A Recommender System for the Gaming Industry

David Lynch, B.Sc., and Daniel Barnes, B.A.I.

A thesis submitted to University College Dublin in part fulfilment of the requirements of the degree of Master of Science in Business Analytics

Michael Smurfit Graduate School of Business,

University College Dublin

September, 2014

Supervisor: Dr. James McDermott

Head of School: Professor Ciarán Ó hÓgartaigh

Dedication

To our families for their support and encouragement.

Contents

List of Figures **v**

List of tables **vi**

1 Introduction **1**

1.1 Opening Remarks 1
1.2 Research Context and Gap 1
1.3 Business Motivation 3
1.3.1 Cross-sell in the Online Sportsbook 4
1.4 Research Aim 4
1.5 Research Scope 5
1.6 Project Outline 6

2 Literature Review **7**

2.1 Introduction 7
2.2 Main Research Themes 7
2.2.1 Overview of the CF Literature 8
2.2.2 Implicit Feedback in Factor Models 11
2.2.3 Content Filtering 13
2.3 Dealing with Temporal Dynamics 14
2.3.1 Time and Context-Aware Recommendations 14
2.3.2 Nearest-Neighbour Methods 15
2.3.3 Combining Neighbourhood and Factor Models 18
2.3.4 Temporal Dynamics in Neighbourhood and Factor Models 20
2.4 Adapting the Models 22

3 Methodology **24**

3.1 Software 24
3.2 Data Granularity 25
3.3 Algorithm 27

3.3.1 Data Acquisition 27
3.3.2 Data Pre-Processing 28
3.3.3 Basic Collaborative Filtering 29
3.3.4 Post Processing 1: Principled Randomisation 32
3.3.5 Post Processing 2: Temporal Filtering 35
3.4 Efficiency 37
3.5 Evaluation 37
3.5.1 Evaluation along the Temporal Dimension 39
3.5.2 Evaluation Results 40
3.5.3 Limitations of the Evaluation 42

4 Results and Discussion **45**
4.1 Introduction 45
4.2 Results 45
4.2.1 Unreliable Recommendations 45
4.2.2 Dithered Recommendations 47
4.2.3 Effects of Temporal Filter 49
4.2.4 Personalised Recommendations 51
4.3 Cross-sell 53
4.3.1 Implementation 53
4.3.2 Patterns 54

5 Conclusions and Future Research **56**

Appendix **59**
5.1 Detailed Program Description 59
5.2 Event Division and Aggregation 62

List of Figures

2.1 Collaborative Filtering an overview 9
2.2 Graphical interpretation of latent factor algorithms 10

3.1 Illustration of the program flow. 27
3.2 Format of the MMSql data pull—Sample Data. 28
3.3 Raw input (l), preference (m) and confidence (r) matrices. . . . 30
3.4 Illustration of Dithering. 34
3.5 Scalability 38
3.6 Evaluation by Percentile Ranking 39
3.7 Evaluation results 43
3.8 Statistical significance of the evaluation 44

4.1 Temporal Filter, US Open Golf 50

5.1 Chord Diagram showing Full Cross-sell model. 64
5.2 Chord Diagram showing Premier League (Betting in Running) Cross-sell information. 65
5.3 Chord Diagram showing Basketball Cross-sell information. . . . 66
5.4 Chord Diagram showing Lottery Cross-sell information. 67
5.5 One Filter does not fit all Sports. 68

List of Tables

4.1 Recommendations, Minimal Betting History 46
4.2 Recommendations, Minimal Betting History, Saturday 46
4.3 Recommendations, Minimal Betting History, Sunday 46
4.4 Recommendations, Non-Dithered 49
4.5 Recommendations, Dithered Results 1 50
4.6 Recommendations, Dithered Results 2 51
4.7 Time Sensitive Output 51
4.8 Personalised Recommendations 1 52
4.9 Personalised Recommendations 2 53

Acknowledgements

We would like to express our appreciation to Mr. Eoin Slowey, Mr. Eamonn O'Loughlin, the entire Customer Intelligence Team at Paddy Power, and Dr. James McDermott for their valuable and constructive suggestions during the planning and development of this research work. We also acknowledge the input of Prof. Cathal Brugha for his helpful suggestions on data visualisation.

Important Abbreviations

CF — Collaborative Filtering
CB — Content-Based Filtering
NN — Nearest Neighbour
r_{ui} — Rating of user u for item i
q_i — Item factor vector (Explicit Feedback)
p_u — User factor vector (Explicit Feedback)
y_i — Item factor vector (Implicit Feedback)
x_u — User factor vector (Implicit Feedback)
RMSE — Root Mean Square Error

Abstract

This project will demonstrate that existing recommender system technology can be adapted to an application in the gaming industry. Key challenges include achieving a scalable implementation, adapting the system to respect high value customers, ensuring the product can drive long term value and extracting cross-sell patterns from implicit customer data. A novel evaluation strategy is proposed and implemented to prove at $\alpha = 5\%$ the utility of post processing a as treatment for the temporal complexities at play in the Sporting calendar. A novel solution to the Cold Start problem in Collaborative Filtering is suggested. The research extends the reach of Recommender Systems into a industry where small edges are big business.

Chapter 1

Introduction

1.1 Opening Remarks

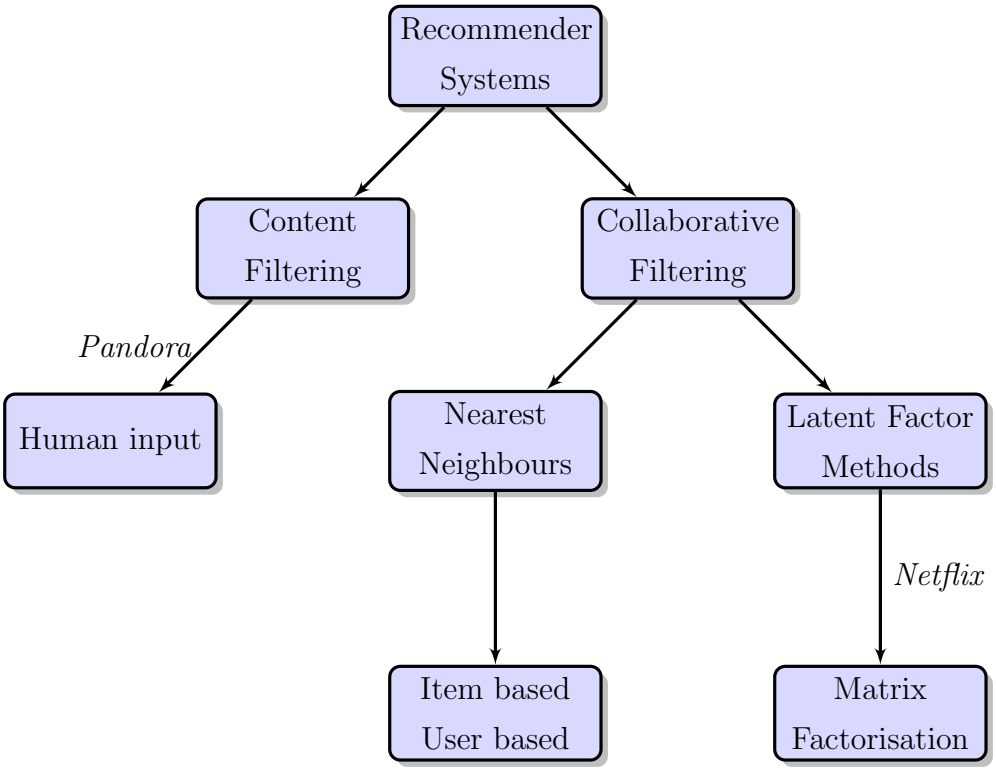
Recommender systems have become an indispensable technology for online service providers. The basic strategy employed by recommender systems is to exploit known information about customers, e.g. their purchasing records or demographics, in order to learn their preferences. Such systems are used in a variety of manners, where Netflix and Amazon.com are successful implementations in the movie streaming and retail spaces respectively. In this project we delve into the recommenders literature, identify fit for purpose algorithms for our application domain within the gaming industry and explore some novel adaptations.

1.2 Research Context and Gap

One industry that has not yet incorporated recommender systems into their business model is the betting industry. This thesis examines how data gathered by such a company, Paddy Power, can be used to generate recommendations for their customers. The betting industry is highly competitive, with all companies essentially offering customers the exact same product. In this crowded marketplace, engaging with customers and retaining their attention is vital, and Paddy Power has a dedicated Customer Intelligence Team (CIT) for this very purpose. As part of their efforts to give customers the best possible experience, Paddy Power now want to implement a recommender system that will offer existing customers new markets to explore. The aim of this project

is to explore the relationships between different sporting markets thus helping Paddy Power to tailor marketing to customers at the individual level.

The literature distinguishes between implicit and explicit recommender systems, in reference to the data used during training. Content Based recommenders are distinguished from their Collaborative Filtering cousins. The former are entirely dependent on domain knowledge and hence are deemed to be unsuitable for this project. The flowchart illustrates the full taxonomy with branches for the Nearest-Neighbour (NN) and Latent Factor models.



NN approaches learn the user-item interaction on the basis of either an item-item or user-user similarity metric. On the other hand, Latent Factor models learn a mapping of users and items to a common high dimensional factor space. Relative positions within this space indicate the degree of likeness between users and items, quantified by computing the dot product between their respective factor vectors. Both NN and Latent Factor models are domain free so that recommendations are derived without the need for human annotation. Latent Factor approaches are more principled as they do not require

the selection of an arbitrary similarity metric but rather the model is learned from minimisation of a suitable objective function. In section 2.2.2 we identify an existing factor algorithm based on implicit feedback that lends itself to analysing cross-sell in the Sportsbook.

Our research has indicated that no online betting companies operating in Ireland or the UK currently employ a sophisticated recommender system, nor was any reference to a recommender system in the betting industry found during the literature review. This project will outline an algorithmic framework that can be actioned by Paddy Power when they deploy a recommender system across their online business. This will be achieved by examining the state of the art in recommender system technology, identifying the most suitable algorithms on which to base the recommender, and modifying.

Now, the main challenge in this project, which seemingly has not been addressed to date, is dealing with the highly transient nature of the sporting calendar. When a company like Amazon recommends a product to one of their customers, there is a high likelihood that that product will still be on sale in the subsequent days, months or even years. Similarly, Netflix is an on-demand service where the customer can watch what they want, when they want. Betting opportunities, on the other hand, are subject to the sporting calendar. For example the Aintree Grand National is one of the biggest events of the year for bookmakers, but it only takes place on a single day in April. Hence it is worthless to provide a recommendation for this even in May or June. The modelling of these temporal aspects is a central theme of this project.

1.3 Business Motivation

Paddy Power operates in a highly competitive industry in which profits are derived from thin margins across few sports. Daring marketing and humour are the vehicles of public engagement for the company, helping to sustain competitive advantage within a saturated market. Concealed by this outward persona is a primarily numbers driven enterprise. This project is motivated by the need to engage with customers at an individual level. We provide proof

of concept for the utility of recommender technology in the context of this business objective.

1.3.1 Cross-sell in the Online Sportsbook

Cross-selling here refers to the process of directing customers to new sports outside of those they would normally bet in. Broadening engagement across the Sportsbook is a key business objective for Paddy Power. We limit the scope of this project to cross-sell within the Online Sportsbook only. That is, we regard cross-selling customers into casino games, poker and so on, as a future extension of the work. With this in mind, what are the requisite features of a cross-sell engine for the Sportsbook?

Firstly, a personalised system is required. The Sportsbook already incorporates a recommendation feature dubbed 'Pimp My Bet' whereby customers are presented with five supplementary betting opportunities to augment an initial stake. Suggestions are made irrespective of customer betting histories and receive limited engagement. Our premise is that personalised recommendations will give Paddy Power the ability to deliver more targeted and hence more effective marketing.

Secondly, the algorithm should be sensitive to high value customers. They are characterised by a tendency to bet in unusual sports, not just big Mass Market events. The algorithm should tailor more unusual recommendations towards customers who have made quirky bets in the past. We will later refer to a customer's 'motivation for randomness'. Customers who adopt higher thresholds on this metric are ideal candidates for cross-sell.

1.4 Research Aim

This project aims to develop a data driven, algorithmic framework that can be incorporated by Paddy Power into their online platforms in order to provide recommendations to their customers. The following questions need to be addressed:

- How do we deal with the complex nature of the sporting calendar?
- How do we try to ensure the customer remains engaged with the system, and doesn't simply get bored of their recommendations?
- Can the data used provide insights into customer behaviour?
- Can a feasible solution to the cold-start problem associated with Collaborative Filtering be found?

1.5 Research Scope

One key point is that this project does not examine how the recommendations are to be delivered to the customer. Paddy Power communicate with their customers in a variety of ways from email to in-app messages, and the channel through which they talk to their customers can be extremely important in the efficacy of their message. The purpose of this project is to solve a problem in the field of data analytics with a strong grounding in business, however marketing specifics are outside the scope.

Furthermore, a choice had to be made on the level of data granularity to be examined. In Paddy Power's database, a sport superclass describes an all-encompassing category into which a sport fits. For example, Premier League, FA Cup, League of Ireland, Major League Soccer, the World Cup, etc are all be encapsulated by the soccer superclass. It was decided, with some minor exceptions to be discussed in Section 3.2, that this was the natural level of granularity make recommendations. There are several reasons. Firstly it simplifies the model to a point where the project could be completed over the course of three months, and secondly it significantly reduced the computational complexity of the model. As a result, the model will be able to provide a recommendation for "Rugby", but not "Ireland v England, 6 Nations". Paddy Power can overlay these lower level events onto the superclass levels revealed by the model.

1.6 Project Outline

The intial step was to take an in-depth look at the current literature describing recommender systems, and decide which model would best suit this project. Latent Factor models were chosen, as they are largely domain free and are suitable for implicit feedback models. The data available was examined and cumulative customer stake per event (i.e. the total amount of money a customer has placed on bets on a particular sport, over the lifetime of their Paddy Power account) was chosen as the implicit measure of a customer's preference for a given sport. A basic recommender model was then built based on existing literature, with initial testing carried out using real customer data. Once we were satisfied the algorithms were working as intended, the development of post-processing steps commenced, with two main goals in mind: firstly to handle the sporting calendar, and secondly to ensure the system remains interesting to the customer. Once these steps were implemented the recommender was essentially in its final form for the purposes of this project, so testing was carried out to ensure the time based recommendations were sensible. Finally a method to establish the relative accuracy of the temporally adjusted recommendations compared to the non-temporal recommendations was devised and implemented.

We believe that this project makes a novel contribution to the field of recommender systems by applying them to a completely new application that is currently not described in literature. The post-processing steps implemented were designed specifically for this purpose. We also present a data-driven method for making sensible recommendations to customers with only a very limited betting history, i.e. a potential solution to the cold-start problem in the context of betting. We also build on the current methods by which implicit recommender systems are evaluated by making the process a function of time, in order to ascertain the efficacy of the post-processing step that deals with the temporal problem. Overall we found our temporally filtered recommendations to be 26.47% more reliable than non-temporally filtered recommendations.

Chapter 2

Literature Review

2.1 Introduction

Recommender systems have been around for some time; Tapestry, the first collaborative filtering (CF) recommender, was developed by Goldberg *et al.* (1992) to reduce information overload in email communication. In the following decade researchers continued working on CF and explored alternatives in content-based filtering (CB). Not just an academic pursuit this research is driven by the increasing need for companies to guide their customers through a jungle of product offerings via personalised recommendations.

In October 2006 Netflix, Inc. sponsored a $1,000,000 prize awarded to any team who could improve their proprietary Cinematch recommender by 10% or better. Impressive advances in CF resulted from the Netflix Prize competition. This literature review will survey the work to date and identify novel challenges posed by an application in the gaming industry. First we justify adopting CF as our platform for developing the analytic core of a time-aware recommender system for Paddy Power.

2.2 Main Research Themes

The literature identifies CB and CF as the two key domains in recommender systems research (Park *et al.*, 2012). The former approach augments detailed user and item profiles with analytic techniques to compute recommendations. A major drawback of the CB approach is that significant human input is required to create the item profiles. For example, Pandora Media, Inc. directs

users to music they might enjoy based on how well their user profile synchronises with song profiles stored in the Music Genome Project database (Koren *et al.*, 2009a). Each profile requires an expert to score the associated song on several hundred musical attributes. On the other hand, CF recommenders automatically learn user-item interactions from the raw ratings data. Therefore, they are preferred in applications where incorporating domain knowledge is expensive (e.g. Amazon.com; a vast array of products types) or if its unclear what features should define the profiles.

Furthermore, CB struggles to capture more subtle aspects of the user-item interaction. Recommendations often fail to surprise because they are based on rigid profiles which capture a narrow slice of the customer's personality. Paddy Power requires a system that is capable of uncovering deeper insights into the personality of their customers. Specifically, recommendations should encourage the customer to explore sports or games that they may not discover autonomously. Hence, we will follow Koren *et al.* (2009a) and develop the analytic core of a CF recommender. What are the main research topics in this area?

2.2.1 Overview of the CF Literature

Koren *et al.* (2009b) introduce latent factor models in the context of the Netflix prize, using as their test-bed Netflix's extensive movie ratings dataset. Latent factors are a set of factors, possibly 20-100 in size, that characterise both a user and item. They are uncovered by the CF algorithm and cannot be directly translated into some metric by which humans would rate a film. The method of uncovering these latent factors is matrix factorisation. In its most basic form, matrix factorisation can be described as follow: we can form a matrix where each row represents a user and each item represents a column. The element a_{ij} represents how user i rated item j. Since a typical user will only have rated a small subset of the total number of items, many values will be missing from this matrix.

We wish to find two lower rank matrices which when multiplied together

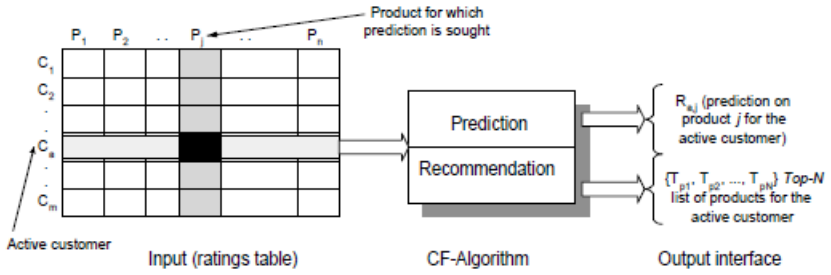


Figure 2.1: Sarwar *et al.* (2001), Overview of CF.

will approximate this ratings matrix, with a minimised error. We do not wish to recreate our original matrix exactly and steps can be taken in the algorithm to prevent overfitting. Once these two component matrices are found and multiplied together, the missing values in our initial ratings matrix will be "filled in". These values now represent recommendations that can be made to users. The most basic mathematical model is as follows:

$$\min_{p*,q*} \sum_{(u,i)\in\kappa} \left(r_{ui} - q_i^T p_u\right)^2. \tag{2.2.1}$$

The aim here is to minimise the squared error between our original ratings matrix, and a matrix calculated by taking the dot product of user and item factor vectors. This particular model would not be very effective, and a number of other considerations must be added to the model. As previously discussed, we only wish to estimate r_{ui} hence a parameter λ is added to prevent overfitting. This allows the algorithm to uncover unknown recommendations for users, rather than just recalculating the initial ratings matrix. Next, a bias is added to the model to capture user preferences, and deviation from the average. The idea here is to capture in the model whether an item is above or below average (b_i), or whether the particular user is critical or soft in their ratings (b_u). The model also takes into account the global average, μ.

Finally, a confidence factor c_{ui} is incorporated to weight ratings. In this paper, Koren is dealing with explicit feedback, i.e. a user explicitly and knowingly rates and item based on how much they enjoyed it. However, many

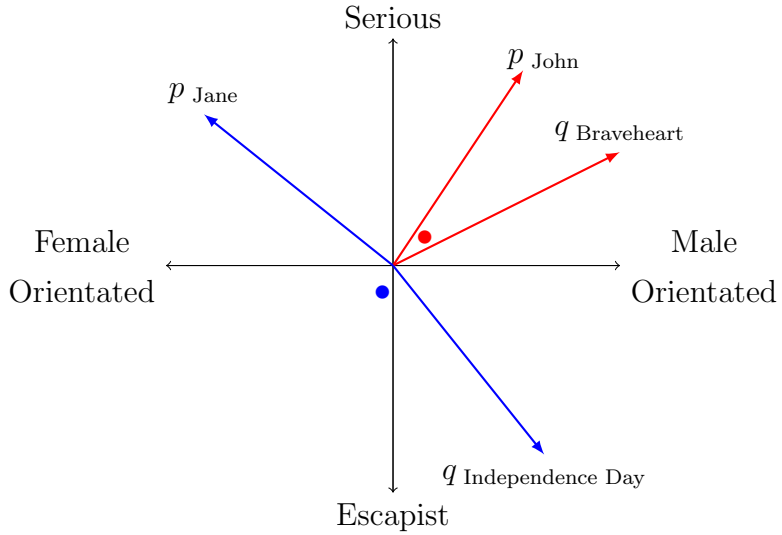


Figure 2.2: In this simplified schematic users and movies are mapped to a 2-D latent factor space. This toy model predicts John would like Braveheart but John and Jane would dislike Independence Day.

recommender systems including the model at the core of this project, are built around implicit feedback where user preferences are inferred from their autonomous activity and not from provided ratings. Implicit feedback might take the form of the number of times a user buys a certain product, watches a TV show, bets on a particular sport. The final explicit model presented by Koren *et al.* (2009b), complete with regularisation to prevent overfitting, bias terms and confidence levels is as follows:

$$\min_{p*,q*} \sum_{(u,i)\in\kappa} c_{ui}\left(r_{ui} - \mu - b_u - b_i - q_i^T p_u\right)^2 + \lambda\left(\| p_u \|^2 + \| q_i \|^2 + b_u^2 - b_i^2\right). \tag{2.2.2}$$

Figure 3.2 shows a simplified view of how the latent factor model might work in the context of a movie ratings application. In this example, John is a male who likes serious films, and Braveheart is a serious film aimed at males. Hence a dot product between these user and item factor vectors will predict

a large value for the predicted rating $\hat{r}_{\text{John, Braveheart}}$. Conversely, Jane is a female who likes serious films but Independence Day is an escapist film aimed at males. Since Jane and Independence Day do not share the same characteristics, $\hat{r}_{\text{Jane, Independence Day}}$ will be small, so we will probably not recommend this film to Jane. It must be noted that the latent factors presented here, gender orientation and degree of realism, are for descriptive purposes since a real model should admit possibly hundreds of factors. In addition, the latent factors uncovered by collaborative filtering, while reflecting genuine characteristics, are mathematical entities that evade direct interpretation.

The model can be optimised using techniques such as stochastic gradient descent, and is shown to be far more effective than models without biases, confidence measures and provisions against the overfitting (Koren *et al.*). It is a good basis for a recommender system suitable for Paddy Power.

As indicated, one downside of collaborative filtering is that the algorithm extracts fairly abstract rules for making its recommendations, which are not easily translatable into a form people can understand. As described, the recommendation scores are formed from the product of two vectors. It can be difficult to interpret the particular characteristics of an item or user that underlie a recommendation, and hence this method returns results which can be difficult to explain.

2.2.2 Implicit Feedback in Factor Models

A recommender system for Paddy Power must be adapted for implicit feedback as no explicit ratings are collected for products in the Sportsbook. If a customer bets on a certain sport repeatedly we can infer that they enjoy betting on that sport; if a customer bets more than their average stake on a given sport then we can infer an implicit preference, or at least a proclivity, towards the sport. Hence, we can easily infer what a customer likes from their implicit feedback but inferring their dislikes is challenging since no explicit negative feedback is available. Moreover, an absence of implicit preference does not necessarily imply that the customer dislikes a sport. For example, they may

not bet on a certain sport simply because they are unaware of it.

Hu *et al.* (2008) introduce the idea of a binary variable p_{ui} which is equal to 1 if a user consumes a certain product/service ($r_{ui} > 0$) and equal 0 otherwise ($r_{ui} = 0$). The analysis yields a vector of user-factors $x_u \in \mathbb{R}^f$ and item factors $y_i \in \mathbb{R}^f$ such that $\hat{p}_{ui} = x_u^T y_i$. Optimisation accounts for all user-item pairs in this implicit scheme, where for all $\{u, i\}$ pairs we have $p_{ui} = 0$ or 1. This is necessary mainly because implicit feedback cannot encode negative user experiences, but in generally it captures less signal than explicit ratings. Since the model is fully enumerated we do not need to include terms for bias but the confidence terms (c_{ui}) are requisite for accurate recommendations (Hu *et al.*, 2008). The model presented is:

$$\min_{x_*, y_*} \sum_{(u,i)} c_{ui} \left(p_{ui} - x_u^T y_i\right)^2 + \lambda \left(\sum_u \parallel x_u \parallel^2 + \sum_i \parallel y_i \parallel^2 \right). \tag{2.2.3}$$

A number of heuristics for calculating confidence (c_{ui}) are proposed. The author settles on $c_{ui} = 1 + \alpha \log(1 + r_{ui}/\epsilon)$, claiming this log scaling method performs well in experimental evaluation on a television services dataset (note ϵ, $\alpha \in \mathbb{R}$). The application domain informs our choice for what r_{ui} measures. A promising candidate for the Sportsbook is log scaling where r_{ui} is customer bet size. This paper stresses the point that when implementing a recommender system trained on implicit feedback, accuracy diminishes without both preferences (p_{ui}) and confidences (c_{ui}) in the model formulation. We first compute a binary value reflecting the customer's preference (or lack thereof) for an item, then a confidence score is assigned to that preference as described.

Since we have a term for every user-item pair, the problem quickly becomes computationally intractable if we use the same optimisation techniques as for explicit feedback models. An alternating-least-squares technique is presented, where in each iteration either the user-factor or item-factor vector is held constant. Taking appropriate derivatives of 3.5.1 and solving for the minima yield formulae for recomputing the user and item factor vectors,

$$x_u = \left(Y^T C^u Y + \lambda I\right)^{-1} Y^T C^u p(u) \tag{2.2.4}$$

$$y_i = \left(X^T C^i X + \lambda I\right)^{-1} X^T C^i p(i). \tag{2.2.5}$$

Where, Y is an $n \times f$ matrix storing the n item factor vectors and X is an $m \times f$ matrix storing the m user factor vectors. The $n \times n$ matrix C^u and the $m \times m$ matrix C^i store c_{ui} on their diagonals. Vectors $p(u) \in \mathbb{R}^n$ and $p(i) \in \mathbb{R}^m$ store the preferences expressed by users (on n items) and on items (by m users) respectively. We alternate between recomputing x_u and y_i over several iterations until they stabilise. Hu *et al.* (2008) exploit the algebraic structure of these equations to achieve linear runtime with respect to the size of the input data. As we will see later the algebraic structure also allows us to recover explainability which is a key requirement.

This paper is relevant to our task because it describes a factor model trained exclusively on implicit feedback. The model lends itself to explainable recommendations in line with Paddy Power's desire to avoid a black box. Finally, we can identify a clear gap for a novel contribution. Hu *et al.* (2008) present a static model, this suggests the opportunity to develop a factor approach for implicit feedback that is time-aware.

2.2.3 Content Filtering

Content-Based filtering (CB) is the third major research interest in the recommenders literature, see Pazzani and Billsus (2007) for a detailed discussion. Profiles are designed by domain experts to capture the salient characteristics of the items. These descriptions are analysed to find matches between users and items they may desire. If we take the example of a movie, the profile of a movie could contain information concerning it's genre, actors, duration and director etc. The music streaming service Pandora, Inc. implements a CB system where a music expert scores each and every song under a large number of different headings, the resulting profile is then used to recommend songs to users. On the other hand, collaborative filtering automatically uncovers the latent factors used to compute recommendations. Since (CB) requires a huge amount of human input in order to characterise each items it has been

identified as an unsuitable method for the purposes of this project.

2.3 Dealing with Temporal Dynamics

CF recommender systems analyse the past behaviour of customers to predict their future preferences. Of course, customer preferences are subject to natural drift over time as tastes evolve or as new products become available. Stefanidis *et al.* (2012) observe that many existing approaches fail to account for these temporal dynamics. This section reviews the early work and how more sophisticated solutions emerged from the Netflix Prize competition.

2.3.1 Time and Context-Aware Recommendations

Tang *et al.* (2003) state that few researchers considered temporal effects prior to 2003. Their paper examines whether the accuracy of a recommender improves when the age of user ratings are considered. A neighbourhood based model is trained on the entire MovieLens dataset[1] (including old movies) and its performance is compared to an identical model trained on only recent ratings. Accuracy improved when the older ratings were discarded. This was the first experimental indication that temporal effects are important to consider in models of human preference.

The intuition that older expressions of preference are less informative has led to various instance weighting methods. Decay schemes were developed to reduce the influence of older and presumably less reliable user feedback. Stefanidis *et al.* (2012) propose a framework in which older preferences are decayed using an exponential function of the form $f(t) = e^{-\lambda \Delta t}$. Where, $\Delta t = t - t_{ui}$ is the time interval between the current time and when user u expressed a preference for item i, λ controls the magnitude of the damping effect. The authors incorporate $f(t)$ into a time-aware neighbourhood model but no accuracy gains are observed with respect to the MovieLens dataset. Koren (2010) runs a similar model on the more extensive Netflix dataset. He

[1] Available from http://grouplens.org/datasets/movielens/.

finds that accuracy improves as $\lambda \to 0$, reaching a maximum when no decay scheme is used at all. Therefore, the advantage of under weighting noisy older preferences is outweighed by the drawback of losing signal latent within this data.

Stefanidis *et al.* (2012) experiment with a sliding window model where only a subset of recent preferences are active. Assuming this subset faithfully represents the current interests of a user, older less reliable ratings are discarded. The approach improves accuracy with diminishing returns when smaller windows include too few instances.

The paper concludes by demonstrating how context aware models improve prediction accuracy. In this approach, only those preferences expressed by a user within the same temporal context as their current query are used to compute recommendations. To illustrate consider a user searching for eBooks during the weekend. This user's weekend reading may differ from their weekday reading. They may prefer novels at the weekend but educational material during the working/study week. Training on preferences expressed at weekends is hypothesised to yield more accurate recommendations. Baltrunas (2009) constructs micro profiles to represent users in different temporal contexts. For example, a user could be represented by micro profiles for the morning, weekend, summer context etc. Recommendations are then computed using the appropriate micro profile for the current query context. Context is clearly an important factor in gaming due to the highly seasonal and transient nature of the sporting calendar.

Under weighting past actions or including only a subset of ratings is discouraged by Koren (2010). One loses too much signal to understand the long term trends in user behaviour. He presents time-aware factor and neighbourhood models that exploit the full set of user feedback.

2.3.2 Nearest-Neighbour Methods

To this point the focus has been on CF, but nearest-neighbour (NN) methods are also valid when designing a recommender system. NN methods can be

either item-based or user-based. With an item-based system, when a customer buys a particular product, the system will then recommend similar products to that customer. With a user-based system, when one customer buys a product, that product will then be recommended to other customers who are deemed to be similar to the first customer, this could be based on factors such as age, gender, and previous purchasing history. NN recommender systems have advantages in that they are generally quite simple and the recommendations are easily explained. It must be decided to to measure similarity between users and items respectively.

Koren (2008a) propose a hybrid CF/NN model, based on the respective strengths and weaknesses of NN and CF methods, stated as follows:

> "... neighbourhood and latent factor approaches address quite different levels of structure in the data, so none of them is optimal on its own [3]. Neighbourhood models are most effective at detecting very localized relationships. They rely on a few significant neighbourhood relations, often ignoring the vast majority of ratings by a user. Consequently, these methods are unable to capture the totality of weak signals encompassed in all of a users ratings. Latent factor models are generally effective at estimating overall structure that relates simultaneously to most or all items. However, these models are poor at detecting strong associations among a small set of closely related items, precisely where neighbourhood models do best."

Vinagre (2011) and Sarwar *et al.* (2001) discuss a hybrid CF/NN model and introduce a number of ways by which similarity between items and users can be measured. With CF, a row of our rating matrix is all the ratings provided by one particular user. If each row of the matrix is considered a vector, one measure is to take the cosine similarity between those vectors in the appropriate dimensional space. However an issue here is that this measure is un-normalised; one user may regard 3 stars out of 5 as a negative review whereas another less critical user may regard it as a positive review. Such

nuances are lost in the cosine measure. A different interpretation of the similarity measure which accounts for this problem is the Pearson Correlation, which measures the similarity between users u and v, where $\bar{r}_u$ and $\bar{r}_v$ are the average ratings given by the respective users and I_{uv} is the set of items rated by both users:

$$\text{sim}(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}. \tag{2.3.1}$$

Similarly there is a Pearson Correlation that measures the similarity between two items, i and j, where $\bar{r}_i$ and $\bar{r}_j$ are the average ratings given to the respective items and U_{ij} is the set of users that rated both items:

$$\text{sim}(i,j) = \frac{\sum_{i \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{vj} - \bar{r}_j)}{\sqrt{\sum_{i \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{i \in U_{ij}} (r_{vj} - \bar{r}_j)^2}}. \tag{2.3.2}$$

Average item/user ratings have a normalising effect on the calculation thus minimising the problem whereby users interpret the rating scale in different ways. These two measures show how, given a user ratings matrix typically used for latent factor methods, we can find a similarity measure between respective users and items which paves the way for a hybrid algorithm.

Sarwar *et al.* (2000) propose a further extension to the nearest-neighbour method by taking the entire data set and applying a clustering algorithm as a preprocessing step. Each cluster will contain a far smaller amount of data than the overall dataset which can result in better computational performance when computing Pearson Correlation. This can be expensive for large and sparse data sets. However, the results returned may not be as accurate as those obtained by the Pearson correlations. One thing that is clear from the literature review is that recommender systems must be tuned in both parameters and design, depending on their intended use. This will be one of the major challenges of this project.

It will be interesting to compare the accuracy of neighbourhood and factor models in our gaming application. Koren shows that the latter achieve only slightly better RMSE relative to their neighbourhood counterparts when

applied to the Netflix data. However, people will express a wider spectrum of behaviours when gambling compared to when watching movies, a much less visceral activity. The movie watcher simply wants to be entertained whereas the gambler may be subject to a large array of motivations. Perhaps neighbourhood models can more effectively capture the local structure within the customer base; the thrill seeker, the risk taker, the professional.

2.3.3 Combining Neighbourhood and Factor Models

Koren (2010) and Koren (2008b) are two notable papers in the evolution of time-aware CF. The latter derives a combined neighbourhood and factor model, while the former extends the static models to track temporal artefacts in the data.

Koren (2008a) is motivated by the need to unify neighbourhood and factor models. In isolation these two approaches to CF cannot simultaneously describe local and global structure in the data. Furthermore, factor models are black boxes because user and item profiles are mapped to abstract factor vectors. Neighbourhood models give explainable recommendations because user behaviour is understood directly in terms of the items that they interact with. This paper is important because a combined model is developed which integrates implicit feedback and recovers explainability. The paper is relevant because Paddy Power have expressed their desire for explainable recommendations.

Building on the model described in 2.3.2, Koren (2008b) goes on to describe an item-orientated neighbourhood model. Predicted ratings are computed based on the similarity of the un-rated item and items already rated by the user. Thus, recommendations are easily explained since users will be familiar with items they previously rated. A novel prediction rule is proposed,

$$\hat{r}_{ui} = b_{ui} + \left|R^k(i;u)\right|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + \left|N^k(i;u)\right|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}. \tag{2.3.3}$$

Where, $\hat{r}_{ui}$ is the predicted rating, b_{ui} is the baseline term, $R^k(i;u)$ is the subset of items rated by user u most similar to i and w_{ij} denotes the similarity between items. The final term, $\sum_{j \in N^k(i;u)} c_{ij}$, accounts for implicit feedback where $N^k(i;u)$ is the subset of items most similar to i for which u provides implicit feedback. Equation 2.3.3 is novel because w_{ij} are global item similarities that do not depend on the user and we exploit the full set of user ratings. Weights ensure offsets to baseline estimates $(r_{uj} - b_{uj})$ are significant if item j is similar to i. The multiplicative factors $\left|R^k(i;u)\right|^{-\frac{1}{2}}$ and $\left|N^k(i;u)\right|^{-\frac{1}{2}}$ prevent the model from . Gradient descent is used to minimise the least squares problem suggested by (2.3.3), to learn weights and the baselines.

Next, the factor model is modified so that users are represented by the items they rate instead of an abstract factor vector p_u. The new prediction rule is given by,

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right). \quad (2.3.4)$$

This formulation recovers explainability because user behaviour is understood in terms of items rated and not in terms of abstract factors. The final term, $|N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j$, accounts for implicit feedback. The key point is that (2.3.4) can be easily integrated with our neighbourhood formulation (2.3.3). In practice the best accuracy is attained when we concede explainability and replace $|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j$ in (2.3.4) with a factor vector p_u giving the final combined prediction rule:

$$\begin{aligned} \hat{r}_{ui} = b_{ui} + q_i^T & \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \\ & + \left|R^k(i;u)\right|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + \left|N^k(i;u)\right|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}. \quad (2.3.5) \end{aligned}$$

2.3.4 Temporal Dynamics in Neighbourhood and Factor Models

Koren (2010) is motivated by the fact that users change their taste over time and items drift in and out of popularity. Keeping track of these time changing effects is difficult due to the diversity of products on offer and because the tastes of each user evolve uniquely. The author states that an adequate model must consider the full ratings history of users and so decay schemes will not suffice. Temporary effects can then be distilled from the long term patterns. With this in mind the author constructs time-aware versions of the models described in subsection 2.3.3.

The factor model "lends itself well to an adequate modelling of temporal effects". Firstly, we can parametrise the biases in t such that $b_{ui}(t) = \mu + b_u(t) + b_i(t)$. Drifting user biases reflect the tendency of users to become more or less critical over time. Similarly, variations in item popularity are absorbed in the time drifting item biases. The functions $b_u(t)$ and $b_i(t)$ should be adapted to the time scales of the corresponding concept drifts. For example, the author proposes a linear function to track gradually changing user biases. This is augmented with a term that absorbs daily variations. Hence, the model accounts for slowly changing patterns but is also sensitive to the more transient variations natural in human preference. Item popularity drifts more gradually so it suffices to split ratings into bins and then fit $b_i(t)$ to the binned data.

The preceding functional forms for $b_u(t)$ and $b_i(t)$ work well when applied to the Netflix dataset (Koren, 2010). However, we expect people to adopt a very different mindset when gambling as opposed to when browsing for movies. We hypothesise that user biases will be more stable because people may be hesitant to bet on sports they are unfamiliar with. Furthermore, major sporting events will cause the popularity of different sports to shift dramatically as the season progresses; a more robust model for the item biases will be required. Thus, we have identified a gap in the context of modelling temporal dynamics in the gaming industry that needs to be addressed. To our knowledge there is no literature on the subject of recommender systems in the gaming industry.

Koren (2010) modifies the factor model by parametrising $p_u(t)$ giving,

$$\hat{r}_{ui}(t) = \mu + b_u(t) + b_i(t) + q_i^T \left(p_u(t) + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right). \tag{2.3.6}$$

Notice that the item factor vector is not a function of time because item characteristics are fixed e.g. a movie does not become more dramatic. The user factor vector does change in time e.g. a user may become more interested in dramatic movies. As usual the RMSE between $r_{ui}(t)$ and $\hat{r}_{ui}(t)$ is minimised to learn parameters. The neighbourhood model (see 2.3.3) is modified as follows:

$$\hat{r}_{ui} = b_{ui}(t) + |R(u)|^{-\frac{1}{2}} \sum_{(j,t_j) \in R(u)} e^{-\beta_u |t - t_j|} \left((r_{uj} - b_{uj}) w_{ij} + c_{ij} \right). \tag{2.3.7}$$

This formulation reduces the influence of items j that were rated further back in time. The exponential decay used here is subtly different from the decay schemes discussed in section 2.3.1 in that signal is preserved from all past ratings. The main point is that neighbourhood and factor models are easily modified to account for temporal dynamics. In doing so we achieve better accuracy without resorting to overly complicated models.

Time-aware models are not just important for improving accuracy. They give us deeper insights into the business. For example, the factor model (2.3.6) was used to explain why there was a sudden jump in the average ratings given by Netflix customers during 2004. Two hypotheses were tested: (1) people have becomes less critical since pre 2004 or (2) improved recommendations have more effectively directed users to movies. Now, (1) would be associated with a shift in the baseline components of the model whereas (2) would arise from the interaction part of the model. By plotting these two components separately Koren was able to show that the shift was due to (2), that is, improvements to Cinematch in 2004 led to better recommendations and hence higher ratings. We believe Paddy Power could benefit from similar insights.

By reviewing the work of Koren and his collaborators on the Netflix Prize competition we have found the envelope in CF research. This dissertation will extend the reach of CF to an application in the gaming industry.

2.4 Adapting the Models

Dunning and Friedman (2014) propose two interesting ideas that support the objectives laid out by Paddy Power. The first is the idea of Dithering, whereby some random items are included with the base output from a recommender system. Dithering provisions against users becoming bored with unchanging recommendations. People will likely pay attention to only the first few recommendations, so it is vital that they change over time to remain interesting. Recommendations deemed irrelevant or boring will simply be ignored. Through the incorporation of dithering, a far greater degree of engagement is achieved which increases the value of the system in the long term, though possibly at the expense of a short term hit in accuracy. Dunning and Friedman (2014) suggest implementing dithering by taking the recommender scores (where, the highest score implies the most relevant item), applying a log scaling, and then adding some random noise element. The modified recommendations list, again sorted in descending order of relevance, is presented to the user. The effect of the random noise is to lift results that would otherwise be quite low down the list up to closer to the top, while still retaining the highest scoring items towards the top of the list. The authors of this paper emphasize that dynamic recommendations are more likely to engage the interest of the user, but this method would also closely fit with the requirement of Paddy Power to return unexpected results to those who make unusual bets.

The second method to be investigated is Anti-Flood. The idea here is to partition the set of recommendable items into categories, and then set a maximum number of items from each category that the system can return. We thus avoid bombarding the user with the same type of item over and over again, which would quickly negate the impact of the recommendations. A simple example of how this could be implemented for Paddy Power is as

follows: if a football match is recommended to a customer, then a second football match would be allowed but a third football match would be penalised by demotion to a position lower down in the recommendations list. This would cause the list to be much more diverse than otherwise, and would again fit the requirement that the recommender system return unexpected results.

Another issue that will be critical to address properly is that of scalability. Sarwar *et al.* (2000) discuss how recommender systems are being used more and more in e-commerce, and describe the stress that is placed on these systems by the ever increasing amount of customer data available. As of March 2014, Paddy Power had roughly 1.6 million customers active across their online services, a number double that of 2 years previous. This suggests a huge growth in the number of people moving to online betting and presages the need for intelligent data processing in any predictive model with a view to the entire customer base. It's suggested by Sarwar *et al.* that dimensionality reduction is a strong candidate for improving the performance of recommender systems. We'll see later how scalability can be achieved for the Sportsbook recommender without the need for dimensionality reduction.

Chapter 3

Methodology

An experimental methodology was used to validate the utility of a recommender system in the context of the gaming industry. This chapter describes the procedures used during model development so that the work is replicable in similar applications.

3.1 Software

As outlined in the literature review, extensive research into the current state of recommender systems and was carried out, with the most promising methods identified. Our initial model was based on the work of Hu *et al.* (2008), wherein he presented a latent factor model for implicit feedback data learned by alternating least squares (ALS). The paper offers no insight into how exactly they implemented the algorithm, so we decided to use R to create our model. There were numerous reasons why R was suitable for implementing the model:

1. The algorithm admits a great deal of linear algebra. R has build-in data structures such as the matrix and data frame that are optimised for operations on large matrices and storing mixed type data respectively. It would have been necessary to write custom methods to carry out the linear algebra operations had we decided to code in Java.

2. R is adept at reading and storing very large data files. Once again, in another language it would have been necessary to write custom code to read in the large data files used in this project, the largest was over 9

GB in size. With the data.table package in R file reading is essentially automated and extremely fast. Furthermore the data.table packaged provides fast methods for subsetting data, so that for example it was possible to quickly extract the information on a single customer from one large dataset.

3. R is open source and widely used in Paddy Power. Therefore while we could have used a proprietary software such as MATLAB, which can also carry out the required linear algebra operations, it's not used in Paddy Power. R therefore harmonised with the business context of the project.

4. Finally, there are many third party packages that are available for R, some of which were invaluable in the course of developing the model.

Despite the initial optimism it became apparent during development that the linear algebra would be intractable in R. Hence it was decided to use a more specialised language to run the core factor algorithm. The Numpy and Scipy packages in Python provide powerful routines for storing and multiplying large sparse matrices. In particular, it was possible to store indices of the matrix non-zeros outside of the loops and avoid unnecessary 0×0 FLOPS during the ALS iterations. This 'masking' approach was found to be more effective that storing the matrices in sparse form (due to element access overhead).[1] All the pre-processing and post-processing steps were still carried out in R but when the program reached the Alternation Least Squares calculation, it passed over to Python. Data was exchanged between R and Python using MATLAB MAT-files. While using two different programming languages added a layer of complication to the project, it facilitated drastically faster execution times.

3.2 Data Granularity

The level of data granularity modelled will determine what the output from the recommender system looks like. There are various levels of data granularity

[1] Inspired by mrec: https://github.com/mendeley/mrec

that are accessible in the Paddy Power database. The highest level is the sport superclass whereby each sport is assigned a numeric identifier. For example, all horse racing events fall under the superclass ID 110. Below this level are the more specific events in a sport. To continue with racing as an example, a specific event may be Cheltenham, Aintree, Galway, or any other race meeting. The next level of granularity would be a specific race within an event, for example the Cheltenham Gold Cup is one particular race on the final day of the Cheltenham festival. A further lower level of granularity is an individual bet on a single race, for example it's possible to place a bet on who will win the race, who will place within the top 10, who will or won't finish the race, etc. There are often dozens of unique individual bets that can be placed on a given event.

If this lowest level of granularity was used then it would obviously allow us to output individual bets as recommendations to customers. However there are two main disadvantages to this. The first, and less important reason is that it would dilute the information being input into the algorithm over a much larger number of items. This would weaken the signals that the collaborative filtering process is able to find in the data and would generate weak recommendations. The second and more important reason is that there would be an absolutely vast number of items that would have to be considered, and this would cause the computational complexity of the process to sky rocket.

Using the sport's superclass level offered a high enough degree of accuracy, while remaining computationally feasible, however exceptions were made for the most popular sports in the Sportsbook. For Racing, Football, Tennis and Golf, all the major events in the sporting calendar were extracted and treated as their own superclass. For example Racing was broken down into the following set: {Cheltenham, Aintree, Royal Ascot, Punchestown, Fairyhouse, Galway, All Other Racing}. Similarly the golf majors and tennis grand slams were treated separately, along with the Premier League, Champions League, World Cup and FA Cup in football. Thus a good trade-off between accuracy and computational efficiency was achieved while giving the model a perspective on the Mass Market and niche sports. Lastly, following consultations with the

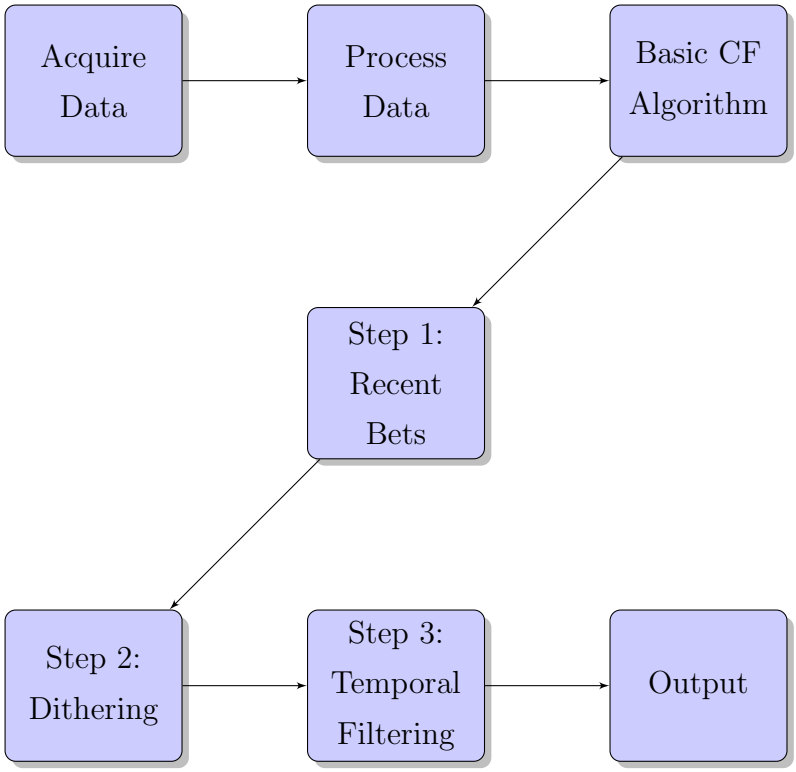


Figure 3.1: Illustration of the program flow.

Customer Intelligence Team we decided that the low turnover events should be aggregated so that for example $\text{Hockey}_{\text{aggr}}$ subsumes Rink Hockey, European Ice Hockey and American Ice Hockey.

3.3 Algorithm

Figure 3.1 outlines the 7 discrete steps from gathering data to providing output. Those steps will now be described in detail.

3.3.1 Data Acquisition

Paddy Power provided full access to their data warehouse, which proved to be an incredibly rich source of clean data. It was decided to obtain data as

customer_nk	event_class_sk	event_type	stake	date
PPO_123456	274	Premier League	10	01/01/2001
PPO_123456	274	Champ. League	10	02/02/2002
PPO_234567	110	Cheltenham	20	03/03/2003
PPO_345678	28	Wimbledon	15	04/04/2004

Figure 3.2: Format of the MMSql data pull—Sample Data.

follows: for each customer in question we would obtain each individual bet they had made, recording which sport the bet was made on, the particular event the customer had bet on, and their stake. Each row of the dataset represents an individual bet made by a single customer. MMSql was used to query the Paddy Power database. The data looks as follows, note this data is not representative of any actual customers but merely for demonstration:

The *event_class_sk* column represents the sport superclass, for example 110 is Horse Racing and 274 is Football. The *event_class_sk* column allowed us to extract out the major sporting events which did not have a unique superclass ID. As previously mentioned the cumulative (lifetime) stake on a given event was used as the implicit measure of the preference a customer has for that event. Finally it was crucial to include the date of the bet as it was needed to incorporate the temporal filters. In our initial testing a very small dataset comprising 10 customers was used. Once we were confident the algorithm was working as intended, a much larger dataset of roughly 7,500 customers was used. This dataset contained over 7 million individual bets and was over 300 MB in size. However this was still only a tiny fraction of the overall active customer base.

3.3.2 Data Pre-Processing

A significant advantage throughout this project was the availability of extremely clean data. Some of the event superclass IDs were undefined but these entries were easily removed. The *stake*, *event_type* and *date* entries were all fully in place with no missing values. An aim of this project is to better understand cross-sell within the Online Sportsbook, so all events other than

sports were also removed from the dataset (for example, online poker was not considered, as either input or output) with the exception of Lotto.

Once these preliminary steps had been taken, the total stakes a customer had placed within each individual superclass were summed. For example, if a customer had made 10 bets on Football and 10 bets on Racing, the total stake that customer had placed on Football and on Racing were calculated and stored. This total stake quantifies a measure of the preference (r_{ui}) a customer expresses for a sport. The most recent date on which a customer had bet on each sport was also recorded, as this was required for one of the post-processing steps. Finally, the data was formed into a large matrix where rows represent events and columns correspond to unique customers. If the customer had placed bets on a given event, their lifetime stake was stored by summing all bets into the appropriate matrix cell.

3.3.3 Basic Collaborative Filtering

Calculating Recommendations

As mentioned previously our collaborative filtering model is based on the work of Hu *et al.* (2008). The matrix from the previous step was transformed into preference matrices contained only binary values and real valued confidence matrices. The former indicate sports in which customers have placed bets, the latter store a 'confidence' in these bets i.e. how confident we are that the bets reflect a genuine affinity. The following equation computes our confidence for the preference customer u expresses for sport i, with $\alpha = 40$ and $\epsilon = 1$ x 10^{-8} (based of values in the paper), and r_{ui} the corresponding lifetime stake:

$$c_{ui} = 1 + \alpha \log(1 + r_{ui}/\epsilon) \tag{3.3.1}$$

These are the key elements required to calculate recommendations based on implicit feedback. Figure 3.3 shows a very simple input matrix of 3 users and 4 items, and the corresponding preference and confidence matrices.

As outlined in the literature review, recommendations can be generated

	u_1	u_2	u_3
i_1	20	0	11
i_2	0	6	8
i_3	9	0	0
i_4	18	0	15

	u_1	u_2	u_3
i_1	1	0	1
i_2	0	1	1
i_3	1	0	0
i_4	1	0	1

	u_1	u_2	u_3
i_1	3.65	0	1.91
i_2	0	1.25	1.79
i_3	1.96	0	0
i_4	3.02	0	2.71

Figure 3.3: Raw input (l), preference (m) and confidence (r) matrices.

once the user and item factor vectors stabilise. Optimisation proceeds as follows. The matrix Y (containing the item factor vectors) is initialised with random numbers between 0 and 1. The user confidence matrices C^u and the user preference vectors $p(u)$ are first determined for each customer. Similarly, the item confidence matrices C^i and the item preference vectors $p(i)$ are determined for each sport. Next, $x_u : u \in \{\text{Customers}\}$ are calculated and stored in a matrix X. X is then passed to equation 2.2.5 yielding the updated Y, which is passed back into 2.2.4. This process repeats until the matrices stabilise. Ten to fifteen iterations of ALS gives an acceptable trade off between execution time and accuracy. The converged X and Y are finally multiplied together to yield a matrix where entry ij represents the predicted preference of user i for item j .

Calculating Explanations

Hu *et al.* (2008) describe how their model supports the generation of rationales to supplement each recommendation. The authors argue that users are more likely to engage with a recommendation when it is explained. For example, in the Sportsbook a customer might be presented with a suggestion as follows:

> "Based on your previous wagers on Basketball, American Football and Baseball, we think you would like to bet on Hockey."

Hockey seems a reasonable suggestion in light of the other American Sports. As a stand alone recommendation however, Hockey could appear arbitrary and cause the customer to lose confidence in the product.

Explainable recommendations are achieved here by exploiting the mathematical structure of the model. Contrast Latent Factor algorithms, which are generally black boxes, with their item-based neighbourhood counterparts: the former abstract out user and item characteristics to un-interpretable factor vectors, whereas the latter compute recommendations in terms of previously rated items, similar to the target item. Clearly then recommendations derived from neighbourhood models lend themselves more naturally to interpretation. It is fortuitous that the Factor algorithm used to build our model recovers the explanatory power of these neighbourhood schemes. How does this work?

When the algorithm converges the predicted preference of user u for product i ($\hat{p}_{ui}$) can be reconstructed via a linear sum:

$$\hat{p}_{ui} \equiv \sum_{j:r_{uj}>0} s_{ij}^{u} c_{uj} \tag{3.3.2}$$

Where, s_{ij}^{u} are the similarities between i and all other items j that the user has rated (i.e. sports staked) and c_{uj} is our confidence in the importance of item j to the user. Equation 3.3.2 is derived by subbing the converged user factor vectors $x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$ into $\hat{p}_{ui} = y_i^T x_u$. Recommendations are thereby interpreted in terms of previously rated items so we recover the interpretative power of item-based neighbourhood models. Our research has revealed the full potential of 3.3.2 when it is applied to cross-sell, an application not considered by Hu *et al.* (2008) and our point of departure.

A detailed discussion of cross-sell is reserved for Chapter 4. Firstly, let us consider how to implement 3.3.2 in a computationally efficient manner. We found the naive approach was prohibitively slow, taking much longer than the optimisation process itself. In the naive implementation, explanations were computed for even the weakest recommendations. However, it seemed reasonable to only compute explanations for the strongest 30 as any weaker recommendation would never be presented to a customer (the number 30 has a significance discussed in Section 3.3.4). In Python the code first sorts a customer's recommended items in decreasing order of strength, then explanations for their top 30 only are computed. Storage is O($n \times 30 \times 3$), where n is the

number of customers and only the strongest three explanations are stored for each recommendation.

3.3.4 Post Processing 1: Principled Randomisation

One of the key objectives of this recommender system for the Sportsbook is to increase customer engagement with Paddy Power. In such a competitive industry, it's vital that the customer is given the best experience possible when using the online platforms, as they can easily move to a competitor who provide an essentially identical product. For this reason, an important business objective is to provide interesting and unexpected results to the customer, based on the thought that simply providing the same recommendations over and over will quickly become boring.

As described in chapter 2 and by Dunning and Friedman (2014), Dithering is a process whereby a random score can be added or subtracted from a base recommendation score to alter the order of the recommendations. In testing our base recommender we decided that in it's basic form this was not a sufficiently robust mechanism by which to provide interesting results as it could result in a customer being recommended an event that they have shown absolutely no proclivity for in the past. Hence the novel idea of principled dithering was formed, whereby the extent to which the recommendations for a customer are randomised is based on a 'dithering score', i.e. a customer's motivation for randomness, that is learned from their betting history. Paddy Power are always striving to identify customers they would consider potentially high value. Those who bet across unusual events are much more valuable that the occasional Mass Market punters. For each customer we compute their dithering score, a relative value which captures how open they are to betting on more esoteric or unusual events.

The score is calculated as follows:

1. First, calculate the 'unusualness' of each sport. This is simply the inverse of the total amount staked on that particular sport by all customers in training set. So a sport like Horse Racing will attain a very low

unusualness value, and a sport like Baseball will take on a much higher unusualness score. This information is saved to a vector (v^1).

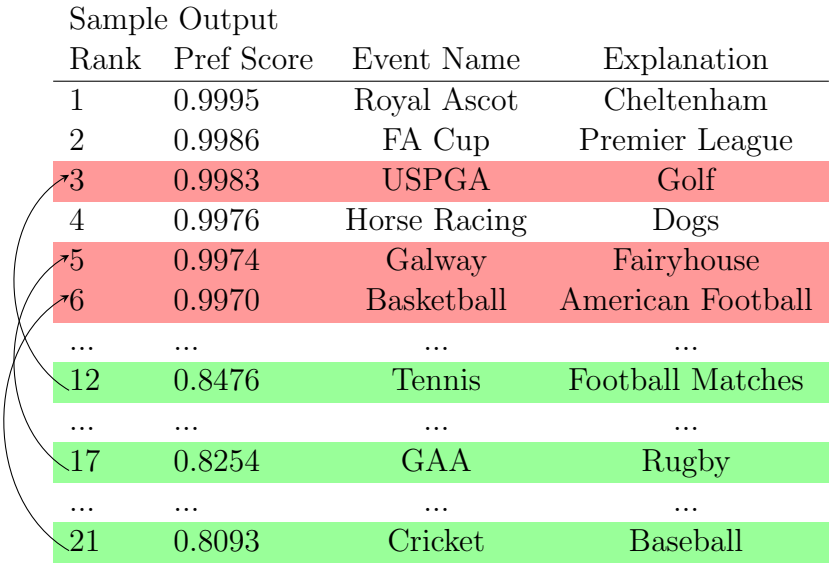
2. Next, for a particular customer, find all the events they've bet on, and record their total cumulative stake in these events. Map the stakes to the unit interval (hence "normalise"), so the event in which they've placed the greatest cumulative stake adopts a value of 1. Save to a vector for each customer (v_u^2).

3. The value obtained from taking the dot product between v^1 and v_u^2 is the dithering score for customer u. It's usually quite a small number, often $O(10^{-4})$ or $O(10^{-5})$. All these values are once again mapped to the unit interval, so the largest dithering score becomes 1.

4. The recommender system is designed to return the top 10 recommendations for a given customer. It has been decided that the top 2 results are exempt from dithering. The dithering score calculated in the previous step is then used to calculate, between the 3rd and 10th recommendation, how many results will be randomly replaced.

For example if a customer u has made an equal number of bets on 5 different events, this distribution would be represented by a vector as follows: $v_u^2 = (0.2, 0.2, 0.2, 0.2, 0.2)$, with each value corresponding to the particular sport they'd bet on. A second vector containing the unusualness score of those events would be pre-computed, and their dot product would yield the customer's dithering score.

So the dithering score quantifies a measure of how adventurous a customer appears to be. Those who place the vast majority of their bets on Horse Racing or Soccer events receive a very low score, while a customer who places an equal number of bets on many events, including some more esoteric events, will receive a high score. The following two facts must be noted when considering this randomised personalisation: the system is set up to generate 30 recommendations for the customer in total, but it's only designed to present the top 10 recommendations to the customer. The dithering score determines



Sample Output

Rank	Pref Score	Event Name	Explanation
1	0.9995	Royal Ascot	Cheltenham
2	0.9986	FA Cup	Premier League
3	0.9983	USPGA	Golf
4	0.9976	Horse Racing	Dogs
5	0.9974	Galway	Fairyhouse
6	0.9970	Basketball	American Football
...	...	...	...
12	0.8476	Tennis	Football Matches
...	...	...	...
17	0.8254	GAA	Rugby
...	...	...	...
21	0.8093	Cricket	Baseball

Figure 3.4: Illustration of Dithering.

two things: The first is how many entries from the top 10 recommendations are to be randomised, and the second is how far down the total list of 30 recommendations their replacements can be drawn from. Consider sample output for a given customer:

This customer's dithering score has determined that three results are to be exchanged. These rows are selected at random and rows 3, 5 and 6 are chosen (highlighted in red). Similarly, based on the customers dithering score, it is determined that the replacements for these rows can come from down to the 24^{th} row of the total recommendation list (replacements are never drawn beyond than the 30^{th} row). Three results are then selected at random between the 11^{th} and 24^{th} rows, which are highlighted in green in the abbreviated table below. The three red rows are then swapped for the three green rows, and the results are returned to the customer. Principled dithering ensures that the recommendations remain fresh, and that the swapped sports still reflect the interests of the customer so something should always seem appealing to them.

Finally, dithering is sometimes motivated by the need to gather data on previously blank areas of the rating matrix. For example, Netflix might give

some random recommendations so that the user can say "Not interested". Of course, recommenders based on purely implicit feedback cannot incorporate this technique so it is not considered further.

3.3.5 Post Processing 2: Temporal Filtering

It soon became obvious that adapting the linear algebra of our base recommender system to incorporate all of the temporal dynamics at play in the Sportsbook would be infeasible. Subsection 2.3.4 describes the successful efforts of Koren (2010) to subsume temporal effects into his model for a movie streaming recommender. However, the kind of temporal stresses therein are much less severe—movies are always available but sporting events are subject to seasonality. Our solution was to use filters in post processing to multiply the base preferences scores generated by the algorithm.

It was not possible to model the seasonality of all sports with a one size fits all filter. Rather several filter topologies were constructed to capture sport specific seasonality. Figure 5.5 in the Appendix illustrates the segmentation of each sport into a category that best describes its seasonality. Some large Mass Market events take place one per year (Annual MM), some sports are available throughout the year but may become more popular towards the weekend or in a given month (Weekly Filters), other sports have occasional big events (Sporadic Filters) and many sports cannot be modelled temporally (Non Temporal).

A data frame (called 'activation') was created in R to store the filters for each sport. Columns were labelled with sport names and rows with dates from August 2008 to December 2014 (filters have a view to future events).

- Annual MM Filters: were constructed by pulling the dates when these events occurred in the past three years from the in house Event Units model in Paddy Power. This model was devised to predict the big days in various important sports. That is, days where many First Time Stakers are expected to set up accounts. Dates for some events like the tennis majors were not included in this model and were added manually. Using

these dates as a lookup for the turnover (pulled from the warehouse) in the corresponding Superclass (e.g. Tennis or Golf etc.), the model learned a filter for each event based on fluctuations in the turnover as the event played out in previous years. Therefore, the filters for Golf account for the lull on the second day of a Major and the filters for a Tennis Major pick up lulls on the Ladies days and peaks on semi-final/final days. Filters values are mapped to the interval $[0, 2]$.

- Weekly Filters: Some sports such as the GAA or Basketball generate revenue throughout the year. Turnover in these sports tends to peak at the weekend. In addition, turnover peaks at certain times during the season and bottoms out during the off season. GAA betting peaks in late summer but is almost non-existent during January say. To model interest drift in these weekly and seasonal events we computed the average turnover on each calendar day and mapped them to $[0, 2]$. The monthly average turnover was also mapped to $[0, 2]$ and was used to scale the weekly filter values. Thus these filters model interest drift at the granularity of day and respect monthly shifts in a sport's popularity. They are projected onto future dates.

- Sporadic Filters: Events like a big boxing match do not occur at regular intervals, rather take place sporadically over the course of the season. The Event Units model nonetheless predicts when these big events will take place for several sports. We use the Event Units data to identify such days and where possible (possible for Horse Racing, Soccer and Golf) overlay this model's prediction of the importance of the event. If no data is available for the event's importance then a filter on the interval $[0, 2]$ is assigned.

In all cases an exponential build-up to the start date of each event is appended to the filters. The effect of this build-up profile is to shuffle up a recommendation for an event as it draws near.

To implement the filters a lookup is performed on the activation data frame which stores the filters. The lookup extracts the row of activation cor-

responding to the desired date (date on which we want to generate recommendations). Then for each sport that is modelled temporally, its preference score is multiplied by the filter value in the corresponding column of activation. For sports that we cannot model temporally the preference scores are left unchanged from their base values. Notice that the filters are bias free as sports can be bumped up ($[1, 2]$) or down ($[0, 1]$) depending on the filter value.

3.4 Efficiency

The largest dataset the program was tested on consisted of data from some 245,000 customers, consisting of nearly 150m individual bets. However this is only a fraction of the entire customer base, over 800,000 customers have been active on their Paddy Power online account in the last 90 days. Due to the huge amount of users and data, computational efficiency was of paramount importance, as if it took an inordinate amount of time to generate recommendations they could be out of date by the time they reached the customer. As discussed in section 3.1 this motivated passing the linear algebra heavy lifting from R to Python, giving a significant speed-up. The runtimes are displayed in Figure 3.4. Clearly the runtime increases in a roughly linear manner with the number of customers. Extrapolating from this graph, it would take in the region of 5 hours to run the program for the userbase who have been active in the previous 90 days, and 8 hours for all active accounts. This is not an especially onerous amount of time as the program could feasibly be executed by night to provide a set of recommendations for the subsequent day.

3.5 Evaluation

Evaluating whether temporal filtering improve the accuracy of the core recommender algorithm was vital to justify it as a post processing step. If no accuracy gains were observed under temporal filtering then the business goal of accurate time-aware recommendations in the Sportsbook would remain elusive. Hu *et al.* (2008) outline an evaluation methodology as follows. A set of

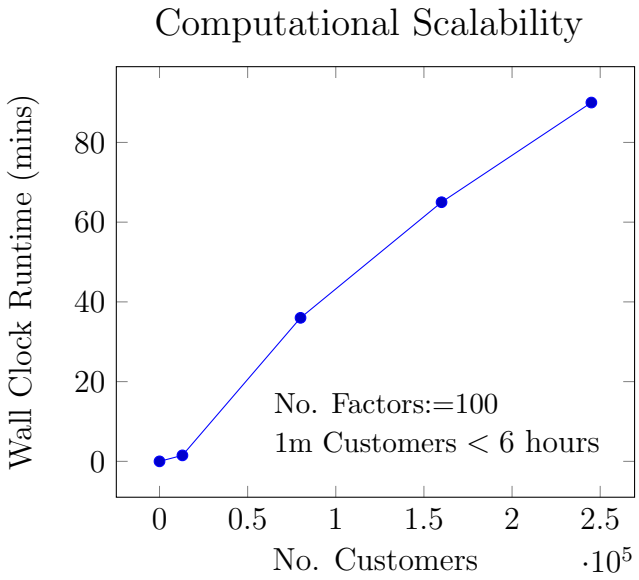


Figure 3.5: Runtime is linear with the number of customers. The model will execute in under 6 hours for one million customers.

recommendations is obtained for a given customer, then each recommendation is assigned a percentile ranking based on its position in the list: the top-rated sport is assigned a 'rank' of 0% and the bottom recommendation receives a rank of 100%. Figure 3.5 displays a toy sample of 6 recommendations where ranks have been assigned to each. The value $rank_u = 16.77\%$ was computed using 3.5.1. A $rank_u = 0\%$ would indicate that the customer has only bet on their strongest recommendation, $rank_u = 100\%$ would indicate that the customer has only bet on their weakest recommendation and we would be better recommending random sports.

The raw SQL data was partitioned into a training and test set, such that the model was trained on the former but evaluated on the latter. The rank metric was then applied to quantify how well the computed recommendations predicted actual betting in the test set. Effectively this method of evaluation captures whether or not recommendations are tailored to actual customer preferences. Our training set consisted of all bets made by approximately $82,200$

Ranked Output

Position	Pref Score	Event Name	Rank%	Total Stake €
1	0.9995	Royal Ascot	0	50
2	0.9986	FA Cup	20	90
3	0.9983	USPGA	40	10
4	0.9976	Horse Racing	60	0
5	0.9974	Galway	80	5
6	0.9970	Basketball	100	0

Figure 3.6: Evaluation by Percentile Ranking: values of $rank_u < 50\%$ indicate that the recommendations for customer u are better than random guesses. We propose a novel extension of this metric to evaluate recommendation accuracy along the temporal dimension.

customers from the opening of their account until the July 31st 2013. The test was composed of all bets made by those customers active in both the test and training set and spans dates from August 1st 2013 to July 31st 2014. In order to apply Equation 3.5.1 a list of recommendations were extracted for these customers from the trained model. Also required were their total stakes in each recommended sport; Total Stake$_{ui}$ were extracted from betting histories in the test set. Accuracy is then calculated by aggregating over all customers in the test set via:

$$\overline{rank} = \frac{\sum_{u,i} rank_{ui} \times \text{Total Stake}_{ui}}{\sum_{u,i} \text{Total Stake}_{ui}} \tag{3.5.1}$$

Where, $\overline{rank}$ is the average $rank$ score of all customers in the test set.

3.5.1 Evaluation along the Temporal Dimension

The evaluation methodology presented above is naive because it does not address the transient nature of the sporting calendar. Required is an evaluation strategy to prove temporal filters facilitate more accurate recommendations than the base model. To this end we selected April 1st to June 17th 2014 (hence Spring 2014) as our testing period because several large sporting events

took place during this time, including the Grand National, US Masters, French Open, several big Premier League matches etc. Separate test sets for each day were extracted yielding 78 overall. In total 42858 customers were active in both the test and training (as defined above) periods. Now, for each test set (i.e. each day), the $\overline{rank}$ was computed via 3.5.1 using temporal and non-temporal recommendations for this set of customers. The evaluation metric is thus projected along the temporal dimension, such that 3.5.1 generalises to:

$$\overline{rank(t)} = \frac{\sum_{u,i} rank_{ui} \times \text{Total Stake}_{ui}}{\sum_{u,i} \text{Total Stake}_{ui}} \tag{3.5.2}$$

Where, for activated filters $i \in \{\text{Temporal Recommendations}\}$ or when filters are inactivated $i \in \{\text{Non-Temporal Recommendations}\}$.

There are two motivations for the proposed (experimental) evaluation strategy. Firstly, computing $\overline{rank(t)}$ on a per day basis is necessary for a fair trial as the non-temporal evaluation benefits from extended test periods. To illustrate consider a customer for whom the US Masters is their top recommendation. It is somewhat useful to predict that this customer will place a bet on the US Masters at some point in Spring 2014. However, an evaluation performed on say a two year test period will reward the system for making a fairly obvious prediction. As the test period spans wider time horizons accuracy will just increase monotonically. Secondly, it is not even clear how to aggregate temporal filters over an extended period in order to compare like with like. The natural granularity for evaluation is therefore 24 hours, the duration over which the temporal filters are invariant.

3.5.2 Evaluation Results

Figure 3.7 plots the temporal and non-temporal rank scores over the test period. Clearly, the temporal recommendations outperform their non-temporal analogues with ranks of 25.18±2.18% ($\overline{rank}_{\text{temp}}$) and 33.72±0.73% ($\overline{rank}_{\text{static}}$) respectively averaged over the test period. Figure 3.8 justifies the use of a t-test to find the statistical significance of this result. Let H_0 be the null hypothesis such that:

H_0 <=> Temporal filters do not improve accuracy during the testing period.

Then the p-value (p) of achieving $\overline{rank}_{\text{temp}} \leq 25.18\%$ assuming H_0 is:

$$p \equiv \Pr(\overline{rank}_{\text{temp}} \leq 25.18\% | H_0 \text{ TRUE})$$

Standardising using the sample standard deviation of temporal ranks ($s = 11.54$) yields the following test statistic,

$$T := \frac{\overline{rank}_{\text{temp}} - 33.72}{\frac{11.54}{\sqrt{78}}} \backsim t(77)$$

$$=> T := \frac{25.18 - 33.72}{\frac{11.54}{\sqrt{78}}} \approx -6.52$$

$$\therefore p = 1 - F_T(-6.52) \approx 0$$

The p-value is approximately zero indicating strong evidence against H_0 at a confidence level of $\alpha = 5\%$. Therefore, the evidence supports the alternative hypothesis that temporal filters improve accuracy.

The 95% confidence intervals for $\overline{rank}_{\text{temp}}$ and $\overline{rank}_{\text{static}}$ were also calculated using the Students t-distribution. Since 95% of the probability mass of the t-distribution with $\nu = 77$ lies between $T = \pm 1.665$ (source: Wolfram Alpha) the confidence intervals are:

CI for $\overline{rank}_{\text{temp}}$:

$$25.18 \pm 1.665 \frac{11.54}{\sqrt{78}} \, [\%] => [23.00, 27.36] \, [\%]$$

CI for $\overline{rank}_{\text{static}}$:

$$33.72 \pm 1.665 \frac{3.86}{\sqrt{78}} \, [\%] => [32.99, 34.45] \, [\%]$$

Where, $n = 78$ samples, $s_{\text{temp}} = 11.54$ and $s_{\text{static}} = 3.86$.

3.5.3 Limitations of the Evaluation

Historical test sets can only reveal lower bounds on the 'true' accuracy of any recommender system. The evaluation described above is no different. In practice recommendations will be presented to customers affording the model opportunity to influence their behaviour. Since our system did not exist during the test period in April 2014 the evaluation treats the it harshly. Therefore, while the average daily rank of 25.18% indicates better than random time-aware recommendations, the live system will exhibit better accuracy still.

Note that the dithering post-processing step was inactivated during evaluation. Randomisation is necessary for the Sportsbook recommender because there are a limited number of sports on offer. Customers would quickly disengage with their recommendations if they find them boring or invariant over time. However, dithering suppresses a customer's most preferred sports thereby suppressing accuracy in the short term. So while dithering is vital for the system to drive value for Paddy Power into the future, its benefit cannot be observed on a fleeting historical test set. It is turned off so that our central research focus, temporal dynamics in the Sportsbook, may be addressed directly.

Figure 3.7 shows that on several isolated days filters degrade the accuracy of the base recommendations. The evaluation strategy discussed in section 3.5.1 makes it possible to pick out such days and examine why the breakdowns occur. Filters can then be fine tuned until a satisfactory performance is achieved.

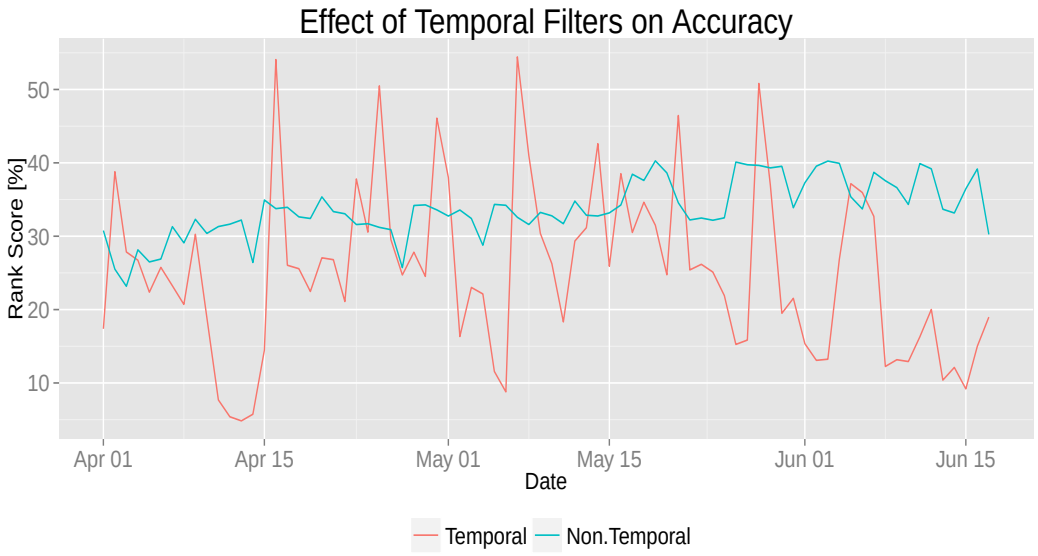


Figure 3.7: Time-aware recommendations (red curve) outperform the base recommendations (blue curve). Accuracy is improved on average by applying temporal filtering but with some capricious exceptions.

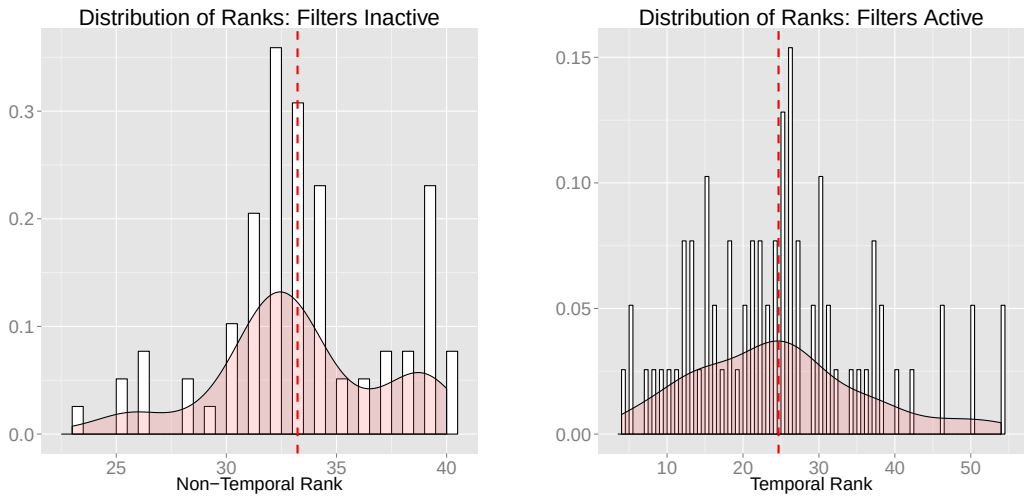


Figure 3.8: To reject H_0 we assume that the daily rank scores satisfy a Student's t-distribution: justified because the rank scores admit a symmetrical distribution around their mean (dotted line) with fat tails.

Chapter 4

Results and Discussion

4.1 Introduction

In this chapter we will present a set of results for a range of different customers. We will also demonstrate the effect of the various post-processing steps employed. We will demonstrate how a user who has made only a small number of bets will generally obtain unreliable results. We will also show how a high dithering score will alter the results for a given user. And finally we will demonstrate how the temporal filters work by examining output for a number of different dates.

It must be noted that the customer data used to create and test this recommender system is protected by a confidentiality agreement. Therefore while all results provided are real outputs from the system, only general and non-specific comments about a customer's betting history can be provided in the following section.

4.2 Results

4.2.1 Unreliable Recommendations

The first set of results, displayed in Table 4.1, are for a customer who has only ever made a single bet, on the Lotto. These results have been abbreviated to 5 per table, as it's not especially instructive to show all 10 results since the recommendations become weak. Notice that this customer's top result is a very strong recommendation for the Lotto, and the second result is a

Ranked Output

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	0.991362	Lotto	Lotto	Aussie Rules
2	0.281115	Lotto Specials	Lotto	Aussie Rules
3	0.223130	US Racing	Lotto	Aussie Rules
4	0.152404	Fairyhouse	Lotto	Aussie Rules
5	0.150670	Music/Weather	Lotto	Aussie Rules

Table 4.1: Recommendations, Minimal Betting History

Ranked Output

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	1.932015	Lotto	Lotto	Aussie Rules
2	0.223129	US Racing	Lotto	Aussie Rules
3	0.164510	Baseball	Lotto	Aussie Rules
4	0.162707	GAA	Lotto	Aussie Rules
5	0.150670	Music/Weather	Lotto	Aussie Rules

Table 4.2: Recommendations, Minimal Betting History, Saturday

recommendation for Lotto Specials but this is far weaker. The reason being that the algorithm can tie the two items together, i.e. people who bet on Lotto are often found to bet on Lotto Specials, but because this customer inputs so little information to the algorithm, it cannot return predictions with any degree of confidence for events he has not bet on. This is a good illustration of the cold-start problem.

No temporal filtering has been applied at this point, and the results provided to this customer are so weak that the principled randomisation method

Ranked Output

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	0.300073	Lotto	Lotto	Aussie Rules
2	0.229055	GAA	Lotto	Aussie Rules
3	0.223129	US Racing	Lotto	Aussie Rules
4	0.162197	Baseball	Lotto	Aussie Rules
5	0.150965	Golf	Lotto	Aussie Rules

Table 4.3: Recommendations, Minimal Betting History, Sunday

is automatically inactivated, as the results from 2 down are all but random guesses anyway. Table 4.2 shows this same set of results after temporal filters have been applied. The customer has a strong recommendation for Lotto, and the day chosen was a Saturday, which is the biggest Lotto day in the week (based on turnover). As a result of these circumstances the temporal filter has significantly boosted the predictive score for Lotto almost by a factor of 2, this is exactly the desired behaviour. Table 4.3 shows how the output changes when we take a temporal recommendation for a Sunday, which is the further day of the week away from the next Lotto draw. The predictive score for Lotto drops to roughly 0.3, this is again expected behaviour as a strong recommendation for Lotto is not desired so far away from the next draw.

A data driven approach to making more reliable recommendations to this customer is to use the cross-sell information to see how Lotto predicts other events in the Sportsbook. Some of the correlations may not be entirely intuitive, nonetheless they suggest the events have deep connections in the minds of the Paddy Power customer base. Figure 5.4 (appendix) shows that the algorithm has uncovered relationships between the Lottery and various Horse Racing and Soccer events. More interestingly, there is a link between the GAA betting and the Lottery as is the case for Music & Weather betting. These sets of similar events could be offered as recommendations for this customer rather than just relying on the basic model which does not have enough information to generate reliable recommendations. This is our novel workaround for the cold start problem that CF recommender systems suffer from.

4.2.2 Dithered Recommendations

The second set of results presented in Tables 4.4, 4.5, 4.6 highlight how dithering can change the set of results. For the purpose of demonstrating dithering, temporal filters were deactivated. A customer with a particularly high dithering score has been selected, and this customer has also made a far greater number of bets than their counterpart in the preceding example, hence their recommendations are much stronger. The customer in question had made 1956 bets covering 46 different events. This large number of events staked, coupled

with the spread of their bets over these events, has resulted in a high dithering score. The base results for this customer are displayed in Table 4.5. A total of 6 results have been dithered—rows 4, 5, 6, 8, 9, 10—selected at random by the program. It can clearly be seen that the dithered and non-dithered results differ. As discussed previously, this is designed to grab the customers attention so the recommendation system doesn't become boring to them by returning the same events every time.

A further point on dithering worthy of mention is that if a customer bets almost exclusively on a relatively esoteric sport, for example Cycling, they will receive a moderately high dithering score. In a sense this doesn't seem especially valid as the customer is showing that they have a clear preference for a single sport and are not showing themselves to be adventurous in their choice of sports to bet on. On the other hand, Paddy Power have an interest in identifying customers who show a willingness to bet outside the usual sports of Soccer and Horse Racing, as over their lifetime these customers can often prove to be quite valuable. The dithering score is an effective means of identifying such customers. It is not a perfect measure that fits with absolutely every customer, but it is a measure that can easily be applied to a vast number of customers in an automated manner and it does identify those who exhibit more unusual betting patterns.

The purpose of Table 4.6 is to show that dithering does in fact return randomised results. Two sets of dithered recommendations were obtained for this customer, one immediately after the other, as illustrated in Tables 4.5 and 4.6. It can be seen that the two sets of results are very different. Each time the function to return recommendations is called, a (not necessarily) different set of events is replaced from the top 10, and similarly a different set of replacements are selected. It is clear in both instances that the Pref Score column is no longer sorted in descending order, which is the behaviour we would expect.

Dithered Output

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	0.9990918	Hockey	Basketball	Baseball
2	0.9990776	Snooker	Tennis	Darts
3	0.9989962	Rugby	Cheltenham	Cricket
4	0.9989833	Lotto	FA Cup	Lotto Specials
5	0.9989730	Motor Racing	Wimbledon (M)	Golf
6	0.9989639	Handball	Volleyball	Beach Sports
7	0.9989475	Basketball	Hockey	Volleyball
8	0.9989385	Athletics	Water Sports	Football Matches
9	0.9989302	Tennis	Snooker	Cricket
10	0.9989222	Cricket	Rugby	Premier League (BIR)

Table 4.4: Recommendations, Non-Dithered

4.2.3 Effects of Temporal Filter

This subsection will examine the effects of the temporal filter over the course of a single event—the US Open golf tournament, a four day long event which ran from June 14th 2012 to June 17th 2012. The filter acts as a modifier that alters the recommendation score of customers as a function of the date. For all dates in 2012 prior to June 10th, the value of the US Open filter is 0. As described in the Chapter 3, the filters for extended events like Golf Majors are created by looking at how the turnover on the event in previous three years varied over the course of the event. In previous years, the first day of the US Open has been the biggest day in terms of turnover generated by Paddy Power. The second day appears to be less interesting to customers. Therefore, we would expect to see the strongest recommendations for the US Open on June 14th 2012 but weaker recommendations on the June 15th and none whatsoever a week before the event. Figure 4.1 and Table 4.7 show how this desired behaviour can be achieved using filtering.

The temporal filters are designed to reflect the exponential growth in turnover observed as the start date of a major event draws near. The rationale being that it is not necessary to have a strong presence for an event in lists of recommendations several days before the event starts. The majority of

Dithered Output				
Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	0.9990918	Hockey	Basketball	Baseball
2	0.9990776	Snooker	Tennis	Darts
3	0.989962	Rugby	Cheltenham	Cricket
4	0.9988859	Volleyball	Handball	Baseball
5	0.9987613	Darts	Snooker	Cricket
6	0.9987446	Boxing	Martial Arts	Darts
7	0.9989475	Basketball	Hockey	Volleyball
8	0.9987362	Baseball	Volleyball	Hockey
9	0.9987356	Australian Open (L)	Australian Open (M)	US Open (L)
10	0.9987231	Australian Open (M)	Australian Open (L)	French Open (M)

Table 4.5: Recommendations, Dithered Results 1

customers prefer to make a bet on an event on the day the event commences, hence the filter peaks on the opening day. Table 4.7 lists the recommendations generated for a customer with a predicted interest in Golf, over the course of the 9 days depicted in Figure 4.1. It demonstrates clearly how a recommendation is modified by the temporal filter as a function of time. The variable affected by the filter here is the Preference Score.

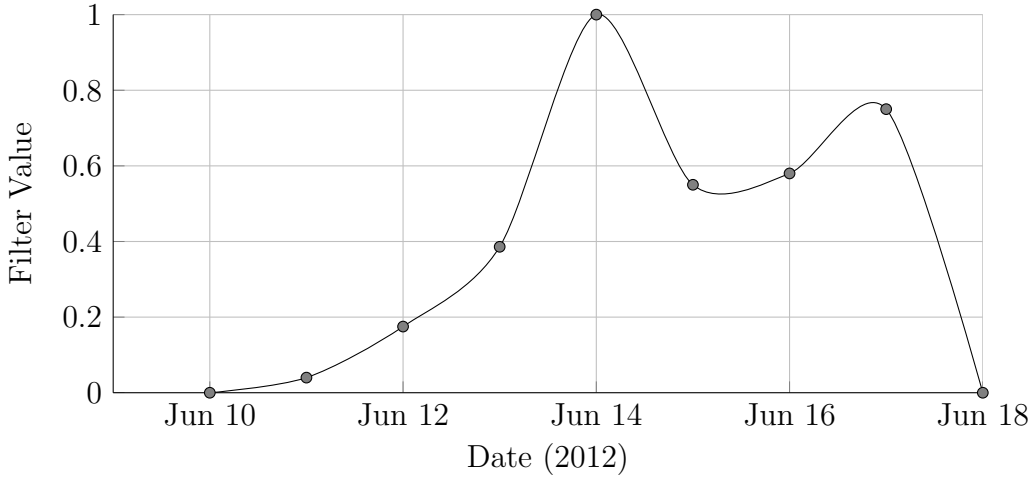


Figure 4.1: Temporal Filter, US Open Golf

Ranked Output

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	0.9990918	Hockey	Basketball	Baseball
2	0.9990776	Snooker	Tennis	Darts
3	0.9988418	French Open (M)	French Open (L)	Australian Open (M)
4	0.9987913	Golf	USPGA Golf	Open Championship
5	0.9987613	Darts	Snooker	Cricket
6	0.9989639	Handball	Volleyball	Cricket
7	0.9987231	Australian Open (M)	Australian Open (L)	French Open (M)
8	0.9987010	Athletics	Handball	Hockey
9	0.9989302	Tennis	Snooker	Cricket
10	0.9986966	French Open (L)	French Open (M)	Australian Open (L)

Table 4.6: Recommendations, Dithered Results 2

Personalised Output 1

Date	Pref Score	Event Name	Explanation 1	Explanation 2
Jun 10	0.00000000	US Open	Golf	US Masters
Jun 11	0.04236412	US Open	Golf	US Masters
Jun 12	0.11589978	US Open	Golf	US Masters
Jun 13	0.86135470	US Open	Golf	US Masters
Jun 14	0.31654123	US Open	Golf	US Masters
Jun 15	0.42150874	US Open	Golf	US Masters
Jun 16	0.43985612	US Open	Golf	US Masters
Jun 17	0.63341129	US Open	Golf	US Masters
Jun 18	0.00000000	US Open	Golf	US Masters

Table 4.7: Time Sensitive Output

4.2.4 Personalised Recommendations

The following set of figures will demonstrate that applying the filtering step does not override the algorithm's capacity to generate personalised recommendations. An effort was made to find two customers with dissimilar betting histories. Nonetheless, it is not uncommon for the most popular events to appear within the top 10 base recommendations for even very different customers. We will take the example of Cheltenham, which is one of the largest single events of the year in terms of stake. When we take base recommendations (i.e. without

dithering or temporal filtering), it is common for Cheltenham to appear in the top 10 list as firstly, it is extremely popular in its own right, and secondly, the algorithm will correlate Cheltenham with other popular Horse Racing events. That is, if the algorithm uncovers a link between Cheltenham and general Horse Racing, and a given customer has revealed a preference for Horse Racing through their betting history, there is a high likelihood that Cheltenham will be returned as a recommendation. The temporal filters were active when Tables 4.8 and 4.9 were generated, the date under scrutiny was March 15th 2013 or the day of the Cheltenham Gold Cup, the single biggest race within the festival. At this time the temporal filter that affects recommendations for Cheltenham would obviously be active.

The first customer examined (Table 4.8) has only made a modest number of bets, of which around 10% were on Horse Racing events. He has bet on both Aintree and Cheltenham but the vast majority of his bets were on Soccer. It should be noted that customers who bet primarily on Soccer also tend to bet heavily on Racing, so we could expect a signal for Cheltenham in his recommendations even if he never staked it previously. The second customer examined (Table 4.9) has bet primarily in Horse Racing events. Therefore one might expect a strong signal for Cheltenham in his recommendations. Notice that the second customer does in fact receive a much stronger recommendation for Cheltenham. This result demonstrates that his dominant interest in Horse Racing is born out by the model and provides confirmation that the output is still personalised after post-processing.

Personalised Output 1

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	1.79003171	Premier League	Cheltenham	World Cup Matches
2	1.38733866	Cheltenham	Aintree	Horse Racing
3	0.99748858	Outrights	World Cup Matches	Premier League
4	0.99471163	Snooker	FA Cup	US Open
5	0.99427423	Music/Weather	FA Cup (BIR)	Aintree

Table 4.8: Personalised Recommendations 1

Personalised Output 2

Position	Pref Score	Event Name	Explanation 1	Explanation 2
1	1.9939492	Cheltenham	Horse Racing	Rugby
2	1.4363270	Rugby	Cheltenham	Champions League
3	0.9973822	Tennis	Wimbledon (M)	French Open (L)
4	0.7721227	Snooker	Tennis	Punchestown
5	0.7722582	US Racing	Fairyhouse	Wimbledon (L)

Table 4.9: Personalised Recommendations 2

4.3 Cross-sell

Early consultations with Paddy Power made it clear that non-obvious insights were required from the model, an insight like "Horse Racing predicts Soccer" could add no value to the business. However, finding unexpected relationships between sports would be of significant value. The idea of using the core Factor Algorithm for cross-sell was introduced in Chapter 3, Subsection 3.3.3. The key point expressed there was that the algorithm generates rationales for each recommendation.

4.3.1 Implementation

The idea is that we can count the number of times a rationale appears with each strong recommendation. So for example, if Basketball occurs in the top 10 recommendations for 20,000 customers we count the number of times every other sport appears as its first, second or third rationale. The result might be that Baseball appears 5000 times as a strong predictor for Basketball, Swimming never and so on for all other sports. In R this aggregation is implemented by initialising an $n \times n$ matrix with zeros, where n is the number of sports in the model. Each row and column corresponds to a unique sport. So when the row corresponding to Basketball is reached in the above example, the cell where row Basketball intersects column Baseball is overwritten with 5000. The end result is therefore a matrix expressing the relationships between all the sports. We toggle to zero all but the strongest six rationales when visualising this

data.

The cross-sell matrix was visualised in a cord diagram. To this end we adapted a package originally developed by Martin Krzywinski in Circos.[1] The package uses JavaScript and HTML to generate an interactive cord diagram from a csv input file (Firefox compatible). Note that it was necessary to 'flatten' the cross-sell matrix using PivotTable in excel. Some events are omitted for clarity like Ladies Wimbledon or very low turnover events. Screen shots are displayed in the appendix. Each outer cord represents a sport with ribbons showing their mutual associations. The size of a cord is an indication of how many times the associated sport appears as a rationale, so for example The Champions League is a strong predictor overall. The wide of a ribbon indicates the importance of an association. For example, in Figure 5.3 Premier League Betting in Running (BIR) is a rationale for Outright Premier League recommendations 8000 times but the latter never predicts the former. Note the asymmetry.

4.3.2 Patterns

Many interesting patterns emerge from the diagram. Firstly, we have taken care to model BIR separately from Outright Betting. Customers who participate in BIR (betting during the course of an event) are typically more valuable that those who make Outright bets (bet before event and collect after) and so addressing this dichotomy in the customer base was vital. Now, Horse Racing splits about 1% BIR vs 99% Outright betting due to the short duration of the races. Soccer on the other hand is the largest market for BIR. Therefore, we would expect to see strong associations between the BIR sports modelled and Soccer events with weak or no associations to Horse Racing. These expectations are borne out by the diagram in Figure 5.3 where one can see links between Premier League BIR and the set {FA Cup, World Cup BIR, Champions League BIR FA Cup BIR, Premier League, Soccer Specials} but only one weak link to Aintree (a racing event). The true value of Figure 5.3 is proba-

[1] Adapted from: http://bl.ocks.org/mbostock/4062006

bly rests in the result that {American Football, Boxing, Cricket, Basketball, Lotto} are also predictors of Premier League BIR. The presence of American Racing and Basketball is perhaps more surprising and could encourage their inclusion in marketing directed at customers with a leaning to BIR.

Figure 5.3 captures an interesting signal in the data. Here the cross-sell correlations are displayed for Basketball with links to the set {Tennis, American Football, Volleyball, Baseball, BIR Premier League, GAA, Hockey, Snooker}. When presented to the Customer Intelligence Team in Paddy Power they commented on the dominating presence of American Sports in the rationales for Basketball. Clearly, Baseball, Volleyball, American Football and Tennis would appeal to customers with an interest in American Sports. When correlations are displayed for Baseball the set {Tennis, American Football, Volleyball, Baseball, Hockey, Football Matches} manifests. Again the signal from the American type sports is obvious. The value of this kind of result is that baskets of related sports can be constructed in a principled manner. Recall that customers and sports were mapped to a factor space dimensionality 100 to generate this data. The model can therefore uncover very deep patterns in how customers interact with the Sportsbook, some of which may elude human vaticination.

Finally, recall Section 4.2.1 where the potential to use this data as a solution to the Cold Start Problem in Collaborative Filtering was described.

Chapter 5

Conclusions and Future Research

As the evaluation and results above demonstrate, recommender systems can be successfully applied to the field of online betting. We have clearly demonstrated that our recommender algorithm is able to select events that a customer may enjoy with a reasonable degree of accuracy. The algorithm scales linearly with the number of customers present so it should be actionable across the entire Paddy Power online userbase. We present a possible workaround to the cold-start problem by using the cross-sell information to determine what a customer may enjoy when they have demonstrated only limited engagement with the Sportsbook. To the best of our knowledge this approach represents a novel contribution to the field.

Furthermore, we demonstrated how important it is to address the transient nature of the sporting calendar. There are two criteria for recommending an event to a customer. Firstly there must be an indication that the customer shows a preference for that given event, and secondly the event must actually be happening within a relatively short period of time. A t-test showed that the temporal filters do improve the accuracy of recommendations. Temporally filtered recommendations were shown to have a relative score of 25.18% compared to 33.72% for non-temporally filtered scores, in an evaluations system where 0% is the most accurate and 100% is the least accurate measure. The paper by Hu *et al.* (2008) achieve a score of roughly 9% for the same number of latent factors as used in this project, so we accept that while there is a reasonable level of accuracy presented it still could be improved significantly. We conclude that it is possible to deal with the temporal issues in the Sportsbook.

This project has delivery the analytic core of a recommender system that will hopefully drive value into the future, there is scope for further work to

improve the system. The first thing that must be mentioned is that not all sports and events were modelled in a temporal fashion. This was for either of two reasons; sports like Greyhound Racing take place virtually all year round so in some sense it is not worth temporally modelling them (they are always available), or secondly for some events there simply was not enough data available to create a temporal filter. The data used to model these filters was historical data concerning how popular an event was in the past, and further data projecting how big an event would be for Paddy Power in the future. Without these two datasets it is not feasible to create a filter for the lower turnover events.

Another issue is that even sports that have been temporally modelled, they have only been modelled to a certain extent. If we take Soccer as an example, a filter was created for the Premier League, Champions League, FA Cup, World Cup, and then all other soccer events. This final category encompassed a huge number of different events within the category of soccer, and splitting up the overall category into more individual events, each with their own temporal filter, would certainly improve the accuracy of the system. This ties in with the level of data granularity this project dealt with. By going to a lower level of data granularity, for example from an overall sport to an event within that sport, further accuracy gains could be made and a more specific recommendation could be returned to the customer. This would of course come at some computational expense however. Another possible option in this instance would be to use the Latent Factor model to obtain a recommendation for an overall sport, then use a Nearest Neighbour model, which can be quite computationally efficient, to obtain a result for an individual event within that sport.

In Section 3.5 it was demonstrated that even though the temporal filters certainly increase the accuracy of the system, there are still some highly inaccurate days present. It was demonstrated that this commonly occurred on a Sunday, so a deeper examination of betting trends on Sundays would certainly help in this regard. One possible theory is that on most Sundays during the football season there are televised Premier League soccer matches running, it's

possible these weren't given enough weight in the temporal filter. Our evaluation methodology facilitates straightforward identification of days when the filters fail to improve on the base recommender.

Finally, our implementation of dithering is subject to future adaptation. The metric used is likely too blunt to adequately identify all high value customers. Additionally, several parameters must be hard coded to make it work, a more data driven approach to randomisation would be desirable. Due to the modular nature of the algorithm we have implemented, it should be trivial for Paddy Power to replace our metric with a more sophisticated instrument.

Appendix

5.1 Detailed Program Description

This section will outline the different files that make up the overall program and describe their function. The data.table package (R) is used extensively and familiarity with its syntax and methods are important for anyone wishing to edit the following files. We have tried to provide comments in the files that explain what everything is doing.

- readTrainingData.R

  This is the file that reads in the customer data and formats it such that the Latent Factor model can operate on it. It reads in customer data, the format of which is outlined in section 3.3.1. Data tables are created to hold the data for all the major events in golf, tennis, soccer and racing. Furthermore, some lesser popular sports are aggregated into a single event. A matrix called *test* is the final output of this file, which contains a row for each event and a column for each customer. The entries to this matrix are the cumulative stake a customer has made on a given item. A record of all the customers in the data set is kept in a vector called *tot.users*. All customers who have made no bets at all are removed from the matrix as they don't really contribute anything.

- Dates.R

  This file is designed to find the most recent bet a customer has made on a given event/sport, which is stored in a data table called *final*. Once again the overall dataset is split into individual data tables for the various constituent events in golf, soccer, racing and horse racing. Each is processed separately to find the most recent bet a customer has made per event, then all the data tables are bound together to create a single

table at the end of the file. This file can then lookup the most recent bet on any event for a given customer.

- dithering.R

  This file is used to calculate the dithering score for each customer, which are saved in a vector called *dithering.score*, which has an entry for every customer in the data set. The first step taken in this file is to calculate the "unusualness" score of each event. Then for each customer the spread of their bets over each event is found, and finally these two factors are used to calculate the dithering score. Code is also included for plotting the dithering score for a selection of customers, or also for finding the score for a single customer.

- filter.R

  This is the file in which the temporal filters are calculated and stored in a matrix called *activation*, which has a column for each event, and rows that correspond to the strength of the filter for a given event on a given day. This file uses FTS (first time staker) data, which is a prediction of how big a given event in the sporting calendar will be, as a lookup for checking the stake for that event in previous years. For example we see an event some time in the future in 2014 that the FTS data signals will be a significant event for Paddy Power in terms of stake. We then go check the total stake for this event in previous years. Using these values for stake, normalised between 0 and 1, we can predict how big the given event will be in the future. This is the essence of the temporal filters. This file is quite long and much of the data has to be processed in a fairly non-automated manner. The advantage is it's quite easy to add more events to be filtered once the data becomes available.

- runFactorModel.R

  This is the main R file that calls all the others and also calls the Python file that does the linear algebra calculation. There are several constants in this file that are defined, $alpha = 40$ and $epsilon = 0.00000001$. Both

of these values are used in calculating the confidence with which we believe a customer has a preference for a sport. There is also *lambda* = 300, which is a constant that is included in a calculation to prevent overfitting. The number of *factors* in the latent factor model is also defined here and set to 100, this can be increased to improve accuracy of the recommendations, but at a significant computational cost. The confidence matrix, number of users, number of items, factors, and regularisation constant are all passed to the Python file FactorModel.py which carries out the Alternating Least Squares calculation. The file type used here is .mat (MATLAB file type) which was chosen because they can be read by both R and Python. Another .mat file is output then which is called *recs*, this file is similar in size to *test* but instead of stakes contains the recommendations for each customer for each event.

There are also several functions defined in this file. Firstly, dither() calculates exactly how many entries of a customer's list of recommendations are to be dithered, selects the events to be changed and the events that replace them at random, and carries out this change. Second, temporal() applies the temporal filters to a list of recommendations. Lastly, get.recs() is the overall function that incorporates the previous two and returns recommendations for a given customer. The following syntax is used: get.recs(recs, customer number, days, TRUE/FALSE, date) where *recs* is as described previously, the customer number is the index of the customer in *tot.users*, days is the number of prior days for which you want to exclude events that a customer has recently bet on (e.g. set to 30 and the recommendations will return no events the customer has bet on in the prior 30 days), the TRUE/FALSE sets whether the results are dithered or not with the default set to TRUE, and the date is the date for which you want the recommendations returned.

- FactorModel.py

  This file is based on the linear algebra equations presented in Hu *et al.* (2008), for calculating a set of recommendations and also calculates explanations for those recommendations. The methods have been ex-

plained in detail elsewhere in the paper, this file is the least likely to have to be modified of all. It simply takes in a matrix of size users * items and returns a similar sized matrix of recommendations. The only constant that must be defined is *numSports* which sets the number of items for which explanations are generated, for the purposes of this project it was set to 30.

5.2 Event Division and Aggregation

As discussed previously, some sports contain within them very large events that attract a huge amount of business for Paddy Power. An example would be the Grand National, this single horse race could easily have a higher stake for Paddy Power than some entire sports over the course of the year. For this reason these large events were extracted from their sport superclass and modelled as an individual event. For the following sports, each of the events listed was treated as its own sport. A temporal filter for all these events was also created.

- Horse Racing

  Aintree Grand National, Cheltenham, Royal Ascot, Punchestown, Galway, Irish Grand National, all other horse racing

- Soccer

  Premier League, Champions League, FA Cup, World Cup, all other soccer

- Tennis

  Wimbledon, US Open, Australian Open, French Open, all other tennis

- Golf

  US Masters, Open Championship, USPGA Championship, US Open, Ryder Cup, all other golf

Conversely, for the following categories, more than one sport may have been aggregated to form a single event that account for several sports. Such an example is the GAA category would contain GAA Football, Hurling and Camogie. These aggregated events were:

Soccer Specials, Horse Racing Specials, Lottery Specials, GAA Specials, Rugby Specials, Greyhound Specials, Outrights, Martial Arts, Greyhound Racing, Basketball, Water Sports, Ahletics, Winter Sports, GAA, Beach Sports, Hockey, Rugby, Music/Weather, Horse Racing, Aussie Rules, Football.

Figure 5.1: Chord Diagram showing Full Cross-sell model.

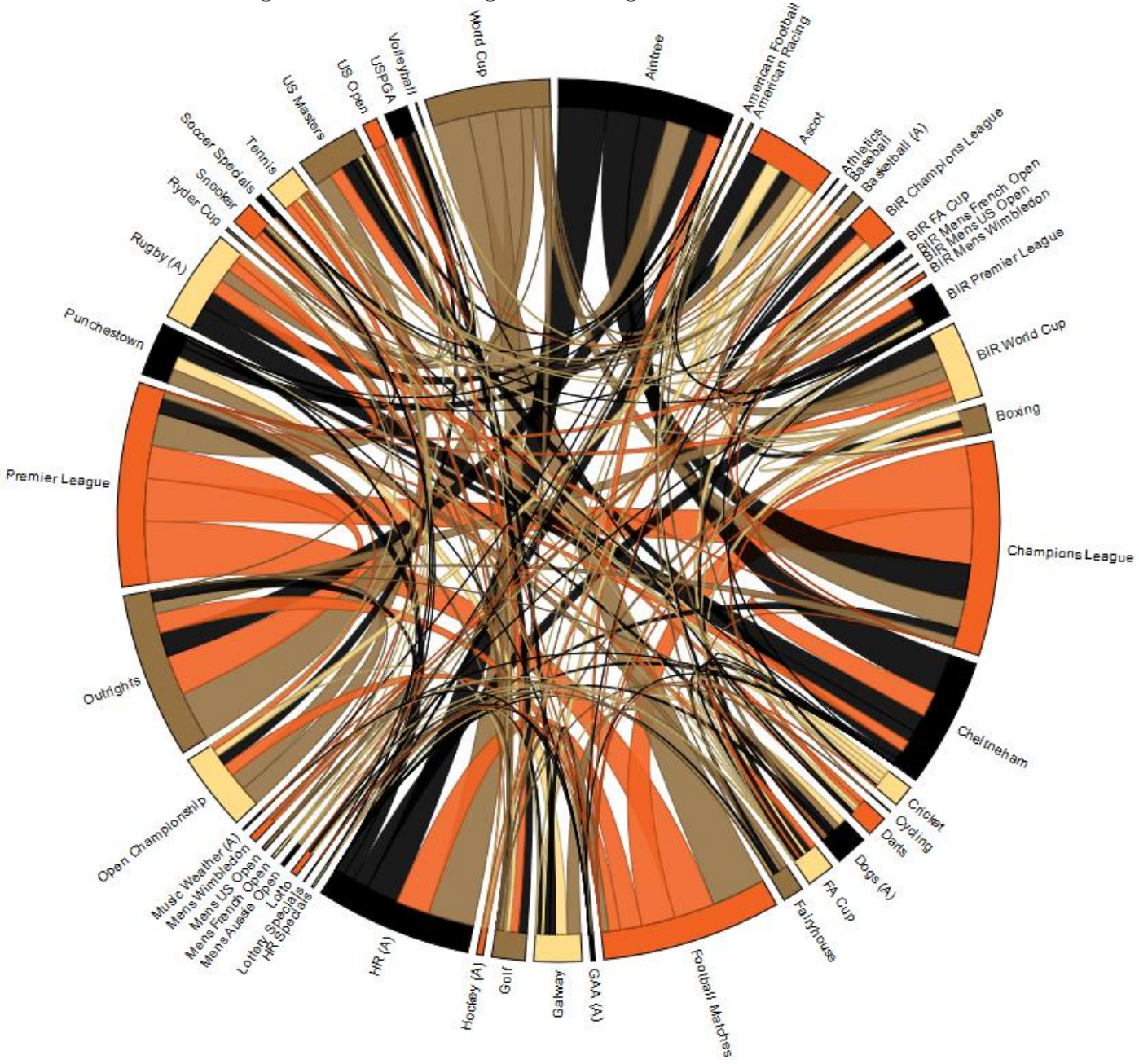


Figure 5.2: Chord Diagram showing Premier League (Betting in Running) Cross-sell information.

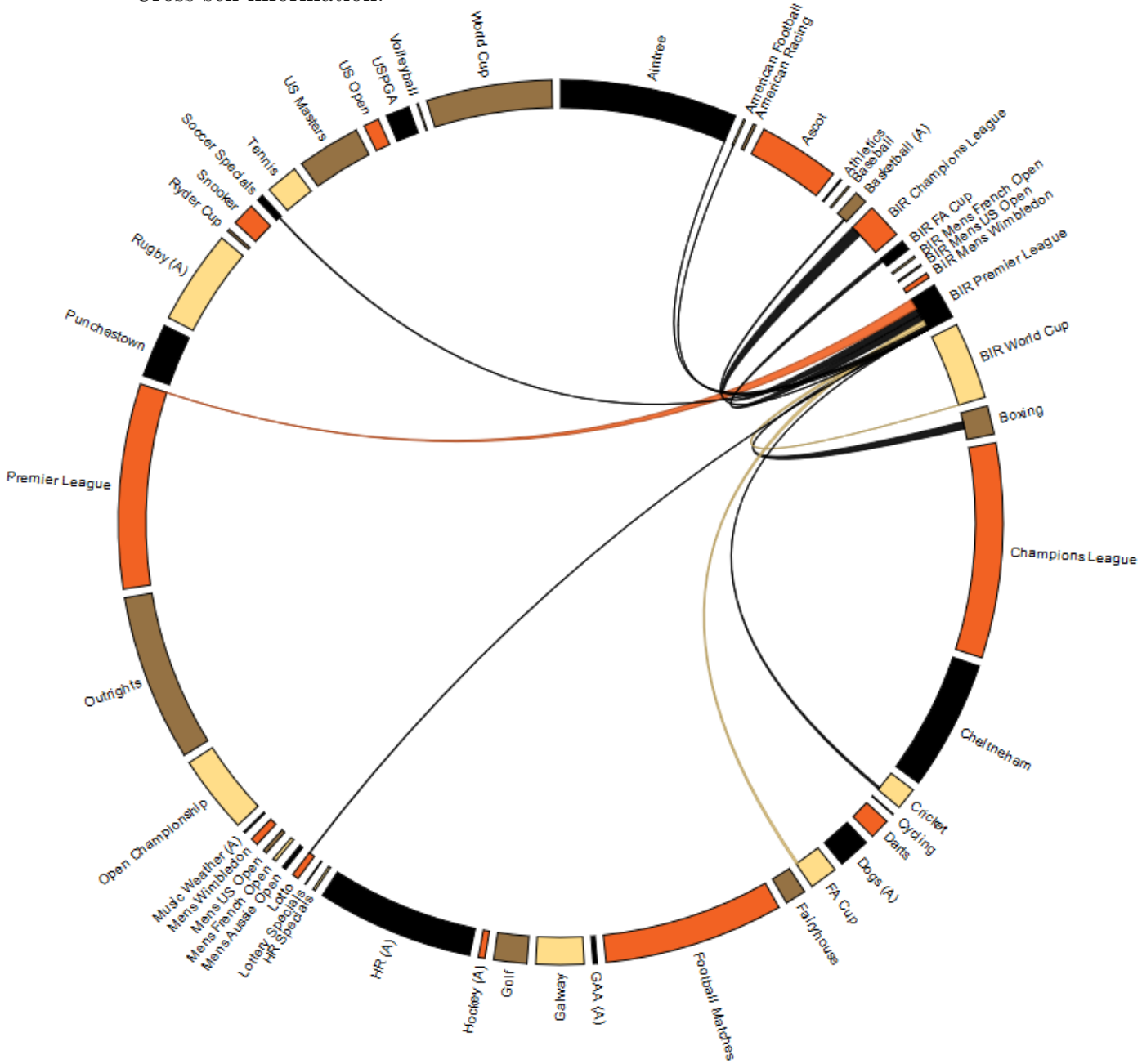


Figure 5.3: Chord Diagram showing Basketball Cross-sell information.

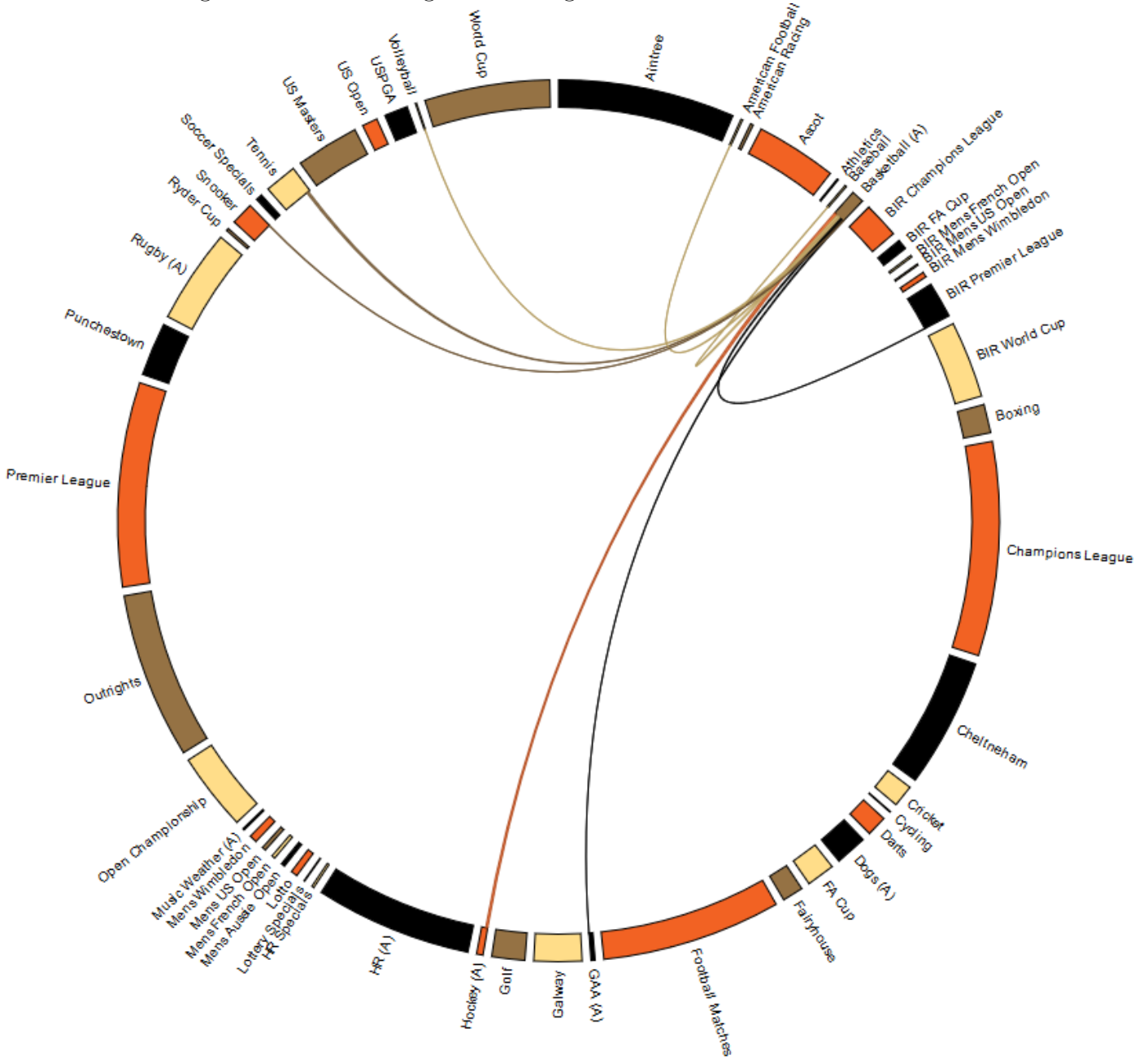


Figure 5.4: Chord Diagram showing Lottery Cross-sell information.

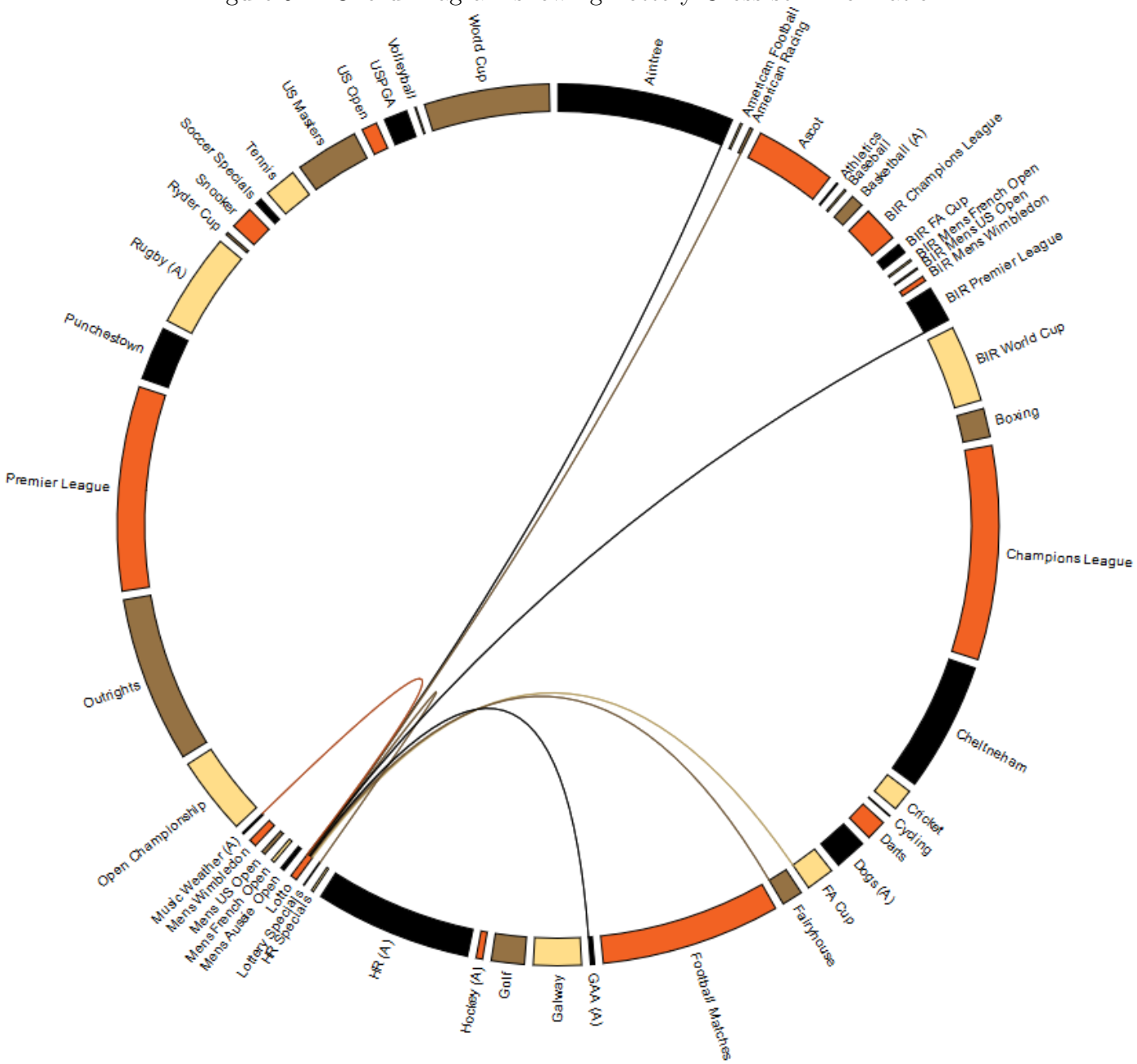


Figure 5.5: One Filter does not fit all Sports.

Annual MM	Weekly	Sporadic	Non Temporal
US Masters	Aussie Rules (A)	Boxing	Outrights
US Open	Baseball	Football Matches	Music/Weather (A)
USPGA Golf	American Football	Soccer Specials	BIR (A)
Open Champ.	Basketball (A)	Horse Racing (A)	Dogs (A)
Ladies Wim.	Darts	HR Specials	Dogs Specials
Mens Wim.	Gaa (A)	Premier League	Todays Specials
Ladies US Open	GAA Specials	Champions League	Tennis
Mens US Open	Rugby (A)	FA Cup	American Racing
Ladies Aus Open	Rugby Specials	World Cup Matches	Martial Arts (A)
Mens Aus Open	Golf	The Ryder Cup	Hockey (A)
Ladies French Open	Lotto	BIR Prem League	Beach Sports (A)
Mens French Open	Lottery Specials	BIR Champ League	Motor Racing
BIR Ladies Wim.		BIR FA Cup Matches	Handball
BIR Mens Wim.		BIR World Cup	Badminton
BIR Ladies US Open			Volleyball
BIR Mens US Open			Bowls
BIR Ladies Aus Open			Table Tennis
BIR Mens Aus Open			Athletics
BIR Ladies French			Archery
BIR Mens French			Gymnastics
Cheltenham			Fencing
Aintree			Swimming
Ascot			Pool
Fairyhouse			Equestrian
Galway			Shooting
Punchestown			Water Sports (A)
			Tri-Athletics (A)
			Cycling
			Rowing
			Cross Country
			Cricket
			Squash
			Winter Sports (A)
			Coursing
			Show Jumping
			Weightlifting
			Snooker

Bibliography

Baltrunas, X., Linas abd Amatriain, 2009. Towards time-dependent recommendation based on implicit feedback.

Dunning, T. and E. Friedman. 2014. *Practical Machine Learning*. O'Reilly Media, Inc.

Goldberg, D., D. Nichols, B. M. Oki and D. Terry. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, **35**(12): 61–70.

Hu, Y., Y. Koren and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference*, pages 263–272.

Koren, Y. 2008a. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434.

——, 2008b. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

——. 2010. Collaborative filtering with temporal dynamics. *Communications of the ACM*, **53**(4): 89–97.

Koren, Y., R. Bell and C. Volinsky. 2009a. Matrix factorization techniques for recommender systems. *Computer*, **42**(8): 30–37.

——. 2009b. Matrix factorization techniques for recommender systems. *Computer, Institute of Electrical and Electronics Engineers*, **42**(8): 30–37.

Park, D. H., H. K. Kim, I. Y. Choi and J. K. Kim. 2012. A literature review and classification of recommender systems research. *Expert Systems with Applications*, **39**(11): 10059–10072.

Pazzani, M. J. and D. Billsus. 2007. Content-based recommendation systems. In: *The adaptive web*, pages 325–341. Springer.

Sarwar, B., G. Karypis, J. Konstan and J. Riedl, 2000. Analysis of recommendation algorithms for e-commerce. In: *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM.

——, 2001. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

Stefanidis, K., I. Ntoutsi, K. Nørvåg and H.-P. Kriegel, 2012. A framework for time-aware recommendations. In: *Database and Expert Systems Applications*, pages 329–344. Springer.

Tang, T. Y., P. Winoto and K. C. Chan, 2003. On the temporal analysis for improved hybrid recommendations. In: *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 214–220. IEEE.

Vinagre, J. 2011. Time-aware collaborative fitering: a review.