Politecnico di Torino

EURECOM
Sophia Antipolis

amadeus

Politecnico di Torino
Institut EURECOM

Master of Science Double Degree in *Computer Engineering* and *Data Science*

Deep Models of Decision

Supervisors
Prof. Bartolomeo Montrucchio[1]
Prof. Marios Kountouris[2]
Dr. Alix Lhéritier[3]
Nicolas Bondoux[3]
Dr. Rodrigo Acuna Agost[3]
Dr. Eoin Thomas[3]

[1]Politecnico di Torino
[2]EURECOM
[3]Amadeus IT Group

Candidate
Federico Tiblias

Academic Year 2022 - 2023

To anyone who believed in me.

Abstract

Decision Theory is a branch of mathematics and economics that studies the process of decision-making under uncertainty. It provides a framework for analyzing the choices made by individuals or organizations and gives guidance on how to make the best decisions possible given the limited information and resources available. Choice Modeling is a subfield of Decision Theory that aims to explain and measure how people or groups make decisions when presented with various options. It achieves this by modeling the link between the characteristics of those options and the likelihood that a person will select one over another. Some of its applications include analyzing consumer behavior in marketing research, predicting voter preferences in political campaigns, optimizing public policy decisions, and designing recommendation systems for online platforms. The main goal of this thesis is to propose and assess novel approaches to Choice Modeling that leverage the expressivity of deep machine learning models. The first approach we investigate is an application of deep learning in conjunction with Quantum Decision Theory, a framework developed by cognitive psychologists that proposes a way of modeling phenomena such as uncertainty and interactions between alternatives inspired by quantum mechanics. The second one is based on the Attention Mechanism, notoriously used in deep learning for modeling long-distance relationships between inputs. We aim to explore the capabilities of these new models and provide a general way of applying them to new choice problems. We evaluate the efficacy of these new methods on three different Choice Modeling datasets of increasing complexity. Furthermore, we compare our methods against reference models from both Classical and Quantum Decision Theory. Finally, we discuss potential avenues for future research.

Résumé

La théorie de la décision est une branche des mathématiques et de l'économie qui étudie le processus de prise de décision dans l'incertitude. Elle fournit un cadre pour l'analyse des choix effectués par des individus ou des organisations et donne des conseils sur la manière de prendre les meilleures décisions possibles compte tenu des informations et des ressources limitées disponibles. La modélisation des choix est un sous-domaine de la théorie de la décision qui vise à expliquer et à mesurer la façon dont les personnes ou les groupes prennent des décisions lorsqu'on leur présente diverses options. Elle y parvient en modélisant le lien entre les caractéristiques de ces options et la probabilité qu'une personne en choisisse une plutôt qu'une autre. Certaines de ses applications comprennent l'analyse du comportement des consommateurs dans la recherche marketing, la prédiction des préférences des électeurs dans les campagnes politiques, l'optimisation des décisions de politique publique et la conception de systèmes de recommandation pour les plateformes en ligne. L'objectif principal de cette thèse est de proposer et d'évaluer de nouvelles approches de la modélisation des choix qui tirent parti de l'expressivité des modèles d'apprentissage automatique profond. La première approche que nous étudions est une application de l'apprentissage profond en conjonction avec la théorie de la décision quantique, un cadre développé par des psychologues cognitifs qui propose une façon de modéliser des phénomènes tels que l'incertitude et les interactions entre les alternatives inspirée de la mécanique quantique. La seconde est basée sur le mécanisme d'attention, notoirement utilisé en apprentissage profond pour modéliser les relations à longue distance entre les entrées. Nous visons à explorer les capacités de ces nouveaux modèles et à fournir une manière générale de les appliquer à de nouveaux problèmes de choix. Nous évaluons l'efficacité de ces nouvelles méthodes sur trois ensembles différents de données de modélisation de choix de complexité croissante. En outre, nous comparons nos méthodes à des modèles de référence issus de la théorie de la décision classique et quantique. Enfin, nous discutons des pistes de recherche potentielles pour l'avenir.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Company overview and business sector

One of the top software firms in the world, Amadeus IT Group is the industry-leading Global Distribution System (GDS) provider for the travel and tourism sectors. It has more than 16,000 employees globally and was founded in 1987 as an alliance between Air France, Lufthansa, Iberia, and Scandinavian Airline System. While the corporate headquarters are located in Madrid, Spain, the primary location for product development is Sophia Antipolis, France. Amadeus supports client operations at the market level via 173 local Amadeus Commercial Organizations (ACOs) in over 190 countries, as shown in Figure 1.1.
More in detail, Amadeus provides two main services to the airline industry:

- through its central reservation system, as a GDS platform, provides the foundation for searching, pricing, booking, ticketing, and a number of other processing services to various travel suppliers and travel agencies (CRS)
- as an Information Technology (IT) company, provides solutions to automate many logistics processes, such as revenue management, data analysis, inventory management, and travel intelligence services to a wide range of customers, such as airlines, tour operators, hotels, etc.

The company currently provides services to 770 airlines, 132 airport operators, 53 cruises, 43 car rental companies, 1M+ hotel properties, 90 rail operators, and 128 ground handlers as well as many other businesses.

1.1.1 Team presentation

This project was developed as part of the Applied Research and Technology (ART) team, whose mission is to conduct exploratory and applied research in the fields

Figure 1.1: Amadeus world presence as the leading company in the travel industry

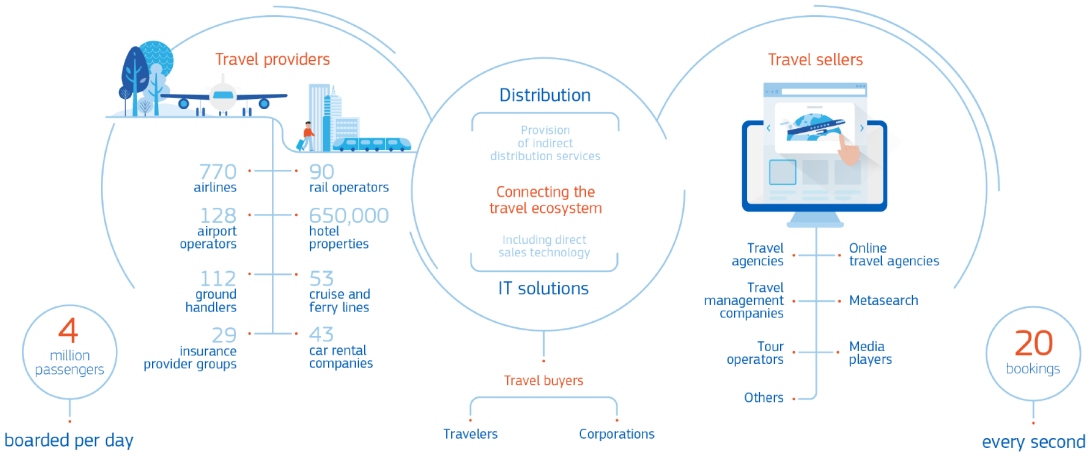


Figure 1.2: Amadeus business model

of Artificial Intelligence, Data Science, and Emerging Technologies. The team's primary objective is to facilitate the development of new applications and enhance existing products throughout Amadeus. The research work carried out by the team is usually results in publications in renowned journals and conferences, and much of it is eventually incorporated into the company's services pipeline or submitted as patents. The applied research branch is typically driven by specific use-cases, with the aim of addressing company or customer needs, while the exploratory branch strives to keep the company up-to-date on technological advances. This

thesis work belongs to the exploratory branch, focusing on long-term objectives.

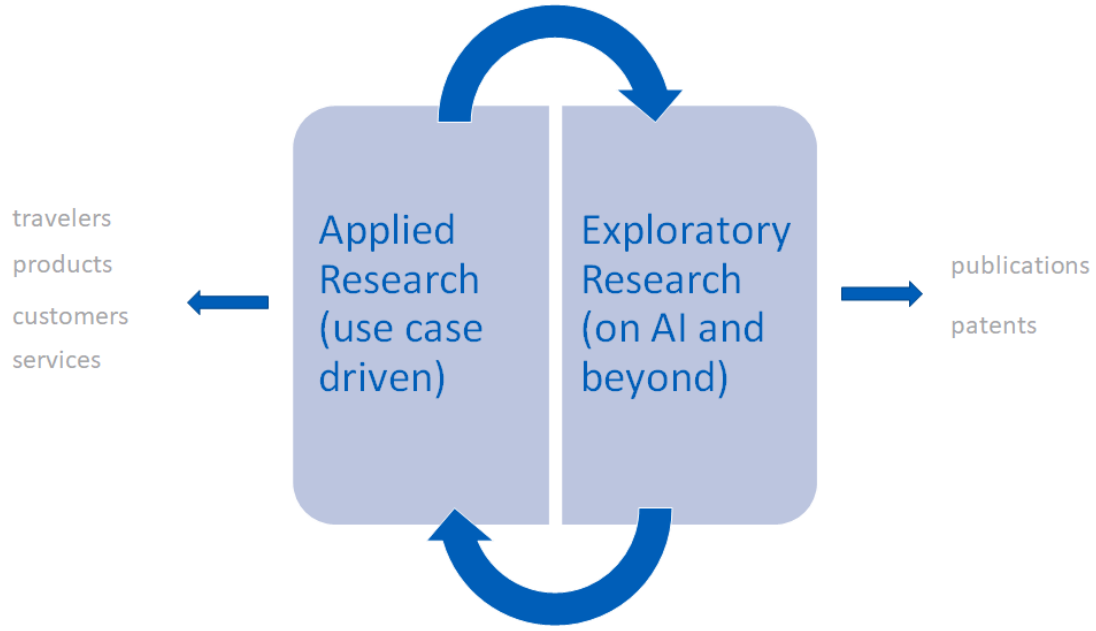


Figure 1.3: Amadeus Applied Research and Technology team activity

In addition, the team's research efforts are strengthened through collaborations with academic institutions such as the Massachusetts Institute of Technology (MIT), EURECOM, the University of Côte d'Azur, and Inria. These partnerships involve various forms of collaboration including internships, industrial PhDs, and joint research projects.

1.2 Thesis objective

The main objective of this thesis is to investigate new paradigms of choice modeling. In particular, we build upon two different paradigms that have received significant attention in recent years: Quantum Decision Theory (QDT), a novel framework that is shown capable of resolving paradoxes and model various psychological anomalies observed in human decision makers, and the Attention Mechanism, a well-established technique from deep learning not commonly applied in choice modeling despite its potential. We evaluate our proposed methods on problems related to the travel industry. The thesis has therefore both a theoretical and applied flavor, first proposing new methods and then assessing their performance.

Chapter 2

Background

This chapter aims at providing the reader with a comprehensive understanding of the theoretical concepts that underpin this thesis. We start by introducing the field of decision theory and its relevance to this work. This will be followed by a brief overview of the methods that will serve as reference points for our comparisons. Next, we delve into quantum probability and quantum decision theory, explaining their key principles and how they relate to decision-making. Finally, we examine some deep learning models that have inspired our work, highlighting their key features and contributions to the field.

2.1 Decision Theory

Decision Theory is a branch of mathematics and economics that studies the process of decision-making under uncertainty. It provides a framework for analyzing the choices made by individuals or organizations, taking into account their preferences, goals, and the information they have available. Decision Theory focuses on how to make the best decisions possible given the limited information and resources available, and it has a wide range of applications in fields such as finance, engineering, and health care.

2.1.1 A theory of choice

Choice modeling is a branch of Decision Theory that aims to measure and comprehend how individuals or groups make decisions when presented with a range of options or alternatives. It is commonly applied to analyze the decision-making process related to products, services, or policies, by constructing a model that shows the connection between the attributes of those options and the likelihood of a person selecting one option over another. The main contributions and most

widespread models in this discipline derive from Utility Theory, an economic framework that assumes that individuals make rational choices in order to maximize some hidden parameter called *utility* [Neumann and Morgenstern, 1944]. Utility is a function of alternative and individual and is unknown. The objective thus becomes to estimate this function and quantify utility for a given choice.

An important class of choice models is the Multinomial Logit (MNL), a classification method that generalizes Logistic Regression to multiple alternatives. MNL defines the probability of choosing an item i from a set S as proportional to the latent value of the item, represented as $P_S(i) = \frac{w_i}{\sum_{j \in S} w_j}$, where w_i is the latent value of item i. MNL models satisfy Luce's axiom, also known as *independence of irrelevant alternatives*. This principle states that the probability of selecting one alternative over another is not affected by the presence of other alternatives in the set. In other words, an individual's choice between two alternatives is only dependent on the relative preference for each alternative, and not influenced by other alternatives that may be available. Any model satisfying Luce's axiom can be expressed as an MNL [Luce, 1977]. More formally, Luce's axiom introduces the concept of *dominance*: if alternatives a and b are compared and a is always chosen, we say that a *dominates* b and express it as $a \succeq b$. Luce's axiom also implies stochastic transitivity for dominance (if $P(a \succeq b) \geq 1/2$ and $P(b \succeq c) \geq 1/2$, then $P(a \succeq c) \geq 1/2$ for all $a, b, c \in S$), which requires a total order of all elements and prevents the expression of cyclic preference situations [Luce, 1977].

However, actual human choices have been shown to be far from rational and often violate the assumptions of Utility Theory. For example, Allais' paradox [Allais, 1953] presents some evidence that individuals prefer a certain outcome over a risky one even if it defies the expected utility principle (e.g. winning \$1M for sure or \$5M with a 20% chance). Ellsberg's paradox [Ellsberg, 1961] shows that individuals tend to avoid uncertainty and prefer alternatives with known distributions, again defying the expected utility principle. In classical literature, two types of responses have been provided to explain those paradoxes. First, non-expected utility theories, the best-known one being the Cumulative Prospect Theory [Tversky and Kahneman, 1992] that takes into account biases in the perception of probabilities and outcomes. Second, non-deterministic utility approaches, the most commonly used framework being the Random Utility Model RUM [Block and Marschak, 1960, Manski, 1977] where the utility has a random component accounting for unobserved features and population heterogeneity.

2.1.2 Context Effects

Although these alternative theories are able to explain the aforementioned paradoxes, they still fall short in explaining the so-called *context effects* that occur when the preference for a given alternative no longer depends on its characteristics only but also on the other alternatives presented in the choice set [Huber et al., 1982, Tversky, 1972, Simonson, 1989], the order in how they are presented and even previous questions or choices that can condition preferences [Shanteau, 1970, Hogarth and Einhorn, 1992]. Luce's axiom itself is a simplifying assumption that often does not hold for empirical choice data. Satisfying Luce's axiom and stochastic transitivity are strong limitations, and a way to build models that account for real phenomena is to abandon these assumptions entirely.

To account for these context effects, one approach is to use more comprehensive and flexible models that can incorporate information from all the alternatives in a choice set to calculate the context-sensitive utility of each option. In this approach, utility is no longer a function of an individual and a single alternative but rather is a function of the whole problem. For example, in [Lhéritier et al., 2019], the utility is derived from the features of each alternative and some information about the context is injected by using relative values (with respect to the other alternatives of the choice set). In [Peterson et al., 2021], where two alternatives are considered, each one with a distribution of the outcomes that is fully known by the decision-maker, a large class of functions represented by neural network taking as input all the known information has been proposed, and shown to be better at explaining the choices than utility based theories. Although this generality can be appealing, it can be difficult to train such models when the number of alternatives is variable and large.

2.1.3 PCMC-Net

PCMC-Net is an extension of a wider and more flexible class of methods called Pairwise Choice Markov Chains first proposed in [Ragain and Ugander, 2016]. This class of models includes the popular Multinomial Logit (MNL) model, but also other models that do not adhere to Luce's axiom, stochastic transitivity, or regularity. Pairwise Choice Markov Chains define the choice distribution as the stationary distribution of a continuous-time Markov chain, which is determined by a transition rate matrix. Despite not adhering to Luce's axiom in its full form, these models still satisfy a weakened version called uniform expansion. This principle states that if additional "copies" of an option are added (without preference between them), the probability of choosing one element from the copies remains constant regardless of the number of copies.

A Pairwise Choice Markov Chain defines the choice probability $P_S(i)$ as the

probability mass on the alternative $i \in S$ of the stationary distribution of a continuous-time Markov chain (CTMC) whose set of states corresponds to $S \subseteq U$. U is identified as the universe of possible alternatives, of which S is a subset (i.e. there are many possible alternatives, but an individual can choose between only a few of them at a time). The rate matrix of the subset of states in the Markov model is computed by restricting the rate matrix Q of the whole universe U to only rows and columns corresponding to alternatives in S. This resulting matrix, Q_S is thus constructed from pairwise transition rates between every two alternatives in the current choice. These rates are obtained by estimation from data: the off-diagonal elements $q_{ij} \geq 0$ directly, and the diagonal elements as $q_{ii} = -\sum_{j \in S/i} q_{ij}$ for each $i \in S$. Thus, the total number of parameters for the model is $|S|(|S|-1)$. A constraint is imposed in order for the stationary distribution to exist:

$$q_{ij} + q_{ji} > 0 \tag{2.1}$$

The stationary distribution π_S is obtained by solving:

$$\begin{cases} \pi_S Q_S = \mathbf{0} \\ \pi_S \mathbf{1}^T = 1 \end{cases}$$

which [Norris, 1997] shows to be equivalent to:

$$\pi_S Q'_S = \left[\begin{array}{c|c} \mathbf{0} & 1 \end{array} \right]$$

.

with $Q'_S = \left[\begin{array}{c|c} ((Q_S)_{ij})_{1 \leq i \leq |S|, 1 \leq j < |S|} & \mathbf{1}^T \end{array} \right]$.

Parameters of Q_S are estimated from a dataset $\mathcal{D}$ by means of log likelihood maximization of:

$$\log \mathcal{L}(Q; \mathcal{D}) = \sum_{S \subseteq U} \sum_{i \in S} C_{iS}(\mathcal{D}) \log P_S^Q(i)$$

. where $P_S^Q(i)$ is the probability of selecting i from S as a function of Q, and $C_{iS}(\mathcal{D})$ represents the number of times in the data that i was chosen out of S.

The authors encounter several problems during training, such as numerical instabilities leading to violations of constraints [Ragain and Ugander, 2016], as well as severe overfitting in cases where the number of examples of each alternative are scarce [Lhéritier, 2019]. The second limitation in particular is what motivates PCMC-Net.

PCMC-Net is an amortized deep learning approach for which the architecture is invariant to the number of alternatives in a single choice ($|S|$) and even of the

whole universe of alternatives ($|U|$). Another difference from PCMC is that the former only learns relationships between alternatives, but there is no observation of the attributes of each choice in order to estimate parameters q_{ij}. PCMC-Net, on the other hand, identifies each alternative by its features and those of the decision maker. Therefore, q_{ij} is a function of the features of the i-th and j-th alternatives and those of the decision maker. It represents this function by means of several fully connected neural layers whose parameters are estimated via gradient descent. While the original PCMC paper used *scipy*'s *SLSQP* optimizer [Virtanen et al., 2020], PCMC-Net is based on *PyTorch* [Paszke et al., 2019] and make use of its automatic differentiation engine to perform gradient descent, specifically using an *Adam* optimizer.

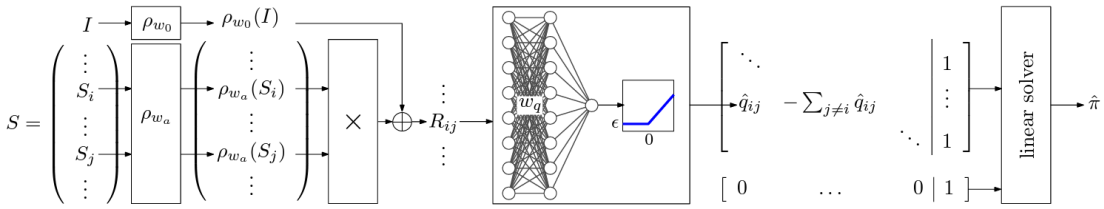


Figure 2.1: Full architecture diagram of PCMC-Net

The architecture is composed of the following layers:

- **Input layer**: Initial layer fed with an alternative set S for a specific choice belonging to a given feature space $\mathcal{F}_a$, as well as *context* features C belonging to a given feature space $\mathcal{F}_c$. The term *context* is an umbrella term that refers to both the attributes of a particular individual making the decision and the environment around it. More generally, it contains all information that is common to all alternatives.

- **Representation layer**: comprised of representation function to map alternatives' features

$$\rho_{w_a} : \mathcal{F}_a \mapsto \mathbb{R}^{d_a}$$

as well as context features

$$\rho_{w_c} : \mathcal{F}_c \mapsto \mathbb{R}^{d_c}$$

where $d_a, d_c \in \mathbb{N}$ are the representation size of alternative and context, and are defined as a hyperparameter of the architecture, and w_a and w_c are weights of the representation layer to be estimated during training.

- **Cartesian product layer**: Builds all possible pairs of alternatives and concatenates them with the context to produce an $|S| \times |S|$ matrix R. This layer allows PCMC-Net to be invariant to the number of alternatives $|S|$.

$$\{\rho_{w_a}(S_1), \ldots, \rho_{w_a}(S_{|S|})\} \times \{\rho_{w_a}(S_1), \ldots, \rho_{w_a}(S_{|S|})\}$$

$$R_{ij} = \rho_{w_c}(C) \oplus \rho_{w_a}(S_i) \oplus \rho_{w_a}(S_j)$$

  where $\oplus$ denotes vector concatenation.

- **Transition rate layer**: The core component modeling the transition rate starting from the features

$$\hat{q_{ij}} = \max(0, f_{w_q}(R_{ij})) + \epsilon$$

  where f_{w_q} consists of multiple fully connected layers parameterized by a set of weights w_q and $\epsilon > 0$ is a hyperparameter. Notice that taking the maximum with 0 and adding ϵ guarantees non-negativity and the condition of Eq. 2.1. The rates $\hat{q_{ij}}$, computed across all possible pairs of alternatives, are combined to form the transition rate matrix $\hat{Q}$ as follows:

$$\hat{Q_{ij}} = \begin{cases} \hat{q_{ij}} & \text{if } i \neq j \\ -\sum_{j \neq i} \hat{q}_{ij} & \text{otherwise} \end{cases}$$

- **Stationary distribution layer**: uses *PyTorch*'s linear solvers to find the solution to the system

$$\hat{\pi} \left[\begin{array}{c|c} \left(\hat{Q_{ij}}\right)_{1 \leq i \leq |S|, 1 \leq j < |S|} & \mathbf{1}^T \end{array} \right] = \left[\begin{array}{c|c} \mathbf{0} & 1 \end{array} \right]$$

  where $\hat{\pi}$ is the probability distribution over S given C.

The model is fitted using gradient descent to minimize the average log likelihood loss against the index of the actual choice Y_S.

$$\text{loss}(\mathbf{w}, C, S, Y_S) = \log \hat{\pi}_{Y_S}$$

with $\mathbf{w} = (w_a, w_c, w_q)$. The original paper uses stochastic gradient descent and dropout to avoid overfitting and obtains stable results on data that PCMC, on the contrary, finds problematic.

PCMC-Net exhibits non-regularity, uniform expansion, and is proven able to approximate any PCMC model arbitrarily well given a function f_{w_q} of sufficient complexity as well as appropriate weights w_a, w_c and w_q. We take major inspiration

from [Lhéritier, 2019] in terms of architecture and applications. The reasons for this are many. Firstly, the codebase for PCMC-Net is readily available[1]. Secondly, there are striking similarities between Hamiltonian evolution in Quantum Decision Theory and Markov Chains that allow us to draw a connection between these two approaches. More details are given in the following chapters where we propose H-Net, a possible feature based application of Hamiltonian evolution to decision making (Section 3.1). Many elements are common, but the core difference is a different quantum-inspired method for computing the final probabilities.

2.1.4 Recommender Systems and Choice Modeling

Recommender Systems are a key area of research in artificial intelligence, aimed at providing personalized recommendations to users based on their preferences and behavior. Recommender systems have become ubiquitous in various domains such as e-commerce, entertainment, and news, playing a crucial role in decision-making processes. They can be thought of as an evolution of Choice Modeling, appropriately adapted to the large volumes they need to handle. At their core, they are machine learning algorithms capable of analyzing large amounts of data on users and items, such as purchase histories, ratings, and interactions, to identify patterns and make predictions about which items a user is likely to be interested in. The RecSys conference, which stands for the Conference on Recommender Systems, is the cornerstone conference for this discipline[2].

Recommender systems are based on collaborative filtering algorithms and have evolved over time, with the introduction of new techniques such as matrix factorization [Jannach et al., 2010], deep and reinforcement learning [Zhang et al., 2017], to name a few. Additionally, with the increasing amount of data being generated, recommender systems are also becoming more personal and dynamic, adapting to the changing needs and preferences of users in real-time [Zhao et al., 2018]. Despite these advancements, there are still several open research challenges in the field of recommender systems, such as the cold-start problem (how to handle new users for whom no information is available in the system) [Gaspar et al., 2019], the scalability issue, and the integration of auxiliary information [Zhao et al., 2018], which continue to inspire new research and drive innovation in the field.

While this thesis is mainly concerned with comparing different possibilities in the domain of Decision Theory, we also try constructing a bridge with recommender systems, drawing upon concepts and techniques from the former discipline to enhance the latter. Our solution is robust to the cold-start issue by means of

[1]Source code repository: https://github.com/alherit/PCMC-Net

[2]More information available at https://recsys.acm.org/

embeddings of the input features to represent each user. This allows us to leverage the information available in the entire dataset to generate a robust representation for new users, even when their preferences are unknown. By doing so, we are never limited by the absence of known preferences for a new user.

2.2 Quantum Probability

Quantum Mechanics is a branch of physics that studies the behavior of matter and energy on the smallest scale, including atoms, subatomic particles, and their interactions. Unlike classical mechanics, which is based on classical concepts of motion, energy, and force, quantum mechanics describes the physical world in terms of wave functions and probability. It provides a mathematical framework to explain and predict the behavior of particles and their interactions, which is crucial for our understanding of the structure and behavior of the atomic and subatomic world. The principles of quantum mechanics have far-reaching implications, from explaining the stability of atoms and the behavior of chemical reactions to shaping our understanding of the fundamental nature of matter and energy. At the foundations of quantum mechanics lies quantum probability (QP), a formalism defining events and probabilities in a different way than its classical counterpart, enabling different operations and phenomena to come in play, such as entanglement and interference. QP is shown to be an extension of classical probability, making it possible to model classical phenomena with the added benefit of a wider scope.

2.2.1 Bra-ket Notation

In quantum mechanics, bra–ket notation (or Dirac notation) is used ubiquitously to denote quantum states. Bra–ket notation is an elegant way of defining vectors and allows defining operations such as inner and outer products. The following is a brief handbook of the notation:

- **Row vector (bra)**: $\langle\psi| = [\psi_1, \psi_2, \ldots]$
- **Column vector (ket)**: $|\psi\rangle = [\psi_1, \psi_2, \ldots]^T$
- **Inner product (bra-ket)**: $\langle\psi|\phi\rangle = \psi \cdot \phi$
- **Outer product (ket-bra)**: $|\psi\rangle\langle\phi| = \psi \otimes \phi$

2.2.2 Quantum Systems

The quantum-probabilistic formalism, as developed by von Neumann [von Neumann, 2018], describes events not in terms of sets, but in terms of vectors and projections.

To describe events in QP we first need to postulate a Hilbert space $\mathcal{H}$ of finite dimension n. This space is spanned by an orthonormal set of basis vectors $V = \{|V_i\rangle, i = 1, \cdots, n\}$. An event s is a subspace spanned by a subset $V_s \subseteq V$ of basis vectors. We can define a projector for subspace as $P_a = \sum_{V_i \in a} |V_i\rangle \langle V_i|$.

If b is a subspace spanned by a subset $V_b \subseteq V$, we can define a meet of events $a \wedge b$ (Note: not an intersection) as $V_a \cap V_b$. In the same way, we can define the join as $V_a \cup V_b$. If two events don't share the same orthonormal basis, then $V_a \cap V_b = \emptyset$ and we cannot correctly define their meet or their join.

We can use the intrinsic randomness assumed by quantum postulates to build objects such as random variables. A random variable is such that observing its value yields one of many possible outcomes with a certain probability. Quantum systems are probability distributions and as such, we can use them to model randomness. We define a quantum system $|s\rangle$ as a vector spanning $\mathcal{H}$. Mathematically, $|s\rangle$ is a linear combination of many elementary events (also called *pure states*) $s_1, \ldots, s_m$ that form an orthogonal basis of $\mathcal{H}$.

$$|s\rangle = \alpha_1 |s_1\rangle + \ldots + \alpha_m |s_m\rangle$$

With $\alpha_1, \ldots, \alpha_m$ complex numbers (their meaning is described in the following section). In quantum jargon, $|s\rangle$ is a *superposition* of its elementary events.

2.2.3 Measurement

Quantum systems are probability distributions, and the way we extract an outcome from them is through a process called *measurement*. Measurement is the act of interacting with a quantum system $|s\rangle$, making the vector space partially or entirely *collapse* into a subspace of its original spanned space. $|s\rangle$ collapses into a subset of its basis states and the probability of this event is given by applying the corresponding projector on the state vector, and then by taking the squared norm of the result

$$P(a) = \|P_a |s\rangle\|^2 = \langle s| P_a |s\rangle$$

This type of measurement is called *projective measurement*. This formula tells us that the probability of an event a given a certain $|s\rangle$ is related to the length of the projection of $|s\rangle$ on the subspace V_a. For multiple events, we represent their conditional probability as:

$$P(a, b) = P(b|a)P(a) = \|P_b P_a |s\rangle\|^2 = \langle s| P_a P_b P_a |s\rangle$$

When an event a is observed, the state vector $|s\rangle$ needs to be revised by renormalizing it to conserve its unitary length:

$$|s_a\rangle = \frac{P_a |s\rangle}{||P_a |s\rangle||}$$

In the next sections, we use specifically elementary events which correspond to one-dimensional subspaces represented by the computational basis. In other words, given an n-dimensional Hilbert space $\mathbf{H}$ where all events and states lie, event i maps nicely to an n-dimensional vector $|s_i\rangle$ where all elements are 0 except for a 1 in position i. For example $|s_2\rangle = [0, 1, 0, \ldots]^T$. As a consequence, all projection matrices are expressed as:

$$P_i = |s_i\rangle \langle s_i| = \begin{cases} 1 & \text{in row } i, \text{ column } i \\ 0 & \text{elsewhere} \end{cases}$$

As a consequence, we can easily compute the probability of base events in a superposition $|s\rangle = \alpha_1 |s_1\rangle + \ldots + \alpha_m |s_n\rangle$ as:

$$P(i) = ||P_i |s\rangle||^2 = |\alpha_i|^2$$

We can also compute the probability of complex events made by a combination of elementary ones by adding together projectors. For example, for a three-state quantum system:

$$P_{1+2} = P_1 + P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Note how there is a subtle difference between a projector obtained as a sum of projectors and a projector obtained from the sum of pure states, like $|s'\rangle = |s_1\rangle + |s_2\rangle = [1, 1, 0]^T$. The corresponding projector $P' = |s'\rangle \langle s'|$ would project vectors onto the one-dimensional $[1,1,0]^T$ subspace, while the aforementioned P_{1+2} would project onto a two-dimensional subspace spanned by $[1,0,0]^T$ and $[0,1,0]^T$ and the resulting probability would be equal to $|\alpha_1|^2 + |\alpha_2|^2$.

As a final remark, we note that the measurement process has many implications when dealing with real quantum objects. For example, the fact that measurement inherently disrupts the system making it unable to recover most of the information contained within before the measurement took place. Our approach, not relying on any such physical constraint, allows us to play with numbers in ways that would not be physically possible on real quantum hardware.

2.2.4 Differences with classical probability

Quantum probability is of particular interest for our work as an extension of classical probability, in the sense that it allows phenomena such as entanglement and interference, that are not explainable with classical probability alone. Using vectors and projections also relaxes many assumptions of classical probability, resulting in

a more flexible model [Yukalov and Sornette, 2015][Busemeyer and Bruza, 2012a]. A visual example of a possible anomaly explained by QP is shown in Figure 2.2, where a conjunction of two events can be more likely than either one singularly.

$$P(a,b|s) \geq P(b|s)$$
$$|P_a P_b|s\rangle|^2 \geq |P_b|s\rangle|^2$$

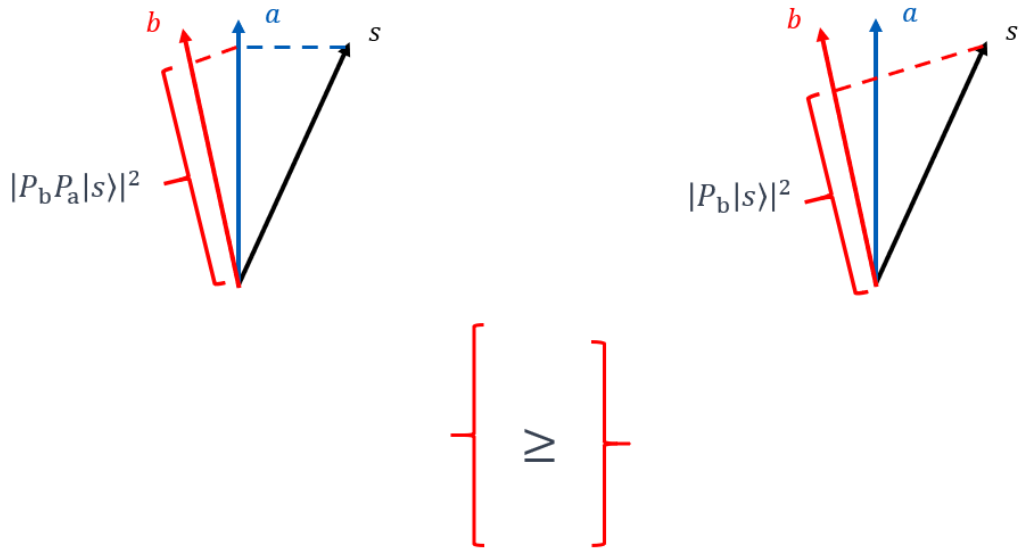


Figure 2.2: Conjunction of two events more likely than the single one

2.2.5 Hamiltonian Evolution

In quantum mechanics, the evolution of a quantum state is described by the Schrödinger equation, which governs the time evolution of a state vector.

$$i\hbar\frac{\partial}{\partial t}\Psi(t) = H\Psi(t) \tag{2.2}$$

The Hamiltonian matrix H is the central object in this equation, as it represents the energy of the system. In particular, we are interested in a time-independent solution, which means that the Hamiltonian does not change with time. A solution to this system is the following Hamiltonian operator:

$$U(t) = e^{-itH} \tag{2.3}$$

Where $e^{(\cdot)}$ represents the matrix exponential function. The solution describes the time evolution of the state vector as a unitary operator U acting on the initial state. This time evolution preserves the norm of the state vector, ensuring that the probabilities of observing the system in any of its states remain constant over time. Also, the unitary property implies that the transition matrix T generated from U is doubly stochastic; that is, both the rows and columns of T sum to one:

$$T_{ij} = |U_{ij}|^2 \qquad \sum_{i'} T_{i'j} = \sum_{j'} T_{ij'} = 1 \quad \forall i, j \tag{2.4}$$

The only condition for this property to hold true is Hermiticity of the Hamiltonian matrix H ($H^\dagger H = HH^\dagger = I$) [Busemeyer and Bruza, 2012b]. Notice also that we incorporate the constant $\hbar$ into the matrix H as the use we make of this formalism is not related to any specific physical quantities.

This operator serves the purpose of describing the flow of probabilities between states of the system and produces sinusoidal behaviors as t increases. An example of evolution for a Hamiltonian operator applied on a four-dimensional state, for different values of time is given in Figure 2.3. Observe how the outcome states begin from a configuration of uniform probability $|s_{t=0}\rangle = [1,1,1,1]^T/2$ and then evolve with a periodic pattern.

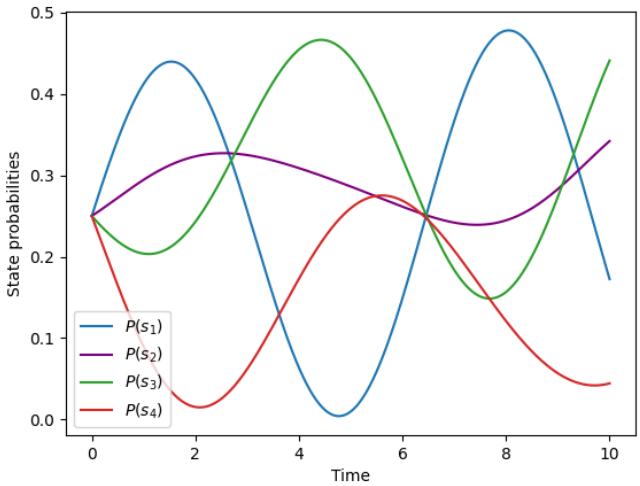


Figure 2.3: Hamiltonian evolution for a 4-dimensional quantum state

Any possible unitary matrix U can be constructed from the matrix exponential of a Hermitian matrix H. This property of the matrix exponential gives a great deal of flexibility to our model as it allows for transforming any unitary vector into any other unitary vector [Busemeyer and Bruza, 2012b]. In the following sections,

we describe how this can be used in conjunction with a neural network to estimate a generic distribution.

2.2.6 Quantum Decision Theory

Quantum Decision Theory (QDT) is a growing field that explores the application of quantum mechanics to the decision-making process. One of the main objectives of this framework is to understand how quantum mechanics can explain aspects of human decision-making and cognition that are not captured by classical models [Yearsley and Busemeyer, 2016a] [Broekaert and Busemeyer, 2017] [Busemeyer et al., 2015]. The fundamental idea behind quantum decision theory is that decisions can be modeled as quantum systems in superposition. A decision existing in many states at once mimics uncertainty, while the act of asking a question and receiving an answer can be seen as a measurement that causes the decision to "collapse" into a definite choice state. This theory has been shown to effectively explain various context effects such as order effects, attraction effects, and conjunction effects.

The focus of Quantum Decision Theory (QDT) is to model certain aspects of human cognition, but is far from making any claims about the existence of quantum phenomena in the mechanisms of the brain, the so-called quantum brain hypothesis [Jedlicka, 2017] [Penrose, 1990]. This hypothesis posits that the human brain operates through quantum processes at a neuronal level and that these phenomena may be the source of consciousness, a long-standing philosophical debate. QDT simply claims that certain cognitive processes, particularly related to decision-making and memory recall, resemble the behavior of quantum objects. Moreover, we'd like to note that the objective of this work is not to weigh in on the ongoing debate about the quantum brain hypothesis, but rather to demonstrate the effectiveness of QDT in addressing decision problems. This controversial topic is better left for cognitive psychologists and physicists to discuss.

QDT has only been applied to simple problems with limited alternatives and a small number of parameters on which to base decisions. The goal of this research is to extend its capabilities to larger, more complex problems that have not yet been addressed.

2.3 Deep Models

Deep machine learning models are a type of artificial intelligence algorithm that has revolutionized the field of computer science. These models are designed to learn and make predictions based on large amounts of data, using a structure composed of many consecutive layers of artificial neurons that allow for increased

representation power. Each layer performs a computation and outputs a representation that can be fed into the next layer. The final layer outputs a prediction or decision [Goodfellow et al., 2016]. The learning process involves adjusting the parameters of each layer so as to minimize the difference between the predicted output and the actual target output, a value called *loss*. This optimization process is performed by calculating the gradient of the loss function with respect to the model parameters, and then using this gradient to update the parameters in an iterative process known as backpropagation [Rumelhart et al., 1986]. Backpropagation is performed using an automatic differentiation library such as PyTorch, TensorFlow, or JAX [Paszke et al., 2019] [Abadi et al., 2016] . The use of automatic differentiation engines and backpropagation has greatly increased the efficiency and effectiveness of deep machine learning models, making them more widely applicable to a wide range of problems.

Deep machine learning models have been applied to a variety of tasks, including image recognition, speech recognition, natural language processing, and game playing, with remarkable results . These models have shown the ability to learn and make predictions with great accuracy, often surpassing human-level performance on certain tasks. With the rapid advances in hardware technology and the availability of large amounts of data, deep machine learning models are becoming an increasingly important tool for solving complex problems in various domains. In this thesis, we take the foundation laid by existing deep machine learning models and apply them to the task of decision theory.

2.3.1 Transformers

Transformer models are a type of deep neural network architecture that have become increasingly popular in recent years for natural language processing (NLP) tasks. At the core of Transformer models is their use of Attention: a mechanism reminiscent of cognitive science that allows them to effectively weigh the importance of each element in the sequence when making predictions. This has proven especially useful in language-related tasks such as machine translation, text classification, and language modeling [Vaswani et al., 2017]. The Transformer architecture has achieved state-of-the-art results on many benchmarks and has become the de-facto standard for solving NLP problems [Lin et al., 2022]. Its popularity is largely due to its flexibility and simplicity, allowing for easy implementation and training even on large datasets.

The model consists of an encoder and (optionally) a decoder block, both of which are comprised of a series of identical layers. Each layer contains an Attention mechanism that computes similarity scores between each element in the sequence and all others, allowing the model to attend to different parts of the input at the same time. Every Attention layer is followed by a feedforward neural network that

processes the attended inputs and generates an output.

The Attention mechanism is fed *tokens* that represent embeddings of some part of the whole input, and outputs tokens that likewise represent parts of the processed output. For instance, in a translation Transformer, input tokens may represent single words in the original language, while the output tokens are translated words. Vision Transformers use as tokens small patches of the image $n \times n$ pixels large and output processed embeddings of the same patches [Dosovitskiy et al., 2020]. Internally, the Attention mechanism works in the following steps:

- First, three different representations for each token t_i are computed: a *query* q_i, a *key* k_i, and a *value* v_i. They are obtained through simple linear operations on the tokens with a common set of weights Q, K, V to reduce complexity:

$$q_i = Q \cdot t_i \qquad k_i = K \cdot t_i \qquad v_i = V \cdot t_i$$

- Then, query and key of each pair of words are multiplied together to obtain an *Attention score* s_{ij}. These scores represent the similarities between each input token:

$$s_{ij} = q_i \cdot k_j \qquad \mathbf{s}_i = [s_{i1}, s_{i2}, \ldots]$$

- Next, a softmax is performed on all scores of a certain token to obtain *weights* $\mathbf{w}_i$ (note that this is a vector). This is done to normalize any possible variation between the scale of scores, which can get very large:

$$\mathbf{w}_i = softmax(\mathbf{s}_i)$$

- Finally, the output o_i of a corresponding input token t_i is computed by multiplying all the weights w_{ij} computed with the input's key with all the token values v_j and summing the result.

$$o_i = \sum_j v_j \cdot w_{ij}$$

The original implementation passes the outputs of each Attention layer through a final fully connected layer. By construction, every input token t_i has a corresponding output o_i. Note however that this correspondence doesn't force the model to process inputs token by token. Instead, Attention makes it so that outputs are constructed by combining representations of other words in the sentence and not necessarily the corresponding input alone. For instance, the English sentence "I

love you" translates in French as "Je t'aime" but the order of words between the two languages changes. Attention is able to account for this by constructing "aime" mostly from "love" instead of "you", which is the corresponding input. If we plot scores for this simple example they would look like figure 2.4. The main concept is that Attention is able to model long-distance relationships easily.

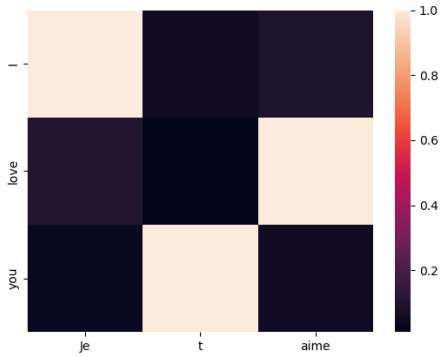


Figure 2.4: Attention scores for "I love you" translated to French

More complex architectures have multiple "heads" in parallel performing different Attention operations at the same time to extract alternative forms of relationships between words, for example one head may capture adjective-noun relationships while another may capture verb-subject ones. Concretely, this amounts to having multiple sets of weights Q, K, V, one for each head.

We find Transformers an inspiration for our search for a new choice model. In particular, cognitive scientists have explored how attention in the human brain can enhance various tasks. Attention's ability to dynamically select information is crucial for memory encoding and retrieval as it ensures that the limited capacity of memory is utilized efficiently [Lindsay, 2020]. Attention also plays an important role in speech recognition, a famous example of which is the "cocktail party problem", where selective attention is used to focus on a particular speaker's speech amidst multiple noises [Bronkhorst, 2015]. Finally, in experiments on visual processing where subjects are cued to attend to a particular visual feature, attention is also shown to enhance performance as it directs resources toward specific aspects of cognition, such as detecting a specific color or shape [Rossi and Paradiso, 1995]. We use these insights to build a Transformer choice model in Section 3.2.

2.3.2 Pointer Networks

Pointer networks are a type of recurrent neural network (RNN) architecture used in natural language processing and reinforcement learning tasks [Vinyals et al., 2015].

The key idea behind pointer networks is to allow the network to "point" to a specific position in the input sequence, effectively choosing which elements of the input to use in generating the output. This mechanism is useful in tasks such as sequence-to-sequence learning, where the output is a different sequence of elements, and the network needs to choose which elements of the input to use in constructing the output. The pointer mechanism is achieved by training the network to predict the probability distribution over the elements of the input sequence, which are then used as soft-attention weights to weigh the contribution of each element to the final output.

Pointer networks are designed to tackle a specific class of problems where the number of target classes in each step of the output is dependent on the length of the input, which can vary. These problems include sorting variable-sized sequences and various combinatorial optimization problems. Pointer networks address the issue of a variable-sized output dictionary by employing the mechanism of neural Attention explained before. This approach differs from previous Attention methods in literature in that it uses Attention as a pointer to select a member of the input sequence as the output, rather than utilizing Attention to blend hidden units of an encoder into a context vector at each decoder step.

Pointer networks have been shown to perform well in tasks such as code completion [Li et al., 2018] and text summarization [Chen and Bansal, 2018], where the input sequence is long and complex, and the network needs to selectively attend to parts of the input to generate the output. Pointer Networks have also been successfully applied to choice modeling for airline itinerary prediction as a way of ranking different alternatives [Mottini and Acuna-Agost, 2017]. This last work inspires us to try a pointer-based approach again in Section 3.2 in an attempt to improve these results.

Chapter 3

Architectures

3.1 H-Net: A Quantum-inspired Discrete Choice Model

3.1.1 Motivation

Our first proposed model, the H-Net architecture, is based on the QDT framework. The goal of this work is to incorporate the flexibility of Hamiltonian evolution into a deep model, creating a method for modeling the flow of probability between alternatives in a choice problem. Previous research uses this approach to model choices on a small scale but the applicability of this method to complex problems is yet to be proven [Hancock et al., 2020] [Busemeyer and Bruza, 2012c]. Moreover, this approach is interesting as the matrix exponentiation operation shown in Equation 2.1 allows for simulating the dynamics of the system through pairwise comparisons between alternatives. This can simplify the modeling process, as it only requires fitting the model for a single pairwise comparison, as then the same model can be reapplied to problems of any length. Another motivation for our work is to enhance the previous methods used in estimating the parameters of the Hamiltonian. The previous methods utilize simple functions with limited parameters or construct the Hamiltonian matrix ad hoc for the given problem. While this approach certainly has benefits in terms of explainability, it falls short when applied to complex problems where more intricate relationships between features may arise and where the flow of probabilities is not so straightforward to decode. By incorporating deep learning, we aim to improve both the expressiveness of the model and its versatility in various applications, firstly because we apply far more powerful functions and secondly because we delegate the design of Hamiltonian parameters to the network optimization process. While the prior studies are limited to using simple numerical features, our architecture has the potential to process

any type of input, be it text or even images. Additionally, our H-Net architecture considers *context* in its computation, a factor that is not addressed in previous works based on the same approach.

Finally, we note that the Hamiltonian model shares strong similarities with continuous-time Markov chains, validating its comparison with models such as PCMC-Net [Busemeyer et al., 2006] [Lhéritier, 2019]. Firstly, the Kolmogorov forward equation can be used to describe the time evolution of the probability distribution of the system, as the system transitions from one state to another.

$$\frac{\partial}{\partial t}\Phi(t) = Q\Phi(t) \tag{3.1}$$

The solution to the Kolmogorov forward equation gives the probability of being in a particular state at any given time, given the initial state and the transition rates between states.

$$T(t) = e^{tK}$$

Note how it is equivalent in shape to the Schrödinger equation (2.2) and has a similar solution (2.3). The differences are the use of complex numbers and the constraints imposed on the two matrices Q and H. The off-diagonal elements of the rate matrix Q are positive and the diagonal elements of Q are negative so that the sum of the columns is zero, which then guarantees that the columns of T sum to one, which finally guarantees that $\Phi(t)$ always sums to unity. The Hamiltonian matrix H is a Hermitian matrix ($H^\dagger = H$) so that U is a unitary matrix ($U^\dagger U = I$), which finally guarantees that $\Psi(t)$ always has unit length. Also, while the transition matrix in the classical case is left stochastic, in the quantum case (Eq. 2.4) it's doubly stochastic; that is, both the rows and columns of T sum to one. These differences cause two different dynamics in the two models. Previous literature described a Markov model probability evolution to look like a "pile of sand swept by the wind", with density slowly accumulating in specific states. A quantum Hamiltonian dynamics, on the other hand, was compared to a "wave sloshing back and forth", with density never settling for a specific state [Busemeyer and Bruza, 2012d] [Lhéritier, 2019] [Busemeyer et al., 2006].

3.1.2 Overview

The resulting H-Net architecture too shares most of the structure with PCMC-Net. The substantial differences are the use of complex values and the final computation to obtain the probability distribution of the result. This section gives an overview of the overall architecture, while the following describes in more detail each layer:

- **Input layer**: Initial layer fed with two alternatives a_i, a_j belonging to an alternative set S of n alternatives for a specific choice. Alternatives belong to a given feature space $\mathcal{F}_a$. The additional input is a set of *context* features C belonging to a given feature space $\mathcal{F}_c$.

- **Embedding layer**: comprised of embedding functions to map alternatives' features

$$\rho_{w_a} : \mathcal{F}_a \mapsto \mathbb{R}^{d_a}$$

  as well as context features

$$\rho_{w_c} : \mathcal{F}_c \mapsto \mathbb{R}^{d_c}$$

  where $d_a, d_c \in \mathbb{N}$ are the representation size of alternative and context and are defined as a hyperparameter of the architecture, and w_a and w_c are weights of the representation layer to be fitted during training.

- **Comparison layer**: The core component modeling the Hamiltonian parameters starting from the embeddings $\rho_{w_a}(a_i)$, $\rho_{w_a}(a_j)$, $\rho_{w_c}(C)$ composed of a neural network function f_{w_h} parameterized by a set of weights w_h.

$$h_{ij} = f_{w_h}\big(\rho_{w_a}(a_i), \rho_{w_a}(a_j), \rho_{w_c}(C)\big) \tag{3.2}$$

- **Hamiltonian dynamics layer**: Parameters h_{ij} of the Hamiltonian are computed across all pairs of alternatives using a cross-product and concatenation operation as in Section 2.1.3 and are then combined to form H. The corresponding unitary U is computed using a matrix exponential and applied to an initial state $|s\rangle$ to obtain the final state $|s_f\rangle$ representing the probability distribution over all S given C. In particular, probability $P(i|S, C)$ is obtained by taking the squared norm of the i-th element of $|s_f\rangle$.

$$H_{ij} = h_{ij} \qquad U = e^{-itH}$$

$$P(i|S, C) = |\langle i|s_f\rangle|^2$$

Parameters w_c, w_a and w_h are all trained using dropout. Dropout is a regularization technique used to prevent overfitting. During training, dropout randomly deactivates (or "drops out") some parameters in the network, meaning they are effectively ignored for that gradient descent iteration. This random deactivation encourages each weight to learn more robust features by preventing its reliance on

specific other parameters. Negative Loglikelihood (NLL) is used as the loss function throughout all experiments as in previous literature [Lhéritier, 2019]. NLL is computed as:

$$NLL = -\frac{1}{|\mathcal{T}|} \sum_{(S,C)\in\mathcal{T}} \log P(Y_S|S,C)$$

with $\mathcal{T}$ either training or test set, S and C respectively the aforementioned alternative set and context for a given choice, and Y_S the index of the actual choice.

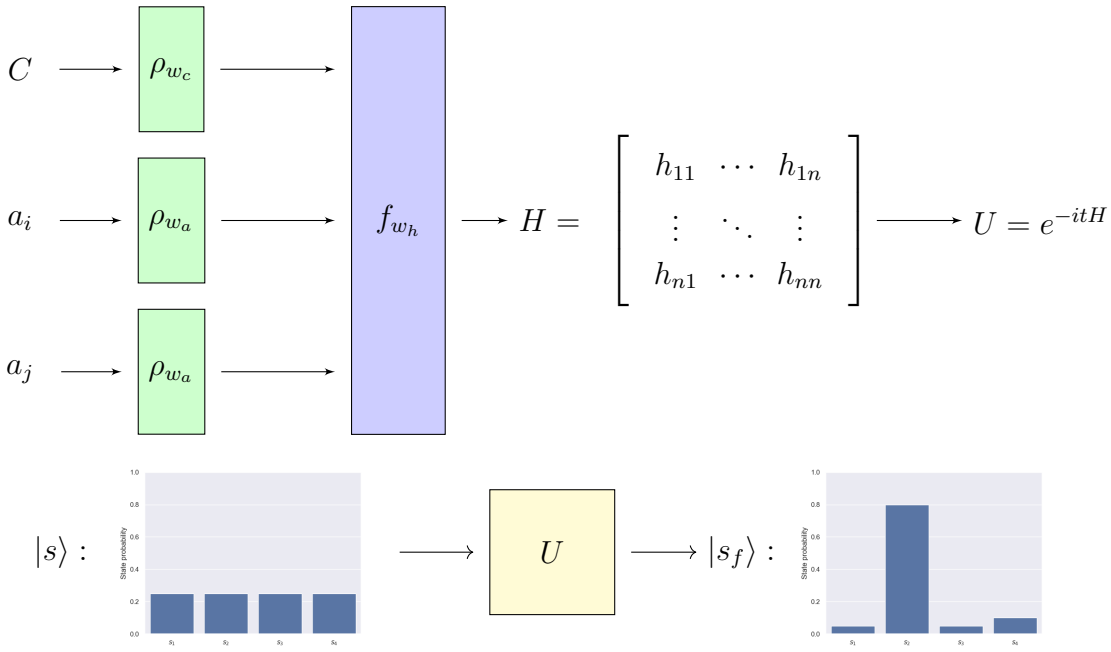


Figure 3.1: Overview of H-Net's full architecture. Embedding layer is denoted in green, comparison layer in blue, and Hamiltonian dynamics layer in yellow

3.1.3 Embedding

The role of embedding functions is to map an input of possibly variable length to a finite-dimensional space of fixed length. This process performs an initial bottleneck and acts as a feature selector during training. In H-Net, we implement two distinct embedding functions: one for single alternatives and one for context. ρ_{w_a} maps a single alternatives a_i from a feature space $\mathcal{F}_a$ to a latent space $\mathbb{R}^{d_a}$, while ρ_{w_c} takes the full contextual information of the choice from space $\mathcal{F}_c$ to $\mathbb{R}^{d_c}$.

In our implementation, ρ_{w_a} and ρ_{w_c} have much the same structure: the input vector x (be it an alternative or the full context) is split into its numerical and

categorical features x_n and x_c. Numerical features are standardized (a function we denote s), while categorical features are converted in numerical ones via PyTorch's *embedding* layer (denoted e). After this step, the two numerical representations are concatenated (denoted $\oplus$). Note that e internally builds a dictionary with weights for every observed configuration of x_c, therefore, a potentially high number of weights is required to create the mapping.

$$\rho_w(x) = s(x_n) \oplus e(x_c)$$

While this is our solution, many more are available that process numerical and categorical features in different ways.

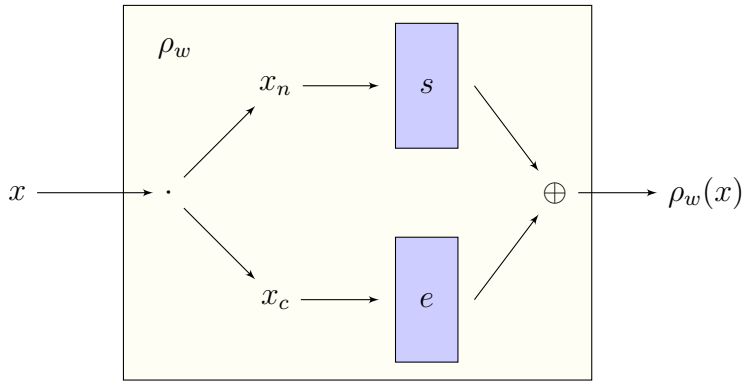


Figure 3.2: Detailed view of H-Net's embedding layer

3.1.4 Comparison

This layer uses the embedded features to estimate a component of the Hamiltonian needed to construct the U operator. Comparisons are done in a pairwise manner between every couple of alternatives i and j. Previous works in QDT compute the h_{ij} value with functions that do not consider contextual information and instead simply rely on a difference between the features, transformed by some non-linear yet simple comparison functions [Hancock et al., 2020]. The behavior is similar to the comparison layer PCMC-Net already detailed in section 2.1.3, however, the key difference lies in the fact that the returned value is a complex number. f_{w_h} is composed, in our implementation, of a fully connected network n_{w_h} that returns two values $\Re(h_{ij})$ and $\Im(h_{ij})$, and of a step that combines them to form h_{ij}. The number of layers and the activation function of n_{w_h} are application-specific. A way of finding an optimal choice for both is discussed in Chapter 4.

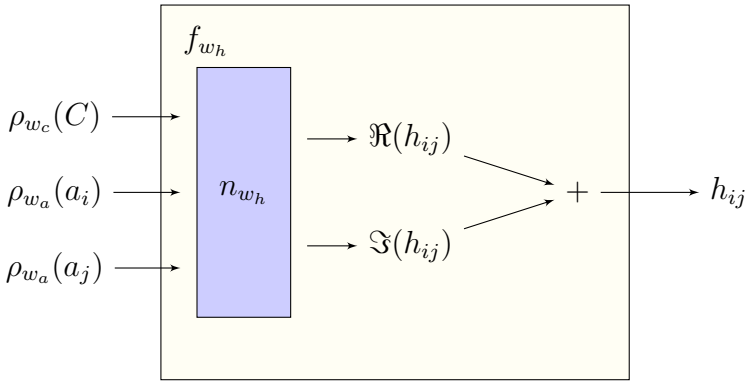


Figure 3.3: Detailed view of H-Net's comparison function

3.1.5 Hamiltonian dynamics

This layer gathers the results of repeated calls of f_{w_h} on all pairs of alternatives (obtained with a cross product as in Section 2.1) and uses them to compose the H matrix. An advantage of parametrizing any comparison with an operation dependent on a single set of weights w_h is that H-Net can tackle problems of any number $|S|$ of alternatives. Since H is Hermitian, one only needs to estimate the upper $\frac{|S|(|S|-1)}{2}$ elements in the upper triangular part, computing the remaining ones from those. Concretely, it means that function f_{w_h} (Eq. 3.2) is called only $O\left(\frac{|S|}{2}\right)$ times. Hermitianity is hard to enforce with optimization constraints. It can, however, be enforced by construction. Using the definition of Hermitian matrix, we compute only the upper triangular part of H ($i < j$) and compute the lower portion directly from there:

$$h_{ij} := \begin{cases} f_{w_h}(a_i, a_j) & \text{if } i < j \\ \overline{f_{w_h}(a_j, a_i)} & \text{if } i > j \\ \Re\big(f_{w_h}(a_i, a_j)\big) & \text{if } i = j \end{cases}$$

Matrix H is then used to construct the unitary operator U through a matrix exponential operation as in Equation 2.3. In literature, the time parameter t is set equal to $\frac{\pi}{2}$. We decide instead to also optimize this parameter during training. Finally, the unitary U is applied on a state vector $|s\rangle$ to obtain the informed state $|s_f\rangle$, and the values of this state are squared to obtain probability distribution on the alternatives:

$$|s_f\rangle = U\,|s\rangle = e^{-itH}\,|s\rangle \qquad P(i|S,C) = |\,\langle i|s_f\rangle\,|^2$$

In QDT, $|s\rangle$ is a unitary vector representing the initial state of the decision maker, this encompasses both individual (e.g. age, gender, income, ...) and contextual

information (e.g. date, weather, ...). A first possibility is to estimate it directly using contextual information C using another nonlinear function f_{w_s}:

$$|s\rangle = f_{w_s}(C)$$

However, we choose another approach, which is to set $|s\rangle$ to a uniform distribution leaving U to enact the complete transition from uniform to final distribution.

$$|s\rangle = \frac{1}{|S|}[1,1,\ldots]^T$$

The reason is twofold: firstly, the matrix exponential of a Hermitian matrix is a universal generator for unitary matrices, U can transform a generic unitary vector into any other unitary vector; secondly, the information contained in C can already be injected into the state by f_{w_h}. We therefore integrate all the complexity of modeling the contextual stage into f_{w_h}, and the Hamiltonian generated from it can take the state directly from an uninformed state to a final one. This is also more expressive because the information from C can be combined with information from a_i and a_j in a non-linear way. By using a single transformation that takes the state from an uninformed state to a context and alternative-informed one, we increase expressivity, avoid the need for separate functions and reduce the number of parameters of the overall architecture. Our way of constructing $|s\rangle$ is not the only one. A more evolved approach that takes into consideration multiple choices by the same user could update $|s\rangle$ for a specific user after every choice. We discuss the feasibility and expressiveness of a similar solution in Section 5.2.

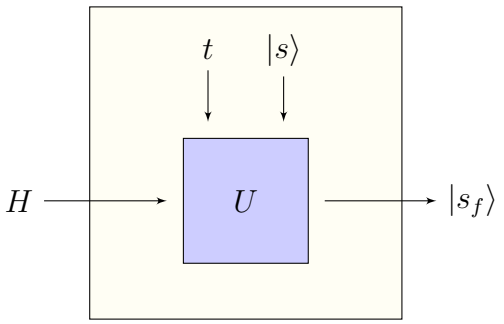


Figure 3.4: Detailed view of the Hamiltonian dynamics layer

3.1.6 SimpleH

Previous approaches either estimate H directly as a parameter in the optimization process [Broekaert and Busemeyer, 2018] [Broekaert and Busemeyer, 2017] [Yearsley and Busemeyer, 2016b], or compute it starting from features

[Hancock et al., 2020]. Since our approach acts on the assumption that more complex functions can better capture comparisons between alternatives and thus produce Hamiltonians that can closely model the decision process, we consider an established and non-neural network-based function as a reference. The choice lands on the approach described in [Hancock et al., 2020], which uses a linear difference function. This model uses the same architectural choice as in H-Net, the only differences being the time t, which is fixed at $\frac{\pi}{2}$, and the comparison function f_{w_h}, which is no longer a stack of non-linear fully connected layers but is instead a simple linear function:

$$f_{w_h}\big(x_i, x_j\big) = \delta + w_h^T(x_i - x_j)$$

where δ is a constant parameter optimized with the network and $w_h \in \mathbb{R}^{d_a}$ is a vector of weights. As in the original paper, all information regarding context C is not taken into consideration.

3.2 Pointer Transformer: An Attention-based Discrete Choice Model

3.2.1 Motivation

The use of Recurrent Neural Networks (RNNs) for processing sequential data remains a common approach in deep learning. However, the standard RNN architecture has limitations when it comes to capturing long-term dependencies in sequential data. To overcome this, researchers have introduced the Attention mechanism into RNNs, which helps alleviate the vanishing gradient problem and effectively handles longer sequences [Bahdanau et al., 2015]. Recently, the focus has shifted towards removing the recurrency aspect altogether and relying solely on the Attention mechanism. This has led to the development of the Transformer architecture, which is specifically designed to handle sequential data in a more effective manner. The Transformer architecture has been highly successful and surpasses the performance of traditional RNNs in a variety of natural language processing tasks [Vaswani et al., 2017] [Hernández and Amigó, 2021]. Prior research in choice modeling uses Transformers for determining the parameters of an MNL model [Phan et al., 2022]. However, as far as we know, there have not been any efforts to use Transformers to estimate the utilities of alternatives directly. In the upcoming sections, we present an implementation of a Pointer Transformer. Our aim is to examine whether utilizing only the Attention mechanism through a Transformer can outperform its recurrent alternative, the Pointer Network.

3.2.2 Overview

In order to have a fair comparison, Pointer Transformer shares many implementation choices with other methods we describe. Since Transformers employ fully connected layers that need to be of fixed length, we lose the capability of applying the same model to problems of any size $|S|$. The maximum number of tokens processed by the model is an architectural choice, and is set equal to the maximum $|S|$ found in the training set. A major difference of this approach from the previous ones we examined is that, while PCMC-Net and H-Net act first in a pairwise manner and then combine these intermediate results (Sections 2.1.3, 3.1), Pointer Transformer performs it as a comparison of all alternatives together. The Attention mechanism mediates this comparison by modeling long distance relationships between all alternatives, multiplying query, key and value representations with the ones from all other alternatives. Pointer Transformers don't suffer from vanishing gradients as Pointer Networks do since there is no recursion involved, but lose the size invariance other models have.

- **Input layer**: Initial layer fed with all n alternatives $a_1, a_2, \ldots, a_n$ belonging to an alternative set S for a specific choice. As before, alternatives belong to a given feature space $\mathcal{F}_a$ and are accompanied by contextual information C belonging to space $\mathcal{F}_c$.

- **Embedding layer**: As in section 3.1.2, this layer is comprised of embedding functions to map alternatives' features

  $$\rho_{w_a} : \mathcal{F}_a \mapsto \mathbb{R}^{d_a}$$

  as well as context features

  $$\rho_{w_c} : \mathcal{F}_c \mapsto \mathbb{R}^{d_c}$$

  where $d_a, d_c \in \mathbb{N}$ are the representation size of alternative and context, and w_a and w_c are weights of the representation layer. The difference here is that context embedding $\rho_{w_c}(C)$ is concatenated to each alternative to form a token t_i of size $d_a + d_c$.

  $$t_i = \rho_{w_c}(C) \oplus \rho_{w_a}(a_i)$$

  The redundancy on the context is introduced in order to have the resulting key, query and value representations include also the contextual information for each token. All tokens are then concatenated together to form the input of the Attention layer. In case of alternative sets of size smaller than n, the missing alternatives are represented with a token of only zeros (padding).

In either case, a value identifying the position of the token in the choice set is summed to the embedding. This is called *positional embedding* and is done to give the model contextual information about the position of the alternative; this is relevant in contexts where the order may affect the final result, or where the choice set is fixed and alternatives in position i always have some property.

- **Comparison layer**: Several multi-headed Attention layers perform a many-to-many comparison of all alternatives. Inputs and outputs represent input tokens and contextualized representations respectively, and they both have length $n(d_a + d_c)$.

- **Aggregation layer**: Values of each token are first averaged through layer μ to provide a single summary value for each token. Then all the means are passed through a final fully connected layer nn_{agg} with a softmax activation. This is done to normalize the values into the [0,1] range. Each value constitutes the final probability $P(i|S, C)$ of the corresponding alternative.

Every weight is trained with dropout, similar to the approach taken in H-Net. Loss, too, is computed in the same way as H-Net.

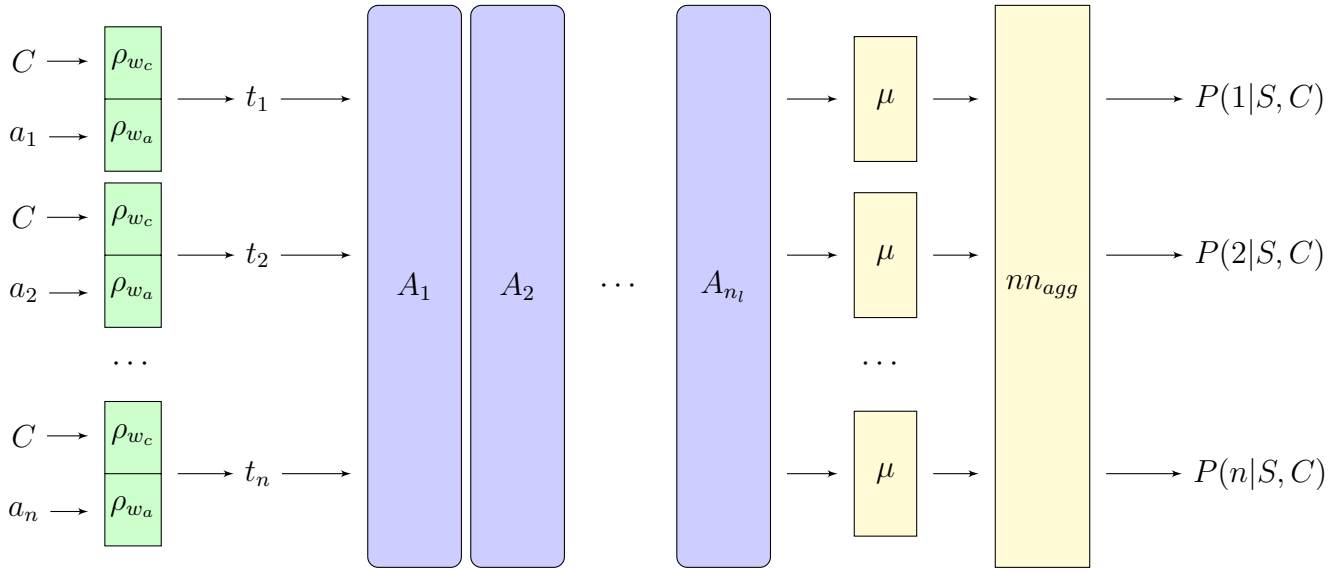


Figure 3.5: Pointer Transformer architecture. Embedding layer is denoted in green, comparison layer in blue, and aggregation layer in yellow

3.2.3 Comparison

This layer is composed by n_l consecutive Attentions $A_1, \ldots, A_{n_l}$ that perform a many-to-many comparison between all alternatives. Each layer A_i is composed in turn by a multi-headed Attention with n_h heads (denoted as h), and a fully connected layer (denoted as nn). The set of weights parametrizing the layer as a whole is w_t, omitted for the sake of readability. The inputs are n vectors of $d_a + d_c$ elements each, comprised of the embedded context concatenated with an embedded alternative. The outputs are likewise contextualized tokens that contain information about the comparison. The fully connected layer nn serves the purpose of both reshaping and introducing a data bottleneck after the Attention layer: since the Attention heads collectively return an output of size $n_h \times n(d_a + d_c)$ some form of aggregation is needed to get outputs of the same shape as the inputs, this also selects the useful information from each head. A schematic view of an Attention block A_i is shown in Figure 3.6

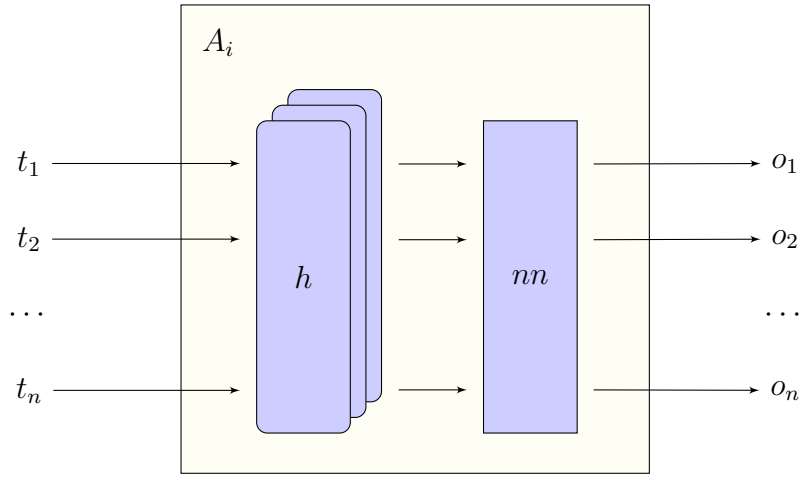


Figure 3.6: Detailed view of a multi-headed Attention block A_i

Chapter 4

Applications

4.1 Methodology

This section describes our approach to training models on all three datasets. We compare performance for an MNL model, a SimpleH model as detailed in Section 3.1.6, a PCMC-Net as described in Section 2.1.3, and for our novel approaches H-Net and Pointer Transformer. Note that, for the MNL model, we decide to use the same embedding technique we used in other implementations. While the training procedure remains the same for all models, the hyperparameters used differ between applications and models and are listed in the respective section. We first perform a train-validation-test split, the details of which vary depending on the dataset structure. They are described in Sections 4.2.1, 4.3.1, and 4.4.2. After partitioning the data, we conduct a hyperparameter tuning phase, assessing numerous architectural and optimization choices. During this phase, the model is trained on the training set and evaluated on the development set. Once we discover an appropriate set of optimal hyperparameters, we commence final training on both the training and development sets, saving the weights of the architecture. We train MNL and SimpleH models for 50 epochs, PCMC-Net and H-Net for 100 and Pointer Transformer for 200. These values are found by observing the average training time over many experiments and should be considered as a rule of thumb. Lastly, we compute various metrics on the test set we left unseen until now. The metrics are:

- **Negative Loglikelihood**: already used as a loss during training, as described in Section 3.1.2. The closer to 0, the better the model.

- **Top N**: proportion of choice sessions where the actual choice was one of the top N rated options. The choice of N varies between applications as they all have a different alternative set size.

- **Mean Reciprocal Rank (MRR)**: Used as the evaluation metric during the RecSys 2019 Challenge, it provides a "softer" indicator of performance compared to Top N. It is defined as

$$MRR = \frac{1}{|\mathcal{T}|} \sum_{(S,C)\in\mathcal{T}} \frac{1}{rank(Y_S|S,C)}$$

  where $rank(Y_S|S,C)$ is the rank position of alternative Y_S as computed using the given model with S and C as inputs. MRR is a value between 0 and 1, the higher it is, the better recommendations the model is providing

4.1.1 Optimizer

The optimizer we choose for performing gradient descent is Adam. Adam (ADAptive Moment estimation) is an extension of stochastic gradient descent that has recently seen broader adoption for deep learning. It has a small computational and memory footprint, and is shown to work well in problems with very noisy gradients, a common occurrence in deep learning [Kingma and Ba, 2015]. The only optimization parameters we tune are the learning rate and the number of epochs. Other optimizers such as RMSProp and AdaGrad have been tested initially, but Adam has proven to be superior for our applications.

4.1.2 Hyperparameter tuning

In deep learning, finding the optimal configuration of hyperparameters for a given problem is not a straightforward task. The search space of possible hyperparameters is exponentially large, and the models can take a prohibitively long time to learn. In our applications, too, we encountered these problems. Pointer Transformer, for instance, has as many as eight different hyperparameters that must be tuned. Given the complexity of this task, a Bayesian optimization approach is often necessary to find the best hyperparameter configuration. While describing the approach in detail is beyond the scope of this thesis, we can say that Bayesian optimization involves creating a probabilistic model of the objective function and iteratively updating this model to find the best hyperparameters. This approach is particularly suitable when evaluating the objective function is computationally expensive, as it allows us to make informed decisions about where to evaluate the function next based on the model. We use the *GPyOpt* library in our implementation as it provides a plug-and-play way of performing Bayesian optimization [authors, 2016].

4.1.3 Hardware

All experiments are either run locally or remotely on a Docker container in a more powerful machine. The local device is a Dell Precision 3570 with 32 GB RAM and a 12th Gen Intel i7-1280P processor. Remote runs make use of a Linux machine with two Intel Xeon E5-2643 v4 CPUs, 251 GB of RAM, and are accelerated by four Tesla K40c GPUs with 12 GB of VRAM.

4.2 Synthetic Stated Preferences

4.2.1 Dataset Description

We first apply our methods on a synthetic dataset containing travel mode choices for 500 travellers. It's a small dataset of total size 560KB and provides us with an initial starting point and sanity check for newly implemented models. For each individual, the data contains 14 stated preference (SP) inter-city trips, where the possible alternatives are car, bus, airplane and train, of which at least two are available. The journey options are described by numerical features such as access time, travel time (in minutes), and cost (in £). The data contains an additional categorical quality of service attribute for airplane and train trips. It can take three levels: no frills, wifi available, or food available. The data then also contains two revealed preference tasks per person, using the same alternatives as those available on the SP journey for that person, but containing no categorical information on the quality of service. We choose to remove the revealed preference tasks from the dataset in order to have more uniform data. For each individual, the dataset also contains information on gender, whether the journey was a business trip or not, and the individual's income. Individuals are identified by a code we choose to remove in order to avoid data leakage and because it does not add any meaningful information about the choice. This helps the model focus solely on the individual's attributes. We perform a 72-8-20 train-development-test split based on the individual's ID.

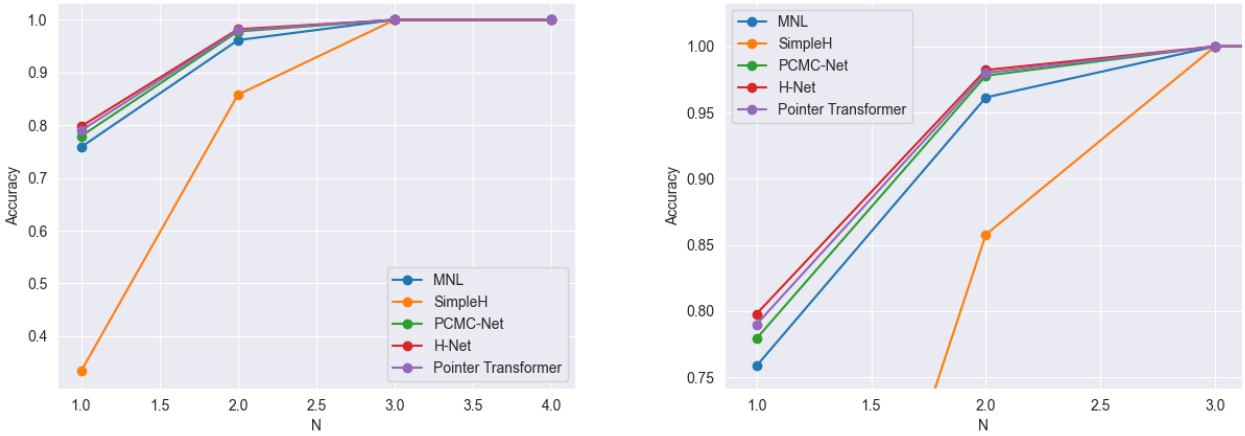
The data is taken from examples of the *Apollo Choice Modelling*[1] R package [Hess and Palma, 2019].

4.2.2 Results

In our first experiment, we notice all methods have similar performance with the sole exception of SimpleH. H-Net performs slightly better, while MNL shows lower

[1]Apollo website: http://www.apollochoicemodelling.com/index.html

performance compared to its deep counterparts. MNL performs fairly well and doesn't make use of contextual information, this can be evidence of the context not being particularly informative for this method. Table 4.7 shows how Pointer Transformer has a large number of parameters compared to other methods, this is surprising since the larger size doesn't show any particular advantage. This is possibly caused by the small size of the dataset that doesn't allow the Pointer Transformer to train adequately.



(a) Full plot (b) Closeup

Figure 4.1: SP Top N Accuracy comparison

Model	NLL	Top 1	MRR
MNL	0.547	0.759	0.872
SimpleH	0.925	0.333	0.643
PCMC-Net	0.510	0.779	0.886
H-Net	**0.474**	**0.798**	**0.896**
Ptr Transf	0.484	0.790	0.892

Table 4.1: Results on SP dataset

4.3 Airline Itinerary

4.3.1 Dataset Description

The Airline Itinerary (AI) dataset is composed of anonymized booking data from different airlines collected by the Global Distribution System (GDS) Amadeus,

Hyperparameter	Range	Best value
MNL		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=0...4}$	2
SimpleH		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-2}
Batch size	$\{2^i\}_{i=0...4}$	2
PCMC-Net		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-2}
Batch size	$\{2^i\}_{i=0...4}$	8
Nodes per layer	$\{2^i\}_{i=1...5}$	16
Hidden layers	{1,2,3}	2
Activation	{ReLU, Sigmoid, Tanh, LeakyReLU}	LeakyReLU
H-Net		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-2}
Batch size	$\{2^i\}_{i=0...4}$	4
Nodes per layer	$\{2^i\}_{i=1...5}$	32
Hidden layers	{1,2,3}	1
Activation	{ReLU, Sigmoid, Tanh, LeakyReLU}	LeakyReLU
Ptr Transf		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=0...4}$	1
Depth (n_l)	{1,2,3}	2
Heads (n_h)	{1,2,3}	2
Nodes (intermediate layers)	$\{2^i\}_{i=0...4}$	8
Nodes (final layer)	$\{2^i\}_{i=0...4}$	8
Dropout (NN)	$\{i/10\}_{i=0...5}$	0.4
Nodes (Embedding)	$\{i/10\}_{i=0...5}$	0.3

Table 4.2: Hyperparameters evaluated on SP dataset

cross-referenced with a large number of search logs from Amadeus' partners. A GDS is a network system used by travel agencies and other travel-related businesses to access real-time information, availability and pricing for airlines, hotels, car rentals, and other travel-related services, as well as book and sell tickets for multiple airlines. Booking data alone, containing information such as origin, destination, and dates is insufficient to fully understand choice behavior. Search logs complement this information by providing information about the market context. In other words, AI contains the travel alternatives that the customer was provided with at the time of booking, which includes information such as different airlines,

flight numbers, time of flights, and prices.

Every user session (set of alternatives) contains a variable number of alternatives ranging from 1 to 50, of which exactly one is a booking. User sessions are sorted by increasing price. Since this is the way most search engines present results to customers, we assume this ordering also reflect which alternatives were viewed first and wich last. We have, however, no way of telling which alternatives are actually seen by the user. Since the original data was of exceptionally high volume (100s of GB), the dataset is limited to a set of European origins and destinations, and only contains requests concerning round trips. The final size of the dataset is 187MB. For legal reasons, no personal features (e.g. gender or age) are present in the dataset. Moreover, the only feature engineering carried out is the conversion of the departure and arrival time from a timestamp to a time of day [Mottini and Acuna-Agost, 2017]. We finally note that most of the flight options are observed only once in the whole dataset [Lhéritier, 2019]. The train-development-test split is the same used in previous works using the same data and consists of a 72-8-20 split on the choice sessions.

4.3.2 Results

On the Airline Itinerary dataset, we observe a clear separation between deep and shallow methods (Figure 4.2). PCMC-Net and H-Net perform comparably well and outperform the Transformer-based model, despite their well-known performance. SimpleH underperforms again, showing how limiting a linear comparison is with respect to the non-linear one performed in H-Net. For H-Net and PCMC-Net, remarkably small models are chosen by Bayesian optimization as can be seen in Table 4.4, hinting that these techniques can generalize well. Pointer Transformer slightly underperforms despite its large number of parameters (Table 4.7), while PCMC-Net and H-Net are comparable in both size and performance.

Model	**NLL**	**Top 1**	**Top 5**	**MRR**
MNL	2.516	0.205	0.588	0.380
SimpleH	3.040	0.152	0.444	0.297
PCMC-Net	**2.216**	**0.289**	0.692	**0.468**
H-Net	2.265	0.281	**0.695**	0.464
Ptr Transf	2.468	0.259	0.667	0.439

Table 4.3: Results on AI dataset

Hyperparameter	Range	Best value
MNL		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-5}
Batch size	$\{2^i\}_{i=0...4}$	8
SimpleH		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-2}
Batch size	$\{2^i\}_{i=0...4}$	4
PCMC-Net		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=0...4}$	1
Nodes per layer	$\{2^i\}_{i=5...9}$	64
Hidden layers	{1,2,3}	2
Activation	{ReLU, Sigmoid, Tanh, LeakyReLU}	ReLu
H-Net		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=0...4}$	2
Nodes per layer	$\{2^i\}_{i=5...9}$	64
Hidden layers	{1,2,3}	2
Activation	{ReLU, Sigmoid, Tanh, LeakyReLU}	Sigmoid
Ptr Transf		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=0...4}$	16
Depth (n_l)	$\{1,\ldots,9\}$	2
Heads (n_h)	$\{1,\ldots,4\}$	1
Nodes (intermediate layers)	$\{2^i\}_{i=3...9}$	8
Nodes (final layer)	$\{2^i\}_{i=3...9}$	64
Dropout (NN)	$\{i/10\}_{i=0...5}$	0.5
Nodes (Embedding)	$\{i/10\}_{i=0...5}$	0.2

Table 4.4: Hyperparameters evaluated on AI dataset

4.4 RecSys Challenge 2019

4.4.1 RecSys Challenge

The RecSys Challenge is an annual event proposed during the RecSys conference. It aims to foster research in the field of recommender systems by providing a platform for researchers and practitioners to showcase their models and compete against others. The data provided for the challenge is usually obtained from a real-world application, and the participants are required to develop a model that can accurately predict the user's preference. The 2019 challenge consists of developing

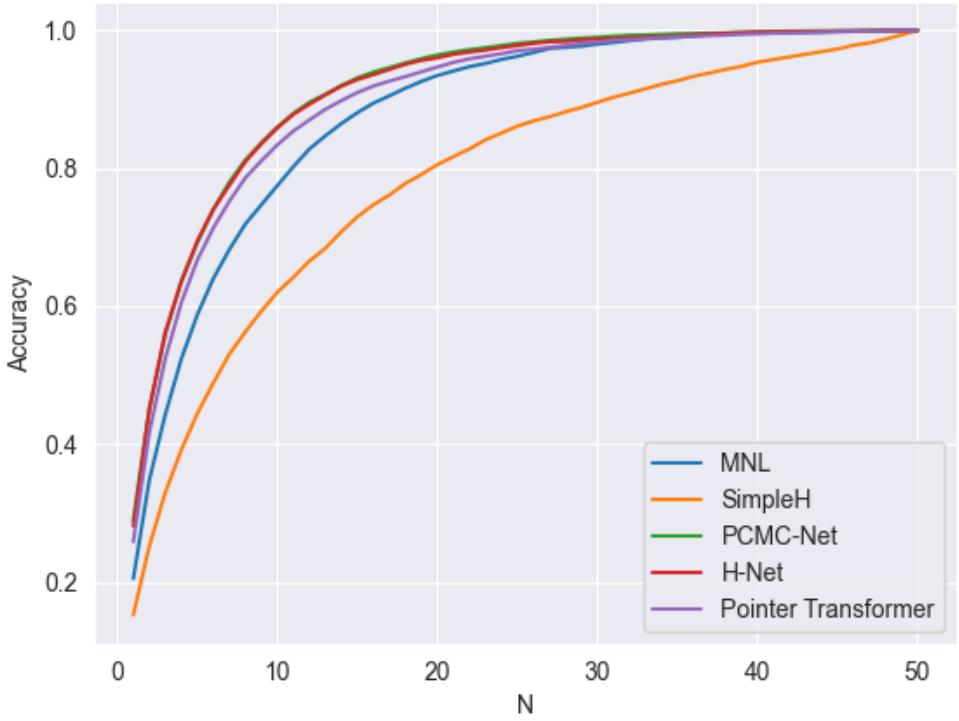


Figure 4.2: AI Top N Accuracy comparison. PCMC-Net (green) and H-Net (red) almost always overlap

a session-based and context-aware recommender system to predict the preferences in a list of accommodations based on the actions of the user. A total of 575 teams participated in the challenge. We do not participate directly in the challenge but use the same data to validate our models and observe how they compare against those developed by the participating teams. The dataset used in the challenge is the largest we tackle in our research, and its analysis gives us an indication of how well our models would fare for a large recommender system in a production environment.

4.4.2 Dataset Description

The RecSys Challenge data (RSC) provided consists of interaction logs on the trivago platform recorded during search sessions. The logs cover a period of 8 days, from 1st to 8th of November 2018 UTC, and record the actions of almost 950k unique users across 55 different countries. Bookings are for accommodations worldwide, with no particular restriction of market or type of accommodation.

This results in more than 1.2 million sessions overall. Most users only perform one search session. Moreover, we are provided with metadata concerning 930k unique accommodations, to be used as alternative-specific information. Recorded actions are:

- **clickout item**: user makes a click-out on the item and gets forwarded to a partner website. Other items that were displayed to the user and their associated prices are also listed.
- **interaction item rating**: user interacts with a rating or review of an item.
- **interaction item info**: user interacts with item information.
- **interaction item image**: user interacts with an image of an item.
- **interaction item deals**: user clicks on the view more deals button.
- **change of sort order**: user changes the sort order.
- **filter selection**: user selects a filter.
- **search for item**: user searches for an accommodation.
- **search for destination**: user searches for a destination.
- **search for poi**: user searches for a point of interest (POI).

Each log has additional data such as destination, active filters, and type of device of the user. The task is to predict which item was selected in a specific clickout action among a set of exactly 25 alternatives the user was presented with ($n = 25$). The training set, as it was provided during the challenge, contains user actions up to a specified timestamp, called split date, corresponding to 23:59 on the 7th of November. The remaining actions are considered our test set and are used during the evaluation of the models. A schematic of the problem setting and the separation of the data is shown in Figure 4.3.

4.4.3 Feature Engineering

The RSC dataset is in a raw format containing data as logs. A winning approach for other competitors is to use several aggregate features rather than modeling the actions with sequence-based learning. We take inspiration from these insights and perform a complex feature engineering step to convert the logs to a multilinear tabular format (one row per alternative, each session identified by an ID). Some examples of aggregates we compute are the number of interactions per type, user,

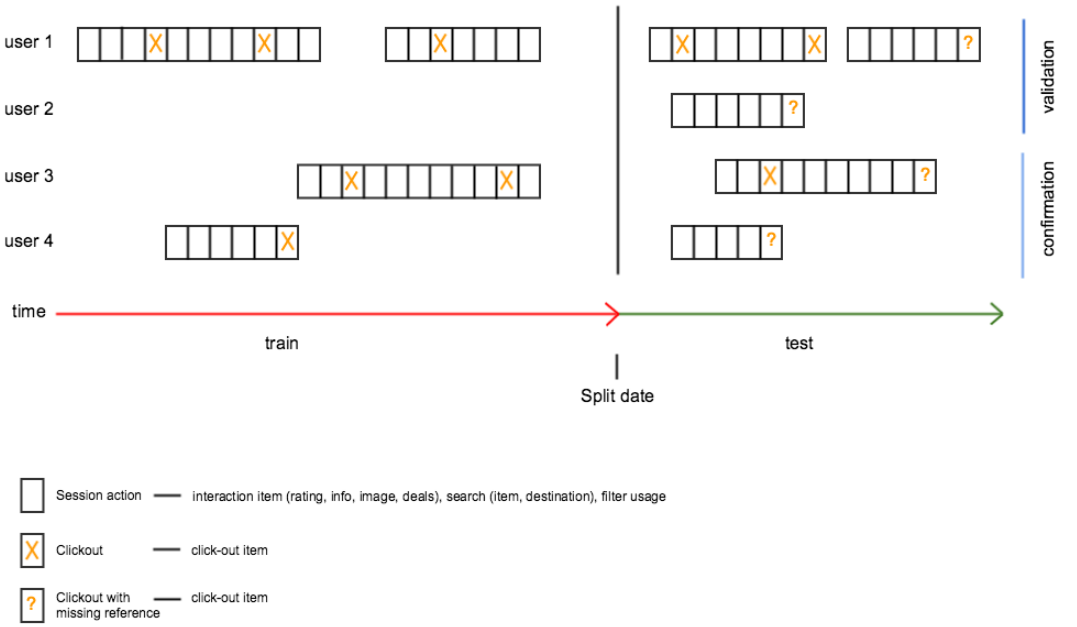


Figure 4.3: RSC task setting

and session, the number of clicks per alternative, the time difference with the previous click, and the relative price of an article in a choice set. A complete list is available in Appendix A.

The first obstacle we face is the sheer amount of temporary data created during processing: many aggregates require expanding each clickout item action into its 25 alternatives and merging large tables over it. A task our hardware, however good, cannot accomplish. We resort to uniformly selecting 150k users out of the 950k, the maximum our hardware can handle. The users are equally distributed among the total set of train and test combined. This is a reasonable assumption and does not cause data leakage, as many recommender systems do use prior information about known users. Solutions based on swapping and lazy evaluation that could mitigate the memory problem, such as the use of the Dask framework, have been implemented for small and expensive parts of the feature engineering process but not for the whole pipeline due to complexity and time constraints.

A second hurdle is the high cardinality of the destination cities. One-hot encoding creates a prohibitively high number of features. Instead, we provide the model with coordinates for every city in order for it to learn some notion of space. City coordinates are gathered from the GeoNames database[2], the airport-codes

[2]GeoNames website: https://www.geonames.org/

repository[3], the simplemaps database[4], as well as internal Amadeus databases, and merged with the original data.

A third and critical problem is how to avoid data leakage while still constructing meaningful aggregate features. Aggregates need to be computed also for the test set, this means that at some point in the process, train and test sets need to be processed together. This carries the risk of using information from the test set in the training phase if one performs aggregations in a naive way. Another subtle mistake is to use "future" data in computing the aggregates: since data is ordered by timestamp, aggregating features on some moment in time using data from a further moment in time is equivalent to reading into the future. This gives a distorted perception of what the real capabilities of the model would be, once put in a production environment. To tackle this, we perform all our aggregation in a cumulative and time-aware way, so that no data ever propagates back in time. An illustration of this problem and its solution can be found in Figure 4.4

Dataset	ID	Timestamp	Action	Total clicks per user
Train	AAA	1	Click	4
	BBB	2	Click	1
	AAA	3	Search	4
	AAA	4	Click	4
	BBB	5	Filter	1
Test	CCC	6	Search	2
	CCC	7	Click	2
	CCC	8	Click	2
	AAA	9	Click	4
	AAA	10	Filter	4
	AAA	11	Click	4
	AAA	12	Search	4

Dataset	ID	Timestamp	Action	Cumulative clicks per user
Train	AAA	1	Click	0
	BBB	2	Click	0
	AAA	3	Search	1
	AAA	4	Click	1
	BBB	5	Filter	1
Test	CCC	6	Search	0
	CCC	7	Click	0
	CCC	8	Click	1
	AAA	9	Click	2
	AAA	10	Filter	3
	AAA	11	Click	3
	AAA	12	Search	4

Figure 4.4: Aggregate problems in the RSC dataset. Data leakage is shown in orange, anticausal relationships in yellow, and the corrected lines in green

A last tricky problem is how to handle categorical data pertaining to active filters and alternative specific characteristics. Both sets of features encompass qualities pertaining to infrastructure and qualities of accommodations, such as the number of stars, accessibility, WiFi, presence of car parking, etc. These features have high cardinalities (hundreds of different filters and characteristics available) and would result in very large models if converted to one-hot encoded features. We perform dimensionality reduction by means of sklearn's *FeatureAgglomeration* class, which builds new features by performing hierarchical clustering. We apply this process on filters and accommodation characteristics separately and go from hundreds to just 40 features.

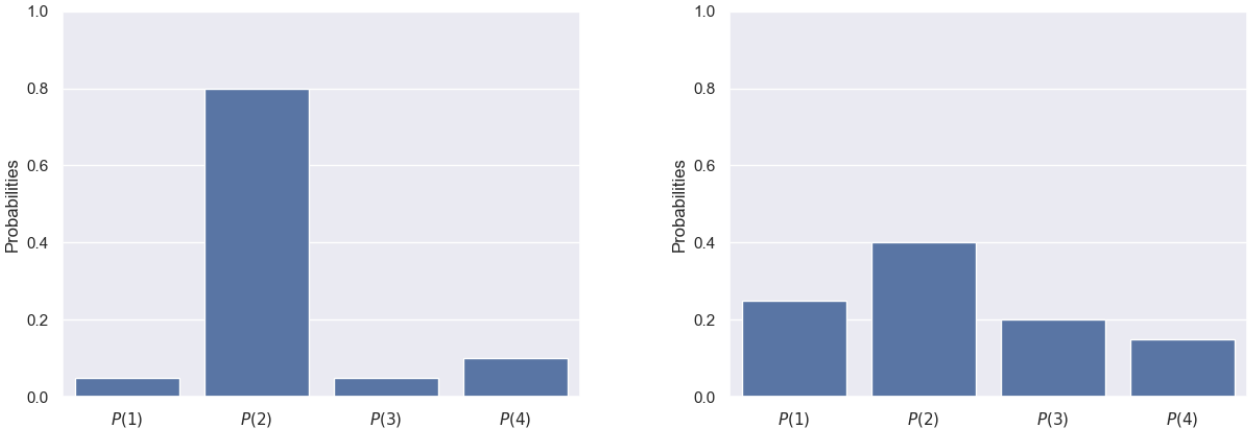
[3]airport-codes repository: https://github.com/datasets/airport-codes

[4]simplemaps website: https://simplemaps.com/

The final size of the parsed dataset is 1.6 GB for the training part, 199 MB for the development part and 603 MB for the test part. In this case, we define the test set based on the split date, and split the remaining data into train and development sets using a timestamp. Specifically, we assign the first 90% of the timestamps to the train set and the last 10% to the development set.

4.4.4 Results

Methods applied to RSC show a similar behavior to the AI case, with the PCMC-Net and H-Net having the best performance without a clear winner, Pointer Transformer being slightly behind, and MNL and SimpleH in the last positions. Here we see a peculiar phenomenon: the model with the lowest loss is not necessarily the best-performing one. This can be explained by the more lossy model being more "undecided" but giving the right answer nonetheless. A visual explanation can be seen in Figure 4.5. Regarding model sizes (Table 4.7), we notice this time the Pointer Transformer is of comparable size to the other deep models, mainly thanks to a smaller final layer. While the best MRR of our models is far from the top score obtained via gradient boosting methods (0.686), we nonetheless significantly beat the baseline which is based on the frequency of alternatives chosen by other users (0.288).



(a) Sure

(b) Unsure

Figure 4.5: Loss difference explained. While both figures predict the same alternative, 4.5b is less certain about the true answer and therefore has a higher loss

Model	NLL	Top 1	Top 5	MRR
MNL	2.686	0.238	0.606	0.409
SimpleH	3.005	0.116	0.394	0.263
PCMC-Net	**2.447**	0.311	0.632	0.462
H-Net	2.451	**0.312**	**0.642**	**0.464**
Ptr Transf	2.605	0.297	0.623	0.450

Table 4.5: Results on RSC dataset

Hyperparameter	Range	Best value
MNL		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-2}
Batch size	$\{2^i\}_{i=6...10}$	128
SimpleH		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-1}
Batch size	$\{2^i\}_{i=6...10}$	256
PCMC-Net		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=6...10}$	64
Nodes per layer	$\{2^i\}_{i=5...9}$	32
Hidden layers	{1,2,3}	1
Activation	{ReLU, Sigmoid, Tanh, LeakyReLU}	LeakyReLU
H-Net		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=6...10}$	64
Nodes per layer	$\{2^i\}_{i=5...9}$	64
Hidden layers	{1,2,3}	2
Activation	{ReLU, Sigmoid, Tanh, LeakyReLU}	Sigmoid
Ptr Transf		
Learning rate	$\{10^{-i}\}_{i=1...6}$	10^{-3}
Batch size	$\{2^i\}_{i=6...10}$	128
Depth (n_l)	$\{1,\ldots,4\}$	2
Heads (n_h)	$\{1,\ldots,3\}$	2
Nodes (intermediate layers)	$\{2^i\}_{i=1...4}$	8
Nodes (final layer)	$\{2^i\}_{i=1...4}$	4
Dropout (NN)	$\{i/10\}_{i=0...5}$	0.3
Nodes (Embedding)	$\{i/10\}_{i=0...5}$	0.4

Table 4.6: Hyperparameters evaluated on RSC dataset

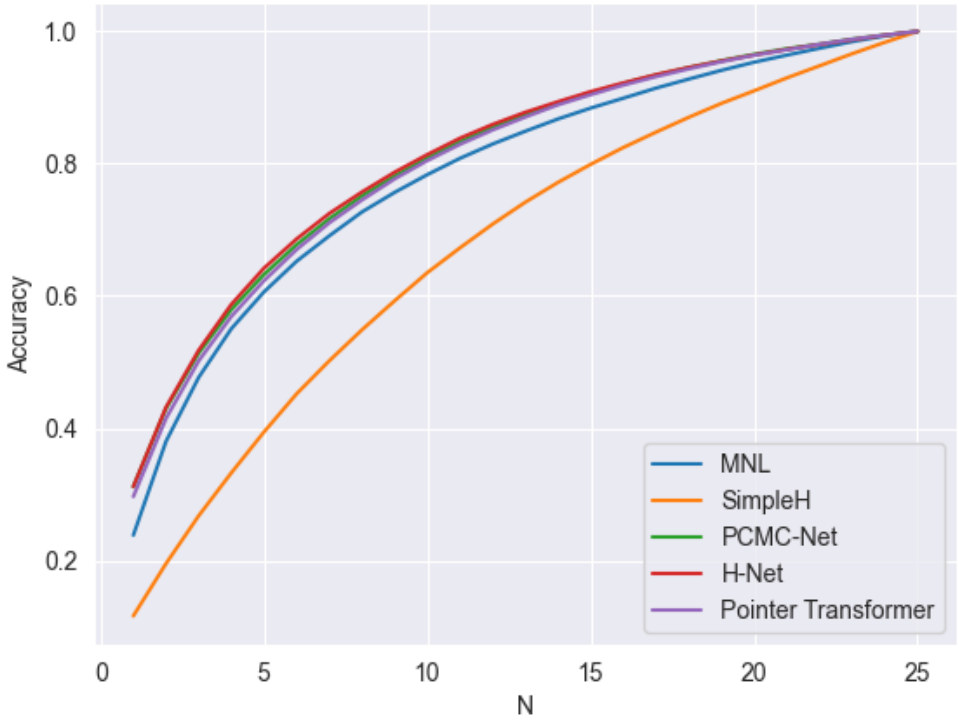


Figure 4.6: RSC Top N Accuracy comparison

Model	**SP**	**AI**	**RSC**
MNL	57	1961	45
SimpleH	58	1962	46
PCMC-Net	795	19790	552046
H-Net	1020	19855	564783
Ptr Transf	5475	42110	581446

Table 4.7: Summary of model sizes across all datasets

Chapter 5

Discussion

5.1 Comments on results

What we gathered from our experiments is that deep learning approaches on simple problems show little advantage over non deep counterparts. For large problems, where contextual information starts to become relevant and anomalies are more likely to occur, deep learning models show instead good accuracy, significantly outperforming non deep counterparts. We expect a sharper separation in problems that are notoriously hard for shallow methods, such as ones involving text or images.

Pairwise comparison models fare better than holistic models that use the full alternative set to compute probabilities, such as Pointer Transformer. This is interesting, as it means that single one-vs-one comparisons can be combined to describe much more complex dynamics in a more accurate way than many-vs-many models, which instead need training to be able to model the same dynamics. It's also possible that Pointer Transformers simply need larger datasets, longer training times or a different architecture altogether to become competitive. For example, one could use a different embedding. Another observation is that it's difficult, in PCMC-Net and H-Net, to disentangle the final probability computation from the neural network. It's unclear whether the results we observe are because the theoretical model behind the architecture can actually describe these processes well, or because the neural network can capture these comparisons by sheer power of approximation. Further testing is necessary to shed light on this aspect.

Some considerations about parameter scaling can be done from the few experiments we run. For instance, Table 4.7 shows how Pointer Transformer starts off using 5 times the amount of parameters of its pairwise counterparts. However, this difference shrinks to 2 times in the AI dataset and is almost absent in the

RSC case, suggesting that Pointer Transformers may have some good scaling properties. It may be worth comparing these models using larger datasets to assess if this holds true.

Looking more closely at simpler methods, MNL still proves to be a superior and easy-to-implement solution compared to its QDT alternative SimpleH, casting doubts on the capabilities of QDT at large.

5.2 The search for a QDT-inspired RNN

A common task recommender systems are faced with is predicting choices for reoccurring users. They may look for specific items, and previous choices could give useful hints about what the following ones would be. Since QDT describes rules for updating a state based on its observed outcomes, it seems a viable candidate for developing a model capable of handling consecutive decisions. We conceive a possible approach that can handle multiple consecutive choices $S_1, S_2, \ldots, S_m$ by constructing and updating a state $|s\rangle$ using several unitary transformations $U_1, U_2, \ldots, U_m$. The final state would be computed as:

$$|s_f\rangle = U_m \ldots U_1 |s\rangle \tag{5.1}$$

The core concept is to stack many comparison layers (as the one described in Section 3.1.4), one for every consecutive choice the user is faced with, in a recurrent fashion. A similar quantum-inspired RNN has never been proposed in literature. However, the approximating power of this architecture is ultimately limited by the linear and unitary nature of quantum operators, likely resulting in inferior performance compared to a classical non-linear RNN. This a recurring issue with a quantum theory of decision used as it is: while relying on the quantum formalism loosens many constraints of classical probability, it bounds the evolution of the state to linear transformations, significantly impacting the expressivity of the resulting model. For these reasons, the idea was discarded.

5.3 Limits of Quantum Decision Theory

Despite its promising results when applied to large problems, QDT is still a discipline at its infancy. Presently, developing new QDT models clashes with some difficulties of practical and theoretical nature. First of all, most applications of QDT are limited to small controlled experiments, and there is a lack of studies describing how QDT-based approaches fare in real-life settings. As a consequence, adapting the theory to new applications takes significant effort. Secondly, not much

is clear about other factors different from prediction accuracy, such as explainability and interpretability. Another setback we encounter during our research is the scarcity of implementations using this theory. We denote, in particular, an absence of deep models leveraging QDT. On a technical note, we empirically observe our implementation based on a matrix exponentiation is noticeably slower to train and run than its very similar classical counterpart PCMC-Net, an aspect one should account when choosing which approach to use. Better implementations of the matrix exponential function could tackle this limit. Finally, some skepticism arises as to whether it's really possible to model human decision making using a formalism, the quantum one, not designed for computational psychology. This thesis simply shows that a particular approach, specifically one based on a combination of QDT with deep learning, is successful at modeling human behavior more than other models. We prefer to leave the QDT hypothesis to physicists and psychologists to prove or disprove.

Chapter 6

Conclusion and Future Work

We propose two novel approaches for decision theory based on deep learning and Quantum Decision Theory (QDT), assessing their performance on similar classical and quantum-based models from literature. The first one takes advantage of the Hamiltonian evolution of quantum systems as a way of using pairwise comparison to compute the final probabilities of alternatives. The second one uses the well-established Attention mechanism to compute a holistic comparison of all alternatives to determine the probability of each option. Our experiments show that pairwise methods are slightly better at modeling decisions than holistic ones. QDT proves to be a powerful tool when applied in conjunction with deep learning methods, while it falls short of modeling decision dynamics when combined with a shallower logic. Multinomial Logit Models (MNL) consistently perform better than quantum-inspired methods of similar complexity. Regarding the Pointer Transformer architecture, further testing is necessary to evaluate its performance. A possible avenue of research for improving this method is using larger datasets and different embeddings. Another promising application for deep choice models is in domains where decisions are based on images or text, where shallow models are known to perform poorly. Finally, we do not find definitive evidence that QDT-inspired approaches can improve performance. Further experiments are necessary to clarify the extent of QDT's explanatory power.

Acknowledgments

This thesis, despite carrying my name, is the work of many people that supported me through the years and made me the person I am now. For their invaluable contribution they deserve to be thanked.

Firstly, I would like to thank my supervisor, Alix Lhéritier, whose attentive eye and skillful direction steered this thesis toward a fruitful and interesting realization. I would also like to thank all my colleagues that contributed by helping me solve problems I, alone, could not have handled, or simply by listening to my crazy conjectures about quantum states and human minds: Rodrigo Acuna Agost, Nicolas Bondoux, Eoin Thomas, Hongliu Cao, Apostolos Avranas, as well as the rest of the ART team. Thanks to you all, I learned what it means to do research.

I would also like to express my gratitude to Professor Marios Kountouris and Professor Bartolomeo Montrucchio for accepting me as their master thesis student and for supporting my project. Their help, suggestions, and expertise throughout this process have been invaluable.

Without the encouragement, inspiration, and humor of many others who accompanied me, this adventure would not have become a reality. I'll do my best to thank them one by one. My lifelong friends: Lorenzo, Federico, Michela and Edoardo, for always being a harbor I could return to in times of need. The friends I grew up with in Torino: Gianluca, Riccardo, Stefano, Daniele and Vito, for always welcoming me back with open arms. My university classmates: Martino, Matteo, Marco (both of you), Gilberto, Paolo, Luca, and Davide, for accompanying me through the highs and lows of an engineering degree. My friends here in France: Alessandro, Demetrio, Dario, Prateek, Simone and Massimiliano, who supported me in a place far from home, making it feel like home.

I want to reserve a special thank you to Yashu: the girl who unexpectedly appeared into my life, made these demanding months a little brighter, and changed so many things in so little time.

And lastly, my deepest thanks and appreciation go to my family. To my grandparents, who taught me kindness and always looked after me. To my mom Nadia and my dad Renato, whose constant love made me the person I am today, who never stopped encouraging my curiosity, who taught me determination, and who always supported me through good and bad times, however far I was. I thank you all, for making this adventure possible.

Appendix A

RSC Features

Feature name	Description	Type	Source	Class
unique_id	Unique identifier for a choice session, obtained by combining session_id and adj_step	ID	Engineered	ID
city	Name of the destination city	Categorical	Original	Context
country	Name of the destination country	Categorical	Original	Context
device	Device that was used for the search	Categorical	Original	Context
platform	Country platform that was used for the search, e.g. trivago.de (DE) or trivago.com (US)	Categorical	Original	Context
step	Step in the sequence of actions within the session	Numerical	Original	Context
timestamp	UNIX timestamp for the time of the interaction	Numerical	Original	Context
adj_step	Adjusted step when session is divided in two	Numerical	Engineered	Context
alt_[item]_click_count_sess	Number of clicks on [item] related to the current alternative in the current session	Numerical	Engineered	Alternative
alt_[item]_click_count_usr	Number of clicks on [item] related to the current alternative by the user	Numerical	Engineered	Alternative
alt_agg_[i]	i-th aggregated feature from the metadata alternative characteristics	Numerical	Engineered	Alternative
alt_is_last_int	Whether the current alternative is the last interacted with	Categorical	Engineered	Alternative
cum_click_ratio	Cumulative proportion of sessions resulting in clickout	Numerical	Engineered	Context
cum_n_cities	Cumulative number of cities searched in the current session	Numerical	Engineered	Context
cum_n_click_item_sess	Cumulative number of items clicked in the current session	Numerical	Engineered	Context
cum_n_click_item_usr	Cumulative number of items clicked by the user	Numerical	Engineered	Context
cum_n_sess	Cumulative number of sessions for the user	Numerical	Engineered	Context
cum_n_sess_click	Cumulative number of sessions resulting in clickout	Numerical	Engineered	Context
cum_price_sess	Cumulative price of clicked alternatives	Numerical	Engineered	Context
cum_sess_[action]	Cumulative number of times [action] was taken in the current session	Numerical	Engineered	Context
cum_usr_[action]	Cumulative number of times [action] was taken by the user	Numerical	Engineered	Context
day	Day of the month, obtained from timestamp	Numerical	Engineered	Context
filter_agg_[i]	i-th aggregated feature from the active filters features	Numerical	Engineered	Context
impression_pos	Position of the current alternative in the list of viewed options	Numerical	Engineered	Alternative
latitude, longitude	Coordinates of city	Numerical	Engineered	Context
price_max, price_min	Maximum and minimum prices in this choice session	Numerical	Engineered	Context
price_rel_max, price_rel_min	Price of the current alternative relative to the maximum and minumum price in this choice session	Numerical	Engineered	Alternative
prices	Price of the current alternative	Numerical	Engineered	Alternative
Stars	Number of stars of the accommodation	Numerical	Engineered	Alternative

Feature name	**Description**	**Type**	**Source**	**Class**
time_sess_curr	Current length of the session in seconds	Numerical	Engineered	Context
time_sess_start	UNIX timestamp for the starting time of the current session	Numerical	Engineered	Context
timediff_last_click	Time since the last clickout action in seconds	Numerical	Engineered	Context

Table A.1: Feature description for the final RSC dataset

Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

[Allais, 1953] Allais, M. (1953). Le comportement de l'homme rationnel devant le risque: critique des postulats et axiomes de l'école américaine. *Econometrica: Journal of the Econometric Society*, pages 503–546.

[authors, 2016] authors, T. G. (2016). Gpyopt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Block and Marschak, 1960] Block, H. D. and Marschak, J. (1960). Random orderings and stochastic theories of response. *Contributions to Probability and Statistics*, 2:97–132.

[Broekaert and Busemeyer, 2018] Broekaert, J. and Busemeyer, J. (2018). Episodic source memory over distribution by quantum-like dynamics – a model exploration.

[Broekaert and Busemeyer, 2017] Broekaert, J. B. and Busemeyer, J. R. (2017). A hamiltonian driven quantum-like model for overdistribution in episodic memory recollection. *Frontiers in Physics*, 5:23.

[Bronkhorst, 2015] Bronkhorst, A. W. (2015). The cocktail-party problem revisited: early processing and selection of multi-talker speech. *Attention, Perception, & Psychophysics*, 77(5):1465–1487.

[Busemeyer and Bruza, 2012a] Busemeyer, J. R. and Bruza, P. D. (2012a). *What is quantum theory? An elementary introduction*, page 1–98. Cambridge University Press.

[Busemeyer and Bruza, 2012b] Busemeyer, J. R. and Bruza, P. D. (2012b). *What is quantum theory? An elementary introduction*, pages 73–74. Cambridge University Press.

[Busemeyer and Bruza, 2012c] Busemeyer, J. R. and Bruza, P. D. (2012c). *What is quantum theory? An elementary introduction*, pages 221–227. Cambridge University Press.

[Busemeyer and Bruza, 2012d] Busemeyer, J. R. and Bruza, P. D. (2012d). *What is quantum theory? An elementary introduction*, pages 267–280. Cambridge University Press.

[Busemeyer et al., 2006] Busemeyer, J. R., Wang, Z., and Townsend, J. T. (2006). Quantum dynamics of human decision-making. *Journal of Mathematical Psychology*, 50(3):220–241. Special Issue: Jean-Claude Falmagne: Part II.

[Busemeyer et al., 2015] Busemeyer, J. R., Wang, Z., Townsend, J. T., and Eidels, A. (2015). *The Oxford handbook of computational and mathematical psychology*. Oxford University Press.

[Chen and Bansal, 2018] Chen, Y. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. *CoRR*, abs/1805.11080.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

[Ellsberg, 1961] Ellsberg, D. (1961). Risk, ambiguity, and the savage axioms. *The quarterly journal of economics*, pages 643–669.

[Gaspar et al., 2019] Gaspar, P., Kompan, M., Koncal, M., and Bielikova, M. (2019). Improving the personalized recommendation in the cold-start scenarios. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 606–607.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Hancock et al., 2020] Hancock, T. O., Broekaert, J., Hess, S., and Choudhury, C. F. (2020). Quantum probability: A new method for modelling travel behaviour. *Transportation Research Part B: Methodological*, 139:165–198.

[Hernández and Amigó, 2021] Hernández, A. and Amigó, J. M. (2021). Attention mechanisms and their applications to complex systems. *Entropy*, 23(3):283.

[Hess and Palma, 2019] Hess, S. and Palma, D. (2019). Apollo: a flexible, powerful and customisable freeware package for choice model estimation and application. *Journal of Choice Modelling*, 32.

[Hogarth and Einhorn, 1992] Hogarth, R. M. and Einhorn, H. J. (1992). Order effects in belief updating: The belief-adjustment model. *Cognitive psychology*, 24(1):1–55.

[Huber et al., 1982] Huber, J., Payne, J. W., and Puto, C. (1982). Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of consumer research*, 9(1):90–98.

[Jannach et al., 2010] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press.

[Jedlicka, 2017] Jedlicka, P. (2017). Revisiting the quantum brain hypothesis: Toward quantum (neuro)biology? *Frontiers in Molecular Neuroscience*, 10.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Lhéritier, 2019] Lhéritier, A. (2019). Pcmc-net: Feature-based pairwise choice markov chains. In *International Conference on Learning Representations*.

[Lhéritier et al., 2019] Lhéritier, A., Bocamazo, M., Delahaye, T., and Acuna-Agost, R. (2019). Airline itinerary choice modeling using machine learning. *Journal of Choice Modelling*, 31:198–209.

[Li et al., 2018] Li, J., Wang, Y., Lyu, M. R., and King, I. (2018). Code completion with neural attention and pointer networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4159–25. AAAI Press.

[Lin et al., 2022] Lin, T., Wang, Y., Liu, X., and Qiu, X. (2022). A survey of transformers. *AI Open*, 3:111–132.

[Lindsay, 2020] Lindsay, G. W. (2020). Attention in psychology, neuroscience, and machine learning. *Frontiers in computational neuroscience*, 14:29.

[Luce, 1977] Luce, R. D. (1977). The choice axiom after twenty years. *Journal of mathematical psychology*, 15(3):215–233.

[Manski, 1977] Manski, C. F. (1977). The structure of random utility models. *Theory and decision*, 8(3):229–254.

[Mottini and Acuna-Agost, 2017] Mottini, A. and Acuna-Agost, R. (2017). Deep choice model using pointer networks for airline itinerary prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1575–1583. ACM.

[Neumann and Morgenstern, 1944] Neumann, J. V. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton, NJ, USA: Princeton University Press.

[Norris, 1997] Norris, J. R. (1997). *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S.,

Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703.

[Penrose, 1990] Penrose, R. (1990). *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Viking Penguin.

[Peterson et al., 2021] Peterson, J. C., Bourgin, D. D., Agrawal, M., Reichman, D., and Griffiths, T. L. (2021). Using large-scale experiments and machine learning to discover theories of human decision-making. *Science*, 372(6547):1209–1214.

[Phan et al., 2022] Phan, D., Vu, H., and Currie, G. (2022). Attentionchoice: Discrete choice modelling supported by a deep learning attention mechanism. *SSRN Electronic Journal*.

[Ragain and Ugander, 2016] Ragain, S. and Ugander, J. (2016). Pairwise choice markov chains. In *Advances in Neural Information Processing Systems*, pages 3198–3206.

[Rossi and Paradiso, 1995] Rossi, A. F. and Paradiso, M. A. (1995). Feature-specific effects of selective visual attention. *Vision Research*, 35(5):621–634.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536.

[Shanteau, 1970] Shanteau, J. C. (1970). An additive model for sequential decision making. *Journal of Experimental Psychology*, 85(2):181.

[Simonson, 1989] Simonson, I. (1989). Choice based on reasons: The case of attraction and compromise effects. *Journal of consumer research*, 16(2):158–174.

[Tversky, 1972] Tversky, A. (1972). Elimination by aspects: A theory of choice. *Psychological review*, 79(4):281.

[Tversky and Kahneman, 1992] Tversky, A. and Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5(4):297–323.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Vinyals et al., 2015] Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, 28.

[Virtanen et al., 2020] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0:

Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

[von Neumann, 2018] von Neumann, J. (2018). *Mathematical Foundations of Quantum Mechanics*. Princeton University Press, Princeton.

[Yearsley and Busemeyer, 2016a] Yearsley, J. M. and Busemeyer, J. R. (2016a). Quantum cognition and decision theories: A tutorial. *Journal of Mathematical Psychology*, 74:99–116. Foundations of Probability Theory in Psychology and Beyond.

[Yearsley and Busemeyer, 2016b] Yearsley, J. M. and Busemeyer, J. R. (2016b). Quantum cognition and decision theories: A tutorial. *Journal of Mathematical Psychology*, 74:99–116. Foundations of Probability Theory in Psychology and Beyond.

[Yukalov and Sornette, 2015] Yukalov, V. I. and Sornette, D. (2015). Preference reversal in quantum decision theory.

[Zhang et al., 2017] Zhang, S., Yao, L., and Sun, A. (2017). Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435.

[Zhao et al., 2018] Zhao, X., Xia, L., Zhang, L., Ding, Z., Yin, D., and Tang, J. (2018). Deep reinforcement learning for page-wise recommendations. *CoRR*, abs/1805.02343.