition to the three model types, I must evaluate the two aggregation methods (*AoS* and *AoP*). As a further complication, two more factors come into play that could influence performance:

<sup>2</sup>Several other cities in the set of 200 also do not appear in the visit histories of any neighbour, so in reality the system orders these randomly at the bottom of the ranking. This means the forgotten city does not always rank exactly last despite having a score of zero. In figure 7.7 I show only a single marker at rank 200 because this constitutes the worst-case outcome of the phenomenon of interest.

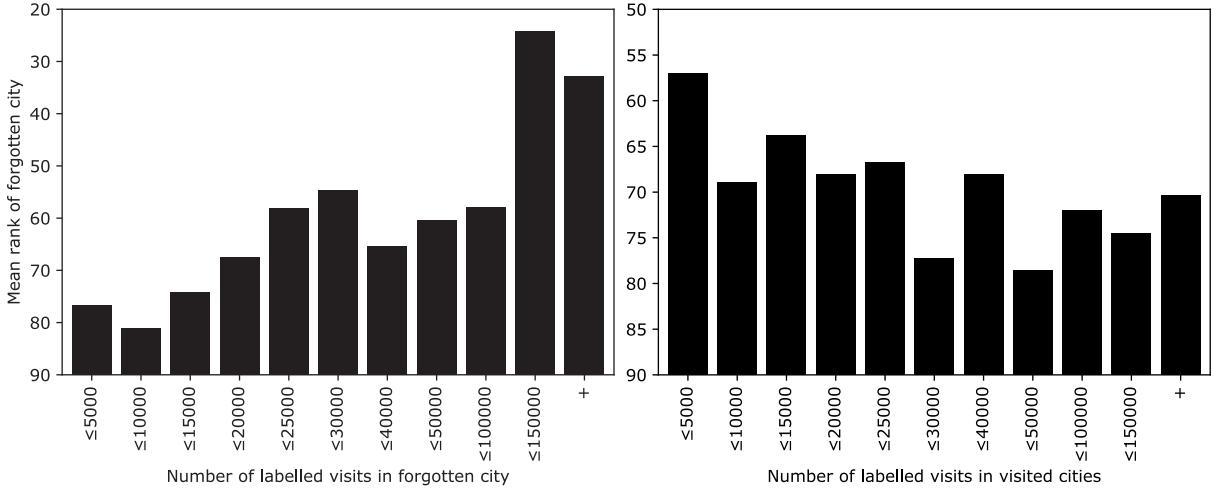


Figure 7.4: Bar charts relating the performance of the *CC* model to the number of labelled visits in the forgotten city (left) and in the tourist's other visited cities (right). The + bars include cases where the number of visits is greater than the maximum value in the penultimate bar.

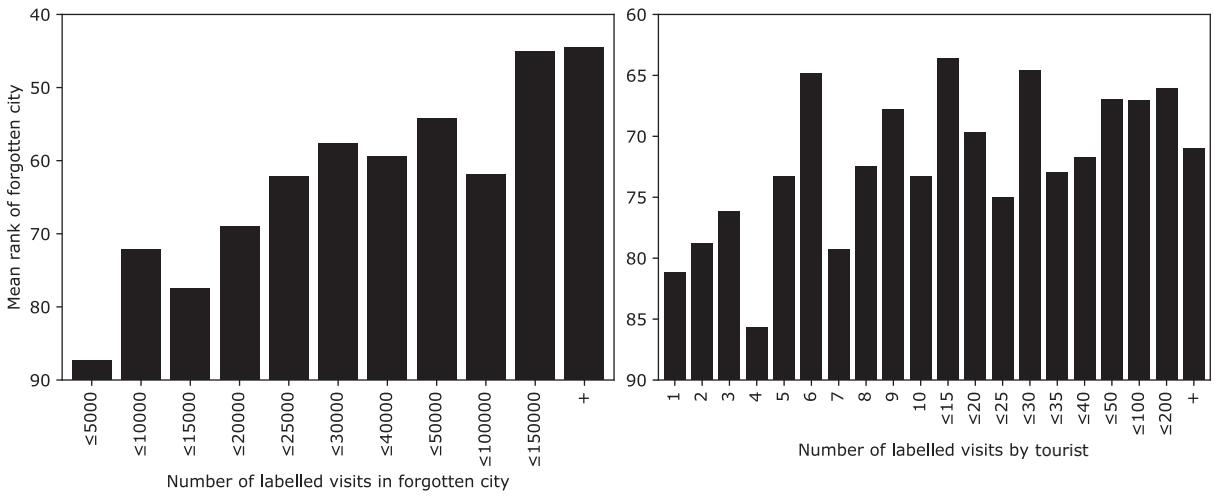


Figure 7.5: Bar charts relating the performance of the *CC* model to the number of labelled visits in the forgotten city (left) and by the tourist themselves (right). The + bars include cases where the number of visits is greater than the maximum value in the penultimate bar.

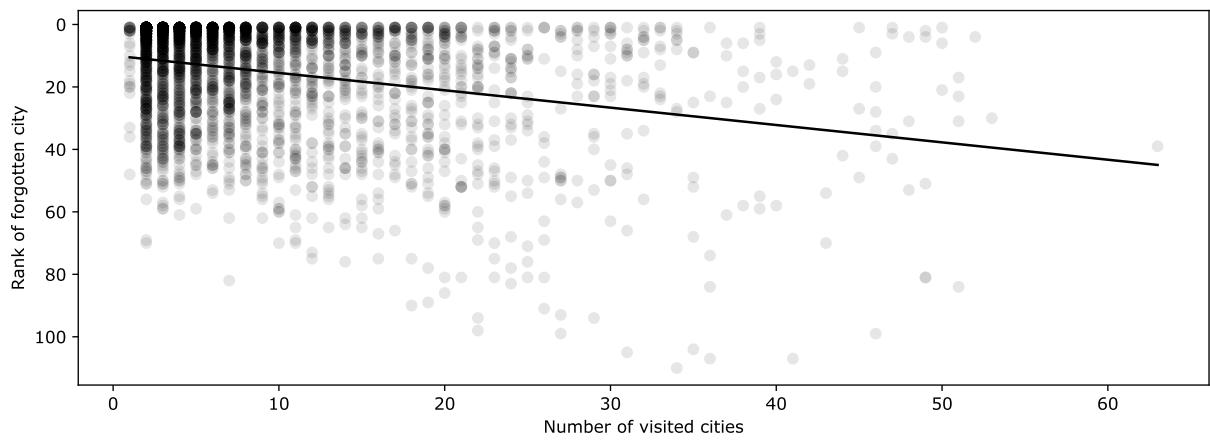


Figure 7.6: Scatter plot relating the number of cities already visited by the target tourist to the  $r^*$  value from the  $TT$  model for each of the 5,000 test cases (city-first sampling). Trend line fitted using least squares regression.

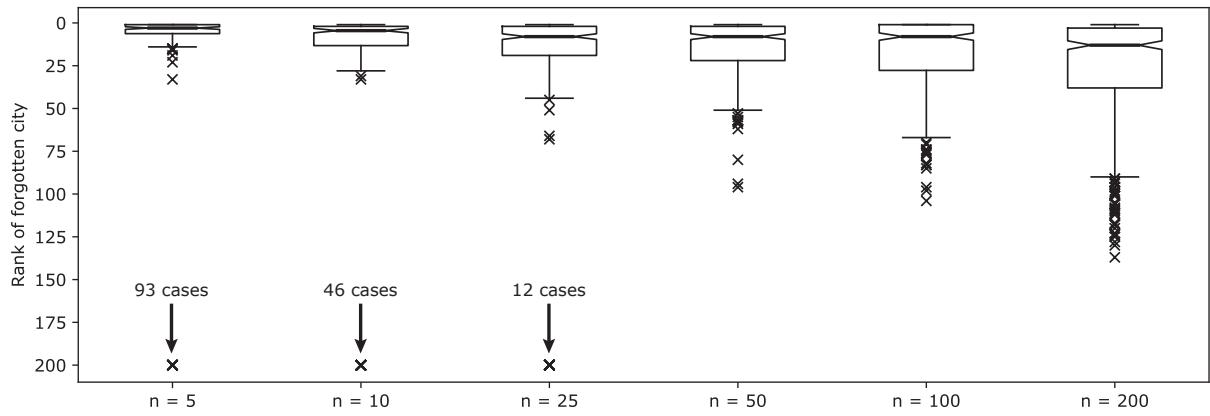


Figure 7.7: Box plots showing the distribution of  $r^*$  values for  $TT$  city recommender model with various neighbourhood sizes across 500 test cases (city-first sampling).

the choice of aggregation operation (sum or max) for either scores ( $AoS$ ) or preference vectors ( $AoP$ ) and the size  $g$  of the tourist group itself (I consider values of  $g = 2, 3, 5$  and 10). This produces a large number of permutations for cross-comparison, 48 in total, which are summarised in table 7.1.

Table 7.1: Schema of group/model permutations assessed for the group city recommendation problem. There are 16 for each of the three models, giving a total of 48.

		Group Size ( $g$ )			
		2	3	5	10
Model	$TT$	$AoS$ , sum	...	...	...
		$AoS$ , max	...	...	...
		$AoP$ , sum	...	...	...
		$AoP$ , max	...	...	...
	$CC$	...			
	$TC$	...			

For each value of  $g$ , I evaluate using 500-case test sets (city-first sampling). The complete results are shown as box plots in figures 7.8, 7.9 and 7.10. Results for  $g = 1$  (i.e. the single-tourist case analysed previously) are also included for comparison. For the  $TT$  model, a neighbourhood size of  $n = 50$  is used throughout. Since these figures summarise the effects of many factors on recommendation performance, there is no obvious systematic way to discuss them. The following is a bullet-pointed outline of my key observations:

- Across all three models and aggregation methods, there is a trend towards improved recommendation performance as the group size  $g$  increases. The rate of improvement is more pronounced with the  $AoS$  aggregation method (i.e. compute recommendation scores for each tourist then combine by sum- or max- operation).
- Across all group sizes and aggregation methods, the  $TT$  model remains the strongest by a large margin (note the rescaled vertical axis in figure 7.8). With this model, there is no clear distinction in performance between  $AoS$  and  $AoP$ , though for larger group sizes it appears that both benefit from sum-based aggregation of scores/preferences. The single best results are attained using sum-based  $AoS$  with  $g = 10$ ; here the  $r^*$  quartiles are at ranks 1, 1 and 6, which implies that in fully half of the 500 test cases, the model ranks the group's forgotten city in first place.
- Despite impressive aggregate statistics, there remain a significant number of outliers in the results for the  $TT$  model. It is likely that these are simply unavoidable on this dataset: the travel decisions of *Flickr* users are not fully predictable given only their history of photographs taken in other cities.

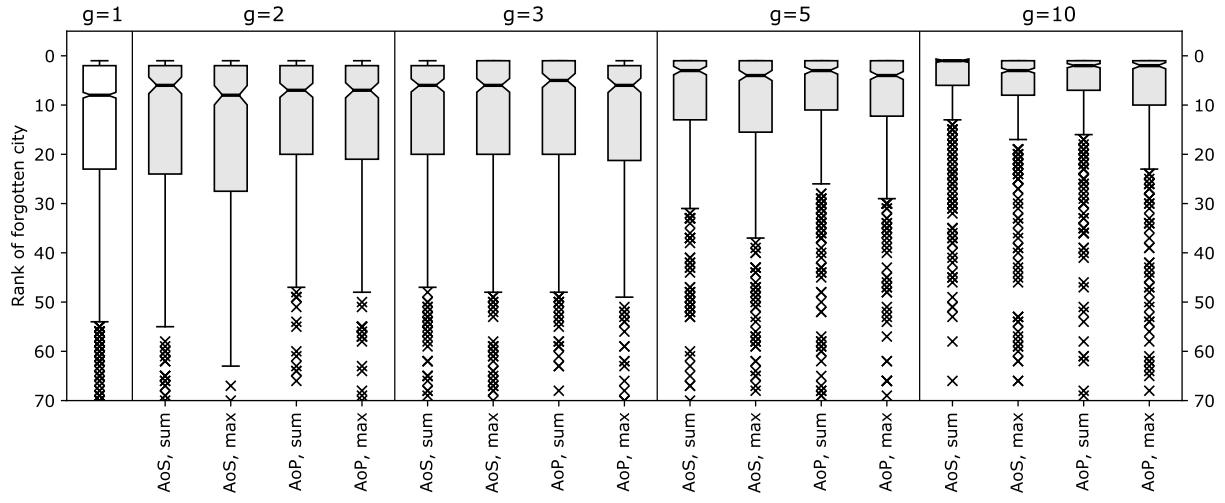


Figure 7.8: Group recommendation results for the  $TT$  model. Vertical axis rescaled to fit the farthest whiskers to improve readability; some outliers hidden.

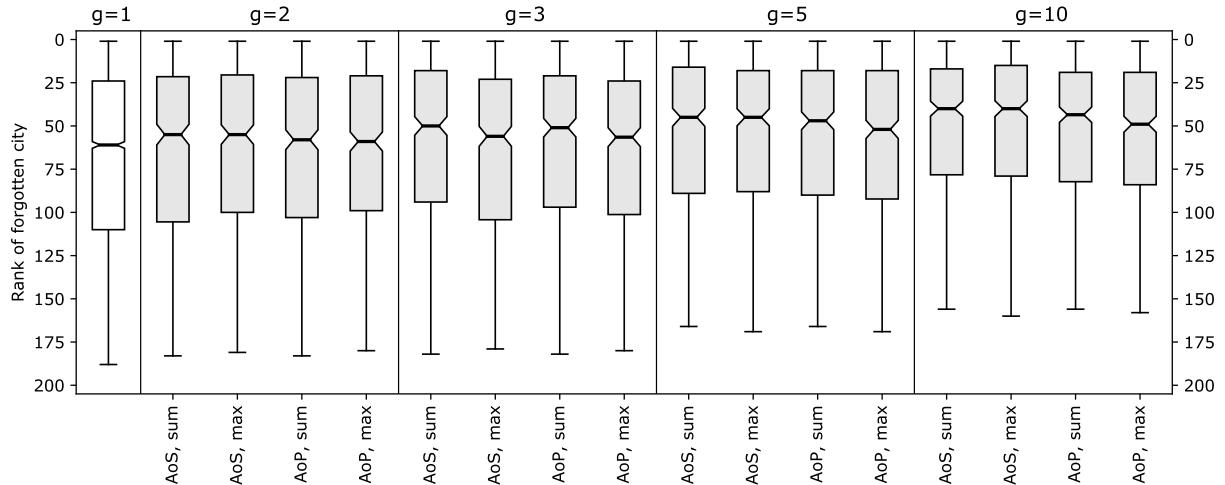


Figure 7.9: Group recommendation results for the  $CC$  model.

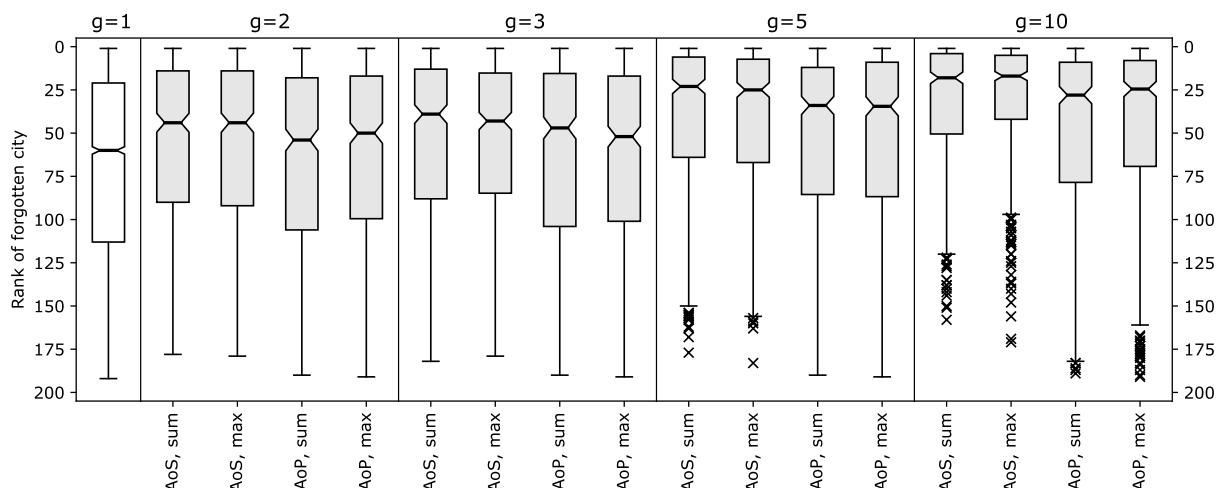


Figure 7.10: Group recommendation results for the  $TC$  model.

- The  $CC$  model is the weakest of the three across every permutation. It also improves little as the group size increases (median  $r^*$  between 40 and 49 for  $g = 10$ , versus 61 for  $g = 1$ ), and shows no consistent dependency on the choice of aggregation method ( $AoS$  vs  $AoP$ , sum vs max). This implies that recommendation based on similarity to the POI category distributions of previously-visited cities is a fundamentally weak method of predicting real-world travel decisions.
- Starting from a point of near-inseparability in the single-tourist case, the  $TC$  model shows a clear advantage over  $CC$  as the size of the group increases. This implies that recommendation based on the POI category preferences of groups of tourists can be an effective method of predicting real-world travel decisions. With this model,  $AoS$  is significantly better than  $AoP$ . The single best results are attained using max-based  $AoS$  with  $g = 10$ ; here the  $r^*$  quartiles lie at ranks 5, 17 and 42.

### 7.1.3 Summary of Findings

In domains where the popularity of available options is highly imbalanced, the measured efficacy of personalised recommendation, compared with that based on population-wide statistics, depends on the sampling method used to generate the test set. Assuming that test cases include all cities in the population of 200 with uniform probability, each of my city recommender models outperforms the naïve and random baselines in terms of its ability to place the forgotten city high on the ranking.  $TT$ , the model based on tourist-tourist similarity (i.e. collaborative filtering), is the strongest of the three by a wide margin, ranking the forgotten city at a median position of 8 in the single-tourist case if a neighbourhood of 50 tourists is used. Larger neighbourhoods reduce the recommendation relevance, while smaller ones carry a risk that the forgotten city does not appear at all in the set of histories. An interesting but tentative result is that the  $TT$  model is most reliable when target tourists have visited relatively few cities in the past, which may be because prolific tourists are less consistent in the kinds of city they choose to visit. An investigation of the weaker performance of the  $CC$  and  $TC$  models has revealed few strong results, although it is clear that more visits to the forgotten city afford a more accurate estimation of its POI category importance vector, and in turn a higher ranking.

To varying degrees, all three models improve in performance when the recommendation target is a group rather than a single tourist.  $TT$  retains its dominance, and with the strongest combination of aggregation methods, places the forgotten city at the very top of the ranking around half the time when the group size is 10. For larger group sizes, a gap also opens up between the performance of  $CC$  and  $TC$ , with the latter ranking the forgotten city around 20 places higher on average for groups of 5 or 10 tourists, most notably when the  $AoS$  aggregation method is used. With the other two models, the choice of aggregation method has no consistent effect.

Table 7.2: Weights learned by the linear model *Lin*, and asymptotic validation set performance, from three runs with each of two settings for the learning parameter  $\omega$ .

$\omega$	Learned weight for feature						Asymptotic $Q_{500}$ value on validation set
	fPop	fCat	fProx	fTime	fDate	fHist	
1	1.07	0.189	1.07	1.00	0.743	1.00	0.849
	1.07	0.0558	1.082	0.996	0.683	1.10	0.848
	1.07	0.0593	1.07	0.988	0.707	1.00	0.849
rand	0.964	-0.0215	0.927	0.875	0.530	0.897	0.848
	0.946	-0.0103	0.912	0.879	0.463	0.841	0.852
	0.876	-0.0323	0.838	0.810	0.419	0.831	0.854

## 7.2 POI Recommendation

### 7.2.1 Optimisation Process

Applying the criteria given in section 6.1.2 to the travel histories dataset, and discarding all but the last qualifying scenario for each (tourist city) pair, yielded 15,326 valid scenarios for the POI recommendation problem. I partitioned these randomly into training, validation and test sets with a 60:20:20 split. Before training the machine learning models for mapping features into scores, I set the hyperparameters for the feature creation process to the same values as given in section 5.2.5<sup>3</sup>:

$$\alpha = 100 \quad \beta = 2000 \quad \gamma = 100 \quad \zeta = 0.1 \quad \kappa = 24$$

I trained the linear model *Lin* according to the method described in section 6.2 with two alternative settings for  $\omega$ . With the first, I set  $\omega = 1$ , which meant the target of training was to swap  $p^*$  with the POI immediately above it in the ranking. With the second (denoted  $\omega = rand$ ), I sampled  $\omega$  uniformly from  $\{1 \dots r^* - 1\}$ . Randomised ordering of the training set meant that both settings for  $\omega$  would yield slightly different weights each time training was completed. To test this, I trained the model three times for each setting, giving the weights shown in table 7.2. Also shown is the asymptotic value of  $Q_{500}$  for the validation set; recall that this is defined as the 500-sample moving average of  $1 - (r^*/|\mathbb{R}^c|)$ . While there is only a small difference in  $Q_{500}$  value,  $\omega = rand$  appears to produce marginally better learning on average. When referring to the *Lin* model throughout the rest of this chapter, I use the weight values learned in the third training run for  $\omega = rand$  (i.e. the bottom row of table 7.2).

Training of the neural network model *NN* was more complex since, in addition to the value of  $\omega$ , the hidden layer topology had an effect on performance. I considered networks with the following topologies:

---

<sup>3</sup>A sensitivity analysis indicated that the exact values of these hyperparameters makes little difference to recommendation performance with this dataset.

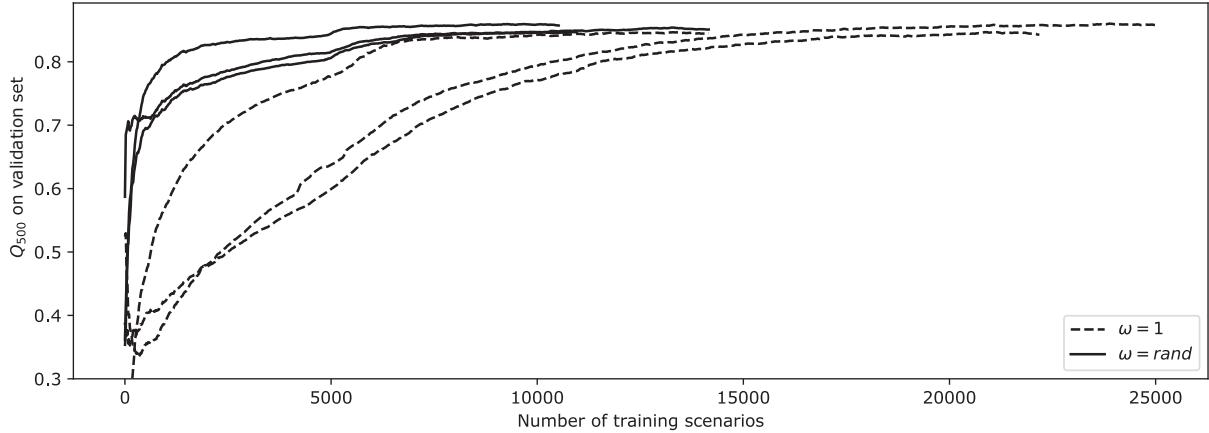


Figure 7.11: Learning curves for neural network model with a single 3-neuron hidden layer, using  $\omega = 1$  or  $\omega = \text{rand}$  for the training method. Each training run was halted when a reduction in  $Q_{500}$  was observed on the validation set.

- One hidden layer 3 or 6 neurons;
- Two hidden layers with 3 or 6 neurons each;
- Three hidden layers with 3 or 6 neurons each.

Starting with the smallest network (one hidden layer, 3 neurons), I again completed three training runs each for  $\omega = 1$  and  $\omega = \text{rand}$ . The learning curves are shown in figure 7.11. It is clear from this visualisation that  $\omega = \text{rand}$  produces markedly faster and more consistent learning than  $\omega = 1$ ; in all three training runs with the former, the validation set  $Q_{500}$  rises permanently above 0.8 within 5,000 training scenarios, whereas for two of the runs with the latter, more than 10,000 are required. The mean asymptotic value of  $Q_{500}$  is also marginally higher with  $\omega = \text{rand}$ : 0.853 instead of 0.850.

Given that larger neural networks tend to require longer training times, I trained all other topologies using  $\omega = \text{rand}$ . Their learning curves are shown in figure 7.12, which demonstrates two clear results. Firstly, the larger networks do indeed take longer to reach convergence. Secondly, asymptotic performance (as measured by  $Q_{500}$ ) sees *no measurable improvement* as the network size is increased, remaining around 0.85 for all topologies. A lack of an upwards trend in performance as model complexity is increased is commonly attributed to over-fitting to the training set. However, figure 7.13 demonstrates that this is not the case in this context. In a surrounding set of experiments, I found that alternative learning rate schedules and topologies also have a negligible effect on asymptotic performance. This suggests that a simple model, even a linear one, is sufficient to come close to maximum recommendation performance on this particular dataset, given my choice of features. This possibility is discussed in greater depth later in this section.

From the set of all training runs for the various network topologies, I selected the two that boast the highest asymptotic validation set  $Q_{500}$  values to carry forward to a full

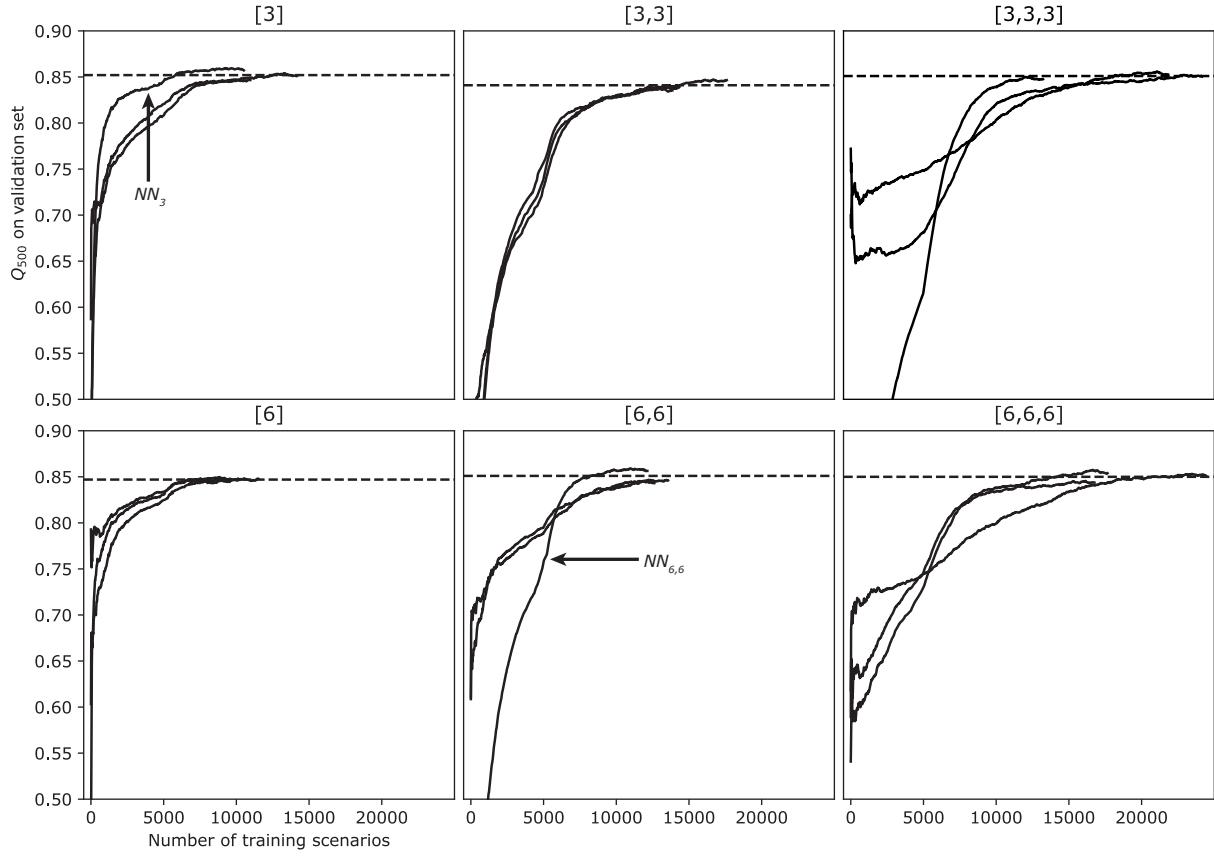


Figure 7.12: Learning curves using  $\omega = \text{rand}$  for neural network model with a variety of hidden layer topologies as indicated by the sub-figure titles (e.g. [3, 3] = two hidden layers, 3 neurons each). Curves for the smallest network [3] copied from figure 7.11 for reference.

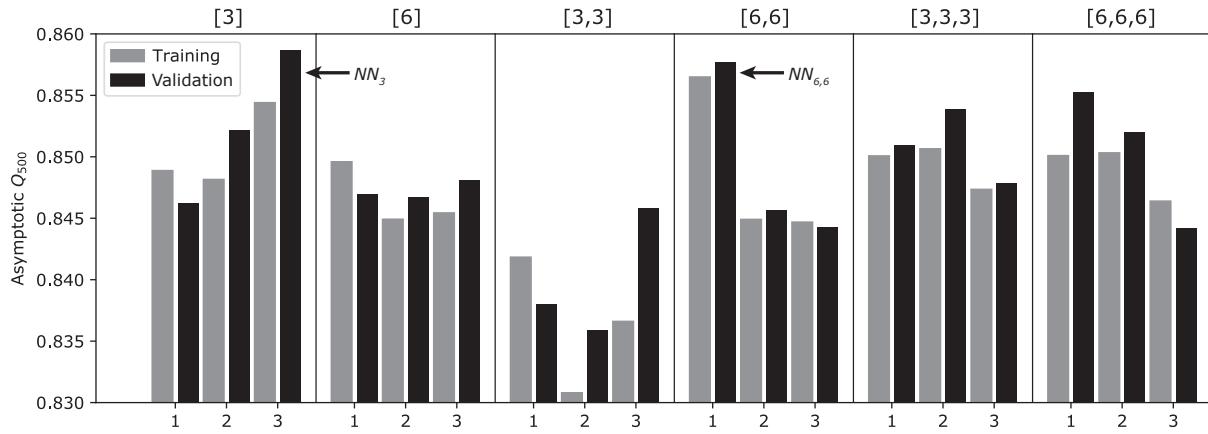


Figure 7.13: Asymptotic  $Q_{500}$  values on both the training and validation sets for each training run from figure 7.12. In the majority of cases (13 out of 18), validation set performance is *higher* than training set performance, a trend which remains unchanged even as the model complexity is increased. This shows that the lack of performance improvement cannot be attributed to overfitting to the training set.

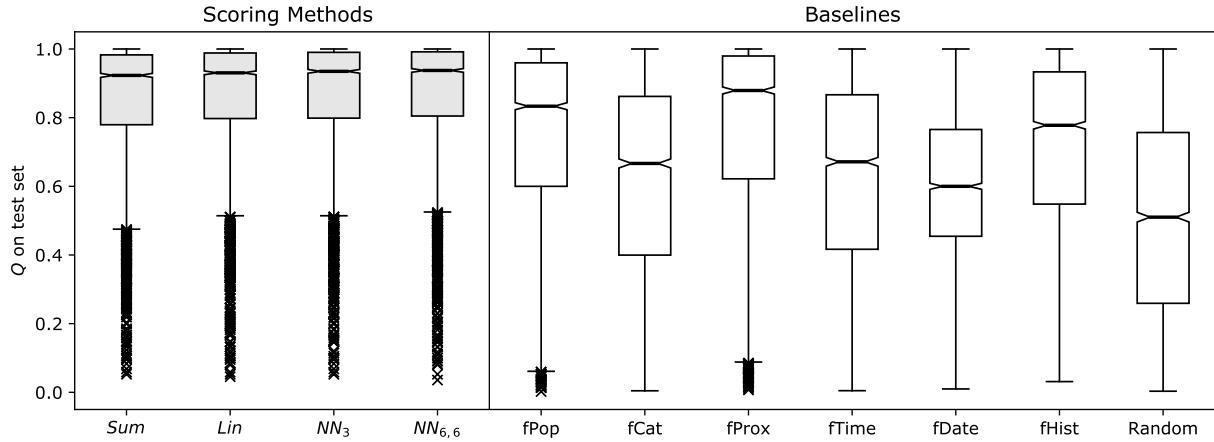


Figure 7.14: Box plots showing the distribution of  $Q$  values for the four POI scoring methods and seven baselines across 3,055 test scenarios.

performance evaluation on the test set. These are the two models indicated by arrows in figures 7.12 and 7.13: one using the smallest topology with a single 3-neuron hidden layer (henceforth denoted  $NN_3$ ) and the other using two 6-neuron hidden layers (denoted  $NN_{6,6}$ ).

### 7.2.2 Performance Comparison

Having trained the scoring methods  $Lin$  and  $NN$  (specifically my chosen variants  $NN_3$  and  $NN_{6,6}$ ), a performance comparison can now be conducted between these options, as well as the simple feature-summing method  $Sum$  and the single-feature and random baselines. For this I use the test set, which consists of 3,055 unseen POI recommendation scenarios. In each scenario, I quantify performance via  $Q$ .

Figure 7.14 contains box plots summarising the distribution of  $Q$  values across the test set for each scoring method, as well as for the seven baselines. It is immediately clear both that all four scoring methods outperform the single-feature baselines, and in turn the baselines are each significantly more informative than random (with fProx, fPop and fHist being the three most effective). Secondarily, in this presentation, there appears to be only a small difference in performance between  $Sum$ ,  $Lin$  and both  $NN$  topologies, which is remarkable in itself: simply summing up the values of the six features is enough to yield comparable recommendation performance to a relatively complex machine learning model trained on tens of thousands of real-world travel histories.

The summary statistics in table 7.3 do show that all three quartiles of  $Q$ , as well as the mean value<sup>4</sup>, increase consistently with model complexity, though differences are on the

<sup>4</sup>It is also worth here noting that the mean  $Q$  values very close to the asymptotic  $Q_{500}$  values attained during training. This shows that the training, validation and test sets all come from the same statistical distribution as desired.

Table 7.3: Summary statistics of the distribution of  $Q$  values for the four scoring methods.

Scoring Method	Quartiles of $Q$			Mean of $Q$	Modal $r^*$
	1	2	3		
<i>Sum</i>	0.779	0.923	0.982	0.848	1 (13.6%)
<i>Lin</i>	0.797	0.931	0.988	0.857	1 (18.5%)
$NN_3$	0.799	0.935	0.990	0.858	1 (19.0%)
$NN_{6,6}$	0.805	0.938	0.992	0.860	1 (19.8%)

order of just 1-2%. A more pronounced difference is visible when considering the modal value of  $r^*$ . All four methods have a mode of 1 (i.e. the single most common rank for the forgotten POI is first), but the percentage of scenarios in which this occurs increases from around 14% for *Sum* to 20% for  $NN_{6,6}$ . This is a promising result, implying that in approximately 1 in 5 cases, the strongest variant of my recommender system can successfully identify the exact place that a tourist will visit next from a set of candidate POIs across an entire city.

Another means of comparison is to look at the proportion of individual scenarios in which each scoring method yields the highest  $Q$ . One complexity of this approach is that in many cases, two methods jointly share first place. This is demonstrated in the left sub-table of table 7.4, which categorises each scenario  $s$  in the test set  $\mathbb{S}$  ( $|\mathbb{S}| = 3055$ ) according to the set of methods that perform best. I denote this set  $\mathbb{B}_s$ . A single performance score  $\text{perf}(method)$  can be computed for  $method \in \{\text{Sum}, \text{Lin}, NN_3, NN_{6,6}\}$  using an equation inspired by the definition of a pignistic probability distribution in Dempster-Shafer theory [53]:

$$\text{perf}(method) = \frac{1}{|\mathbb{S}|} \sum_{s \in \mathbb{S}} \begin{cases} \frac{1}{|\mathbb{B}_s|} & \text{if } method \in \mathbb{B}_s \\ 0 & \text{otherwise} \end{cases} \quad (7.1)$$

The right sub-table of table 7.4 shows this calculation for each method. The resulting values, which sum to 1, can be interpreted as the *probability* that each scoring method is the strongest. While no method dominates, the larger neural network model  $NN_{6,6}$  is clearly separated from the other three, with a probability over twice that of *Sum*. A reasonable conclusion of this performance comparison is therefore that POI recommendation performance can be improved by a more complex scoring method, but not by a very large amount.

### 7.2.3 Correlation Analysis

The correlogram in figure 7.15 visualises the Pearson correlation coefficients between the  $r^*$  values produced by the four scoring methods, and by the six single-feature baselines, on the test set. It is best described as comprising three distinct regions, separated by

Table 7.4: Categorisation of the 3,055 test scenarios according to the set of scoring methods with highest  $Q$  [left], and application of these values to compute the pignistic probability that each method produces the strongest overall recommendation [right].

$\mathbb{B}_s$	#	Pignistic Probability
$\{Sum\}$	316	
$\{Lin\}$	277	
$\{NN_3\}$	370	
$\{NN_{6,6}\}$	847	
$\{Sum, Lin\}$	46	
$\{Sum, NN_3\}$	92	
$\{Sum, NN_{6,6}\}$	31	
$\{Lin, NN_3\}$	94	
$\{Lin, NN_{6,6}\}$	154	
$\{NN_3, NN_{6,6}\}$	135	
$\{Sum, Lin, NN_3\}$	82	
$\{Sum, Lin, NN_{6,6}\}$	41	
$\{Sum, NN_3, NN_{6,6}\}$	19	
$\{Lin, NN_3, NN_{6,6}\}$	185	
All	366	
<i>Sum</i>		$\frac{1}{3055} \left( \frac{316}{1} + \frac{46+92+31}{2} + \frac{82+41+19}{3} + \frac{366}{4} \right) = \mathbf{0.177}$
<i>Lin</i>		$\frac{1}{3055} \left( \frac{277}{1} + \frac{46+94+154}{2} + \frac{82+41+185}{3} + \frac{366}{4} \right) = \mathbf{0.202}$
<i>NN<sub>3</sub></i>		$\frac{1}{3055} \left( \frac{370}{1} + \frac{92+94+135}{2} + \frac{82+19+185}{3} + \frac{366}{4} \right) = \mathbf{0.235}$
<i>NN<sub>6,6</sub></i>		$\frac{1}{3055} \left( \frac{847}{1} + \frac{31+154+135}{2} + \frac{41+19+185}{3} + \frac{366}{4} \right) = \mathbf{0.386}$

dotted lines in the figure and denoted **A**, **B** and **C**.

Region **A** shows the correlations in  $r^*$  between the four scoring methods, which are all high, lying between 0.882 (*Lin* and *NN<sub>6,6</sub>*) and 0.958 (*Lin* and *NN<sub>3</sub>*). This indicates that, as expected, the methods generally perform well in similar sets of scenarios, so are not using the underlying features in radically different ways. It also suggests that any instances of poor performance are not due to particular failings of each learned model, but rather due to certain POI visits being inherently unpredictable given the available features. However, the correlation values concerning *NN<sub>6,6</sub>*, the larger neural network, are consistently somewhat lower than the others, which points to this model having learned to prioritise the features slightly differently.

Region **B** shows the  $r^*$  correlations between the scoring methods and each of the single-feature baselines, which can be interpreted as indicating how strongly each feature is taken into account by each method. As should be expected, the uniform-weighting model *Sum* shows the least polarisation between features, although correlates unusually heavily with fProx. This indicates that this feature exhibits greater variability than the other five, highlighting a potential drawback of the simple z-normalisation method. Normalising by standard deviation is most meaningful for symmetric distributions, whereas in cities with clustered POIs, the distribution of fProx will have a highly asymmetric exponential-like form. For the other scoring methods, there is an apparent trend of increasing polarisation towards fProx, fPop and fHist (in order of weighting) as the model complexity is increased. Referring back to figure 7.14, it can be seen that this corresponds exactly with the predictive efficacy of these features. The learning models have therefore successfully

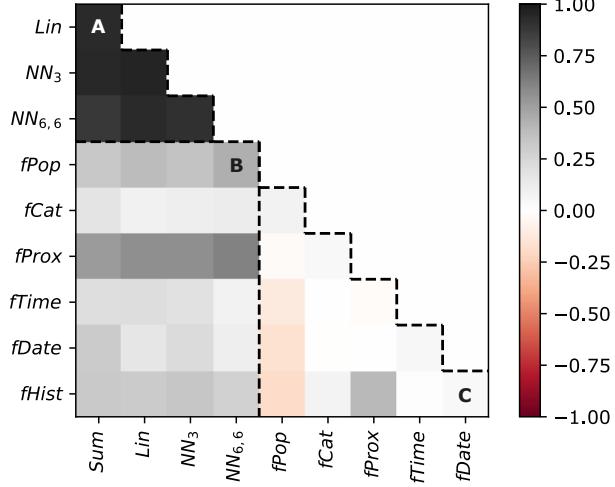


Figure 7.15: Correlogram visualising the Pearson correlation in  $r^*$  values for all POI scoring methods and single-feature baselines on the test set.

identified which features are more useful than others. The polarisation effect is most pronounced with  $NN_{6,6}$ , which may help to explain the slight departure of this method from the others, as discussed above.

Finally, region **C** concerns the  $r^*$  correlations between the features themselves. The high prevalence of near-white cells in this region indicates that for the most part, the features are statistically independent. This is a desirable property: independent features are less redundant and contain more information than highly-correlated ones. There are, however, two notable exceptions. Firstly, fProx and fHist are positively correlated (value = 0.398), which suggests that ranking candidate POIs by proximity to the current location has a somewhat similar effect to ranking by the historical time-weighted coincidence rate with previously-visited POIs. This in turn suggests that the informativeness of fHist could be improved if it were normalised by proximity<sup>5</sup>. Secondly, and more subtly, the performance of fPop correlates negatively with each of the three features which involve normalising by popularity. This may be an instance of the subtle *explaining away* phenomenon: a high value of fTime, fDate or fHist is sufficient to explain a tourist’s visit to a POI, so the likelihood that the POI is *also* globally popular is reduced [54].

#### 7.2.4 Performance Patterns

As suggested in the previous subsection, it appears that certain POI visits in the dataset are inherently more predictable than others. My final category of analysis seeks to identify any patterns in recommendation performance as a function of aspects of the scenario. In figure 7.16, the mean  $Q$  value for each scoring model is separated by city. To ensure reasonable sample sizes, only those cities with at least 20 scenarios in the test set are

<sup>5</sup>It would also be worth experimenting with different values for the time-discounting parameters  $\zeta$  and  $\kappa$ , to reduce the bias towards POI pairs that are visited very close in time.

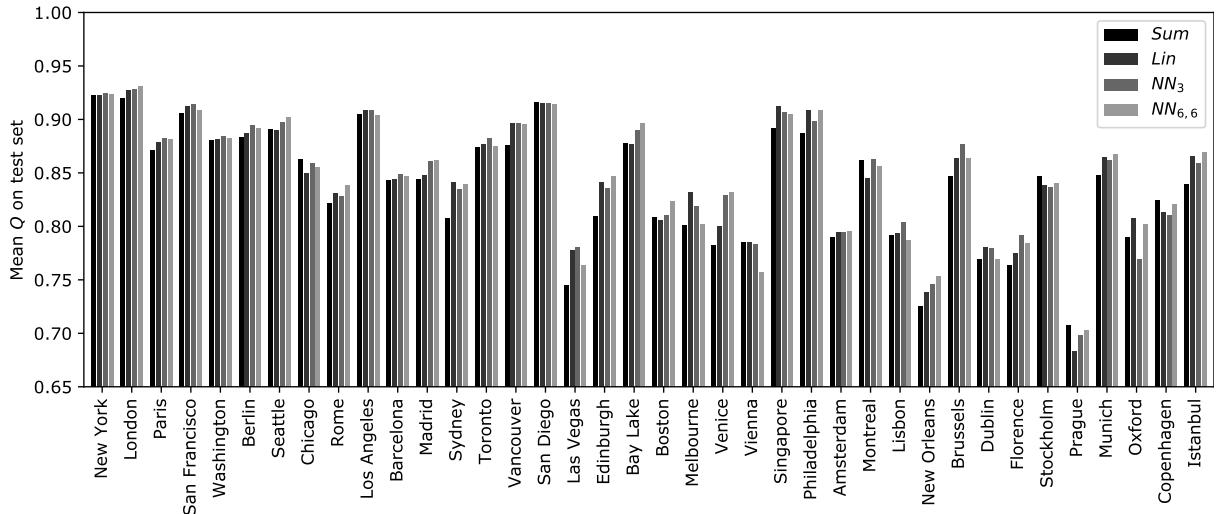


Figure 7.16: Mean  $Q$  value for each scoring model, separated by city.

shown. The cities are ordered left-to-right according to the number of labelled visits in the dataset at large (see figure 4.2). There is a generally negative trend in recommendation performance as the global popularity of the target city is decreased, which is common across all scoring methods. In particular, the two cities in which the mean  $Q$  is highest – London and New York – are also the ones with the most recorded visits. This suggests that, all else being equal, obtaining more data about tourists’ histories in a city improves recommendation performance. Another possible trend is that performance is better in English-speaking cities: all of the cities in which at least one model attains a mean of  $Q = 0.9$  or above are wholly or partly Anglophone. This may be attributable to the dataset itself. OpenStreetMap provides the majority of its POI metadata in English, so POI labelling is likely to have been somewhat more reliable in these locations.

Figure 7.17 similarly separates out performance on the test set according to the category of the forgotten POI. Again, only categories with at least 20 scenarios are included, and the left-to-right ordering corresponds with overall popularity across the dataset. In this case, there is no visible trend with popularity, but there significant variability between individual categories. A tentative observation is that the categories for which performance is highest (memorial, garden, monument, town hall) are all static features with predictable flows of tourists, while the ones that are least well predicted are highly dependent on specific events such as plays, sports games and lectures (theatre, stadium, university), or alternatively on weather conditions (beach). A more sophisticated recommender system could incorporate weather forecast information and event times to yield more appropriate recommendations with respect to these categories.

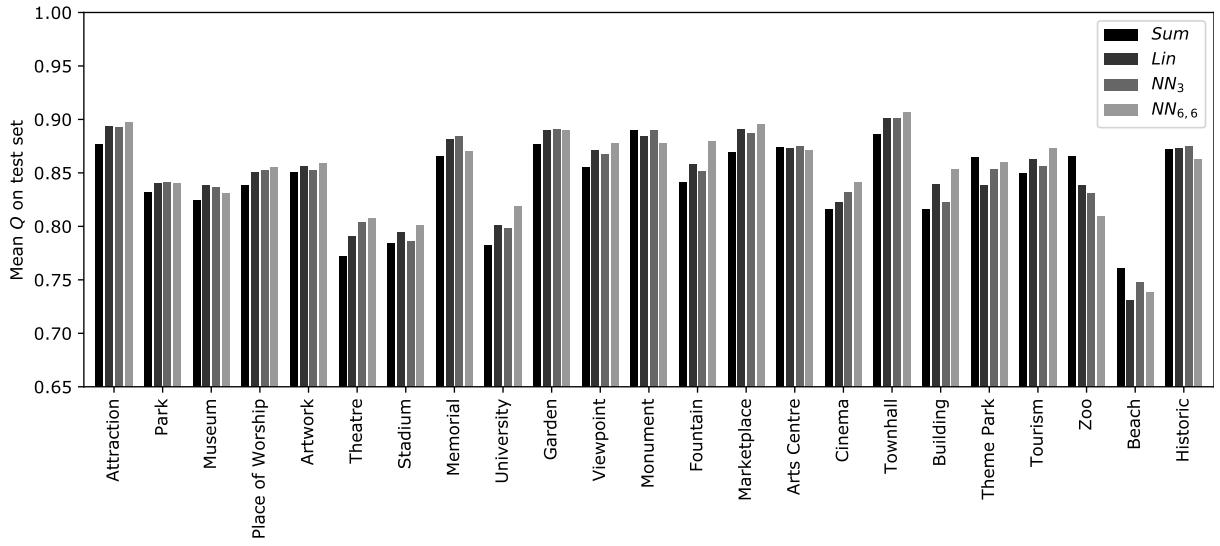


Figure 7.17: Mean  $Q$  value for each scoring model, separated by forgotten POI category.

### 7.2.5 Summary of Findings

Using the learning approach outlined in section 6.2, a neural network with two 6-neuron hidden layers can be trained to output POI recommendation scores that place the forgotten POI  $p^*$  at a median position of 6% down the ranking of candidate all POIs. This is markedly better than all single-feature baselines, which rank POIs naïvely based on metrics such as overall popularity and proximity to the target tourist’s current location. However, it is also remarkable that far smaller networks or linear models optimised via the same learning approach, and even a straight summation of the (z-normalised) features, perform just 1-2% worse in terms of average rank percentage. Reframing the comparison as a tallying of the single highest-ranking method on a per-scenario basis reveals a greater advantage for the larger neural network, but not enough for its superiority to be incontrovertible. This suggests that the choice and engineering of the input features themselves, rather than the exact method of their combination, is the source of most of the recommendation performance. In some real-world contexts it may be desirable to sacrifice some of the additional performance of the more complex model to afford greater interpretability.

Correlation analysis of the four scoring methods shows a high degree of similarity in their ranking of  $p^*$  across the set of scenarios; if one method performs well on a given scenario, the others are likely to do the same. Further correlation of each method with respect to each individual feature shows that the learning models place greater weight on the fProx, fPop and fHist features than the other three. This result, combined with the greater predictive efficacy of the baseline rankings generated by these features, makes for compelling evidence that they are more valuable metrics for the POI recommendation problem. A final set of feature-to-feature correlations shows that they are largely

independent, with little redundancy. The instances where this is not the case could be addressed by small changes to the feature engineering process.

Recommendation performance varies as a function of the city in which the recommendation is made. Visits in more popular cities are generally easier to predict, likely due to better data availability. All exceptions to the trend are Anglophone cities, suggesting that the underlying dataset is higher quality in these locations. Performance is also dependent on the category of  $p^*$ . Categories containing static POIs such as monuments and town halls are more readily predicted than those whose appropriateness is highly sensitive to weather conditions or events schedules.

# Chapter 8

## Conclusion

### 8.1 Summary of Outcomes

In this project I have developed a variety of models and techniques for the data-driven recommendation of touristic experiences. The first problem I have addressed is that of ranking candidate cities to visit for a group of travellers, based on their previous travel histories and preferences over experience types as summarised by their POI category preference distributions. I proposed three alternative recommendation models based on collaborative filtering (*TT*), content-based relation of candidate cities to those already visited (*CC*) and matching of cities to tourist's preference distributions (*TC*). I also considered two aggregation methods based on either scores (*AoS*) or preferences (*AoP*). The second problem addressed is that of ranking POIs to visit at a designated time during an ongoing city holiday; for this problem I considered only a single target tourist. I developed six numerical features to quantify the suitability of candidate POIs based on overall popularity, the target tourist's known preferences, contextual factors (location, time and date) and historic correlations with previously-visited POIs. I also proposed three methods for mapping features into recommendation scores: a simple summation *Sum*, a linear regression model *Lin* and a feedforward neural network *NN*.

Implementation of the recommender systems required a dataset of travel histories, and for this I followed numerous previous works in harnessing the *Flickr YFCC100M* dataset of location- and time-tagged photos. I developed a novel pipeline for converting this raw data source into the desired form, consisting of: identifying photos taken within each of the top 200 most popular cities worldwide; clustering each user's photos into visits based on spatiotemporal proximity; assembling a set of words for each visit from the associated photo metadata; comparing these words to descriptions of nearby POIs on *OpenStreetMap*; and assigning a POI to a visit if the match was sufficiently strong. The single greatest departure from the dataset creation methods in existing works was the inclusion of a secondary bootstrapping sweep, which sought to label more visits by

relating their word sets to distributions of word popularities across already-labelled visits. This process brought the total number of labelled visits to around 1.3 million across the 200 cities, concerning 64,000 individual tourists.

Recommender systems are notoriously challenging to evaluate in offline environments, especially for tourism where feedback on recommendation quality is sparse and delayed. I took a cue from previous work in evaluating my models via cross-validation with the dataset itself, effectively reformulating the recommendation problem as that of predicting the travel decisions of real tourists. Performance was quantified in terms of the recommendation rank of a single ‘forgotten’ city or POI amongst the set of candidates. I compared model performance to numerous more naïve baseline ranking techniques, some of which have also been used in existing work. On the city recommendation problem, the *TT* model significantly outperformed the other two, ranking the forgotten city at a median position of 8 out of 200 in the single-tourist case with a neighbourhood of 50 tourists. For all three models, performance improved when the recommendation target was a group rather than a single tourist. *TT* remained the strongest, placing the forgotten city first in the ranking around half the time for groups of 10 tourists. The *CC* model also began to outperform *TC* with groups, ranking the forgotten city around 20 places higher on average for groups of 5 or 10 tourists, most notably when the *AoS* aggregation method was used. Otherwise, the choice of aggregation method had little effect.

For the POI recommendation problem, two of the features-to-score mapping techniques were machine learning models that required training before use. For this I developed a developed a learning-to-rank algorithm that performed two kinds of weight update: one to move the forgotten POI higher in the ranking by swapping it with a higher-ranked POI, and another to spread the set of recommendation scores across the range [0, 1] to aid learning stability and speed. This algorithm proved effective for training both the *Lin* model and various *NN* topologies. However, all models consistently converged to essentially the same performance level on the validation set. This lack of an increase in performance with model complexity could not be attributed to overfitting, and instead suggested that a linear model is sufficient to attain near-optimal performance on the problem as defined, given the set of six features. That said, when evaluating the various methods on a held-out test set, the larger of two *NN* topologies (two 6-neuron hidden layers) was shown to be the strongest, placing the forgotten POI at a median position of 6% down the ranking of candidate POIs. The other models performed around 1-2% worse in terms of average rank percentage, and all outperformed the baselines by a wide margin. Reframing the comparison as a tallying of the single highest-ranking method on a per-scenario basis suggested a greater advantage for the larger neural network. In surrounding analyses, I found that visits in popular and Anglophone cities were predicted best, likely due to better data quality in these locations. Static POI categories were also better predicted than those that are weather- or event-sensitive.

## 8.2 Summary of Contributions

I summarise the key contributions of this project as follows:

- **Concept:** A design for recommendation tools that uses the same underlying social dataset, and several of the same derived features, to perform both city- and POI-level recommendation. In both cases, the output is a full ranking of candidate options, rather than a single suggestion or shortlist, which maximises the scope for subsequent applications.
- **Dataset:** A novel dataset of real-world travel histories derived from the *Flickr YFCC100M* archive in combination with *OpenStreetMap*. The dataset is larger in absolute terms than any similarly-derived set that could be found in the existing literature, covers 200 cities worldwide instead of the 4-10 typically seen, and is markedly higher in accuracy than the single most comparable extant dataset.
- **Models:** For the city recommendation problem, a collaborative filtering approach that uses per-city photo counts as an implicit rating device, and a content-based approach that characterises a city by its distribution of visits to various POI categories. For the POI recommendation problem, a selection of preference-, time- and history-sensitive features for quantifying contextual POI suitability, which complement the commonly-used measures of overall popularity and proximity.
- **Algorithms:** A process for reliably deriving POI visitation from photo metadata, including a novel bootstrapping stage that uses the population-wide distribution of words in user-provided text to infer the most likely POI to assign to a set of photos. Elsewhere, a stable learning-to-rank algorithm that works to move a single target item in each training example to the top of the ranking.
- **Results:** At a high level, a consistent outperforming of baseline methods across both problems. Strong evidence that, with the given problem formulation, conventional collaborative filtering over previous travel histories outperforms content-based recommendation at predicting the chosen city destination of a single tourist, and that performance further increases when recommending to groups. For the POI recommendation problem, the indication that a linear model attains close to the maximum possible prediction performance, though can be beaten by a neural network with two 6-neuron hidden layers.

## 8.3 Areas for Improvement and Future Work

The following are some limitations of the approach taken in this project, and avenues for future work that could improve or extend the outcomes:

- **Improved Dataset Creation:** This project left unexplored many additional methods for processing *Flickr* and *OpenStreetMap* data into travel histories. Topic modelling techniques, such as latent Dirichlet allocation, could be used to more robustly identify the most likely POI for a visit and gracefully handle uncertainty. A different direction would be to utilise the image content itself, visually matching uploaded photos to known template images or *Google Street View* data to determine exact locations. The two approaches need not be mutually exclusive.
- **Hybridisation of City Recommendation:** While the collaborative filtering model *TT* has been found to outperform the alternative models *CC* and *TC* on the city recommendation problem, the latter two are still significantly better than random. This begs the question as to whether further performance improvement could be attained by combining the three approaches into a single model. This could take a form similar to the feature-to-score mapping used by the POI recommender.
- **Refined POI Features:** In this project, the effect of the feature hyperparameters for POI recommendation was not systematically studied, and optimisation of these values could improve performance. Additionally, fHist may benefit from being normalised by proximity, and a better expression of user preference than fCat could be sought, since this feature does not appear to carry much predictive power.
- **Extended Functionality:** A natural extension of the POI recommender system would be to adapt it to work with tourist groups. In addition, the models could be adapted to output sequences of POIs (i.e. solve the orienteering problem) rather than focus exclusively on the single next visit. The simplest method of doing this would be to run the model iteratively, estimating the length of each visit using historic trends. Finally, additional data sources could be harnessed to enable the consideration of travel durations, opening times and weather conditions.
- **User Testing:** The assessment of predictive performance on a dataset of real travel histories is a reasonable and popular approach for tourism recommendation, but is an incomplete substitute for a real user study. User testing would be a crucial next step in the development of the project, with performance measured by eliciting feedback on perceived recommendation quality from the study participants.
- **Deployment:** Ultimately, the study of recommender systems should be oriented towards real-world deployment. Once sufficiently tested and refined, the city- and POI-level recommender systems could be deployed as an integrated web or mobile application for use by real tourists. The deployed system could operate using a user's own data from Flickr or another social media service. Alternatively, the generic models presented in chapter 3 could be adapted to work with another source of histories, such as the location-tracking functionality of modern smartphones.

# Bibliography

- [1] World Tourism Organization (UNWTO). *UNWTO Tourism Highlights: 2018 Edition*. Aug 2018.
- [2] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.
- [3] Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. *Group recommender systems: An introduction*. Springer, 2018.
- [4] Spotify Technology S.A. *Spotify*.
- [5] Netflix Inc. *Netflix*.
- [6] Amazon.com Inc. *Amazon*.
- [7] Facebook Inc. *Facebook*.
- [8] Match Group Inc. *Match.com*.
- [9] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, volume 83. WoUOn-gong, 1998.
- [10] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.
- [11] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.
- [12] Alexander Felfernig and Robin Burke. Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, page 3. ACM, 2008.
- [13] Li Chen and Pearl Pu. Evaluating critiquing-based recommender agents. In *AAAI*, pages 157–162, 2006.

- [14] Li Chen and Pearl Pu. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1-2):125–150, 2012.
- [15] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.
- [16] Virginia Tsintzou, Evangelia Pitoura, and Panayiotis Tsaparas. Bias disparity in recommendation systems. *CoRR*, abs/1811.01461, 2018.
- [17] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.
- [18] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13, 2016.
- [19] Sriharsha Dara, C. Ravindranath Chowdary, and Chintoo Kumar. A survey on group recommender systems. *Journal of Intelligent Information Systems*, Jan 2019.
- [20] Francesco Ricci. Travel recommender systems. *IEEE Intelligent Systems*, 17(6):55–57, 2002.
- [21] JA Delgado and Richard Davidson. *Knowledge bases and user profiling in travel and hospitality recommender systems*. Citeseer, 2002.
- [22] Joan Borràs, Antonio Moreno, and Aida Valls. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications*, 41(16):7370–7389, 2014.
- [23] Montserrat Batet, Antonio Moreno, David Sánchez, David Isern, and Aïda Valls. Turist@: Agent-based personalised recommendation of tourist activities. *Expert Systems with Applications*, 39(8):7319–7329, 2012.
- [24] Luis Castillo, Eva Armengol, Eva Onaindía, Laura Sebastiá, Jesús González-Boticario, Antonio Rodríguez, Susana Fernández, Juan D Arias, and Daniel Borrajo. samap: An user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications*, 34(2):1318–1332, 2008.
- [25] R Logesh, V Subramaniyaswamy, V Vijayakumar, and Xiong Li. Efficient user profiling based intelligent travel recommender system for individual and group of users. *Mobile Networks and Applications*, pages 1–16, 2018.
- [26] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. Clr: a collaborative location recommendation framework based on co-clustering. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 305–314. ACM, 2011.

- [27] Yue Shi, Pavel Serdyukov, Alan Hanjalic, and Martha Larson. Personalized landmark recommendation based on geotags from photo sharing sites. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.
- [28] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 363–372. ACM, 2013.
- [29] Ranieri Baraglia, Cristina Ioana Muntean, Franco Maria Nardini, and Fabrizio Silvestri. Learnext: learning to predict tourists movements. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 751–756. ACM, 2013.
- [30] T Yamasaki, A Gallagher, and T Chen. Personalized intra-and inter-city travel recommendation using large-scale geo-tagged photos. In *2nd ACM Multimedia Workshop on Geotagging and Its Applications in Multimedia*, 2013.
- [31] Kwan Hui Lim, Jeffrey Chan, Shanika Karunasekera, and Christopher Leckie. Personalized itinerary recommendation with queuing time awareness. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 325–334. ACM, 2017.
- [32] Liangliang Cao, Jiebo Luo, Andrew Gallagher, Xin Jin, Jiawei Han, and Thomas S Huang. Aworldwide tourism recommendation system based on geotaggedweb photos. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2274–2277. IEEE, 2010.
- [33] Martin Goossen, Henk Meeuwesen, Jappe Franke, and Marjolijn Kuyper. My ideal tourism destination: Personalized destination recommendation system combining individual preferences and gis data. *Information Technology & Tourism*, 11(1):17–30, 2009.
- [34] Ulrike Gretzel, Nicole Mitsche, Yeong-Hyeon Hwang, and Daniel R Fesenmaier. Tell me who you are and i will tell you where to go: Use of travel personalities in destination recommendation systems. *Information Technology & Tourism*, 7(1):3–12, 2004.
- [35] Liliana Ardissono, Anna Goy, Giovanna Petrone, Marino Segnan, and Pietro Torasso. Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. *Applied artificial intelligence*, 17(8-9):687–714, 2003.

- [36] Anthony Jameson. More than the sum of its members: challenges for group recommender systems. In *Proceedings of the working conference on Advanced visual interfaces*, pages 48–54. ACM, 2004.
- [37] Amra Delic, Julia Neidhardt, Thuy Ngoc Nguyen, and Francesco Ricci. An observational user study for group recommender systems in the tourism domain. *Information Technology & Tourism*, 19(1-4):87–116, 2018.
- [38] Kevin McCarthy, Lorraine McGinty, Barry Smyth, and Maria Salamó. The needs of the many: a case-based group recommender system. In *European Conference on Case-Based Reasoning*, pages 196–210. Springer, 2006.
- [39] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.
- [40] Lin Wan, Yuming Hong, Zhou Huang, Xia Peng, and Ran Li. A hybrid ensemble learning method for tourist route recommendations based on geo-tagged social networks. *International Journal of Geographical Information Science*, 32(11):2225–2246, 2018.
- [41] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. Personalized tour recommendation based on user interests and points of interest visit durations. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 1778–1784, 2015.
- [42] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. Towards next generation touring: Personalized group tours. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.
- [43] Michalis Korakakis, Evangelos Spyrou, Phivos Mylonas, and Stavros J Perantonis. Exploiting social media information toward a context-aware recommendation system. *Social Network Analysis and Mining*, 7(1):42, 2017.
- [44] Shuhui Jiang, Xueming Qian, Jialie Shen, and Tao Mei. Travel recommendation via author topic model based collaborative filtering. In *International Conference on Multimedia Modeling*, pages 392–402. Springer, 2015.
- [45] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, volume 8, page 6, 2013.
- [46] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

- [47] Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 2003.
- [48] OpenStreetMap contributors. OpenStreetMap: the free wiki world map. <https://www.openstreetmap.org>, 2019.
- [49] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
- [50] OpenStreetMap Wiki. Overpass api — openstreetmap wiki, 2019. [Online; accessed 3 June 2019].
- [51] TripAdvisor.com. Top attractions, 2019. [Online; accessed 10 August 2019].
- [52] Robert McGill, John W Tukey, and Wayne A Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.
- [53] Philippe Smets and Robert Kennes. The transferable belief model. *Artificial intelligence*, 66(2):191–234, 1994.
- [54] Judea Pearl. Embracing causality in default reasoning. *Artificial Intelligence*, 35(2):259–271, 1988.