Università degli Studi di Trieste

Dipartimento di Matematica, Informatica e Geosicenze

Corso di studi in Intelligenza Artificiale e Data Analytics

Tesi di Laurea Triennale

Contextual Preference Ranking for Card Recommendations in Magic: The Gathering

Laureando:
Lorenzo Bortolussi

Relatrice:
Tatjana Petrov

Anno Accademico 2024/2025

Abstract

Games have and continue to serve as a critical domain for advancing artificial intelligence. Among them, the card game *Magic: The Gathering* offers two distinct but linked challenges: the game itself and the deck building process which precedes it. The quality of a deck depends on individual cards and their mutual compatibility. For many game formats of *Magic: The Gathering*, the players typically consult web-sites for building the decks they play with.

In this thesis, we design and implement a tool to support deck building in the Commander (EDH) format of *Magic: The Gathering*. Relevant data is collected from online sources, and a model based on the Contextual Preference Learning framework is trained to capture card compatibility within decks. This model is used at the core of a recommendation system which also supports text-guided queries, providing users with relevant cards to complete their decks.

The recommendation system is evaluated by human experts. Results show that it is capable of providing relevant recommendations on most queries made by users. Possible improvements are also discussed.

Contents

Chapter 1

Introduction

1.1 AI applications in Magic: The Gathering

"Games historically served as a critical domain for advancing artificial intelligence (AI) research. While AI agents have outperformed humans in many classical games, contemporary board and video games continue to present significant challenges." This observation by Bertram et al. [1] summarizes well the current landscape of game-related AI research, where games like Chess and Go have given way to more complex challenges.

Within this modern panorama, the collectible card game *Magic: The Gathering* (*MTG*) emerges as a particularly compelling subject of study. It is characterized by an high degree of complexity, not only because of its extensive rules but also thanks to its dynamic and imperfect-information gameplay. This challenging nature has attracted attention from the research community, leading to explorations of the game from multiple computational and strategic perspectives.

From a formal computational standpoint, the game's complexity has been extensively studied. Churchill et al. [2] demonstrated that determining the optimal play in a generalized game of *Magic* is at least as hard as the Halting Problem, thereby establishing *MTG* as the first tabletop Turing-complete game. Even verifying the legality of a single game action has been shown to be a coNP-complete problem [3]. More recently, the game's expressive power was strikingly illustrated by Howe and Churchill [4], who constructed a functional microcontroller programming language entirely within the rules of a tournament-legal deck of cards.

Research into *MTG* as a playable game is often divided into two distinct domains: the deck construction phase, which precedes gameplay, and the gameplay phase itself. This separation is a practical necessity as the computational intractability of the game makes a fast, reliable simulator unfeasible. Without such a simulator, it is exceptionally difficult to generate the vast quantities of game outcome data required to inform deck construction through reinforcement learning or other policy-based methods.

Consequently, research into gameplay has focused primarily on developing automated

agents, or bots. The advent of digital platforms like *Magic: The Gathering Arena*[1] has sparked interest in this area, with several studies exploring heuristic-based and model-based botting solutions [5]. However, the immense state space and imperfect information inherent to the game continue to be significant barriers, and current automated agents have yet to achieve a level of strategic play comparable to that of an expert human.

The second domain, deck construction, has proven to be a more tractable area for machine learning applications. The existence of large, publicly available datasets documenting human choices — particularly from the "draft" format — has enabled researchers to model the decision-making process of card selection. These studies have employed various methods with the goal of replicating human-like intuition in drafting [6, 7, 8, 1].

Within this subfield, a novel approach has been pioneered by Bertram et al. [1, 8], whose work goes beyond simple card-picking heuristics and introduces a Contextual Preference Ranking (CPR) framework. By leveraging contrastive learning techniques, their models learn to evaluate which candidate cards are most synergistic with a given, incomplete deck. This focus on contextual synergy and preference learning is the inspiration for the recommendation engine developed in this thesis, which shifts the focus from draft to Commander (EDH).

1.2 Problem of Interest

In the EDH format, players build a deck of 100 cards choosing from a large pool of cards considering two main factors: the play style they want to emphasize and the functional roles that individual cards should fulfill.

As stated in section 2.1, the available card pool exceeds 31 000 unique cards, which means that players typically do not build decks doing a comprehensive search through all possible additions. Instead, they often rely on decks created and published by other players, with aggregated data presented on sites such as *EDHREC* [9].

While such aggregated data can serve as a useful starting point, it has been observed that this tends to reinforce common strategies while obscuring more nuanced approaches to deck building. Consequently, identifying viable alternatives to commonly used or expensive cards demands considerable time and effort.

To address this challenge, in this work a tool is developed to assist users by suggesting cards that complement a partially constructed deck, while also enabling targeted searches for specific roles through a natural-language prompt.

The central **research questions** of this thesis are the following:

1. Can the Contextual Preference Ranking framework introduced by Timo Bertram et al. [8, 1] be generalized to recognize the much more complex relations and dynamics among cards in the EDH setting?

2. Can this learned knowledge be used to build a functional recommendation system for *MTG* cards?

[1] `https://magic.wizards.com/en/mtgarena`

1.3 Thesis Structure

This thesis presents the design, implementation, and evaluation of a novel card recommendation system for the Commander format of *Magic: The Gathering*. The system leverages a Contextual Preference Ranking (CPR) framework to model synergistic relationships between cards, moving beyond traditional collaborative filtering or content-based approaches. A complete software infrastructure, from data collection to an interactive demo, has been developed to support this research. The full source code is publicly available on GitHub [10].

Chapter 2 provides the theoretical foundations for this work. It begins with an overview of *Magic: The Gathering*, focusing on the specific complexities of the Commander format. It then introduces the core machine learning paradigms employed: Natural Language Processing for transforming unstructured card text into meaningful feature vectors; Contrastive Learning as the mechanism underlying the CPR framework; and principles of Information Retrieval that are used in the final vector search and recommendation engine.

Section 3.1 details the data collection and preprocessing pipeline. This chapter describes the methods used to gather raw data from public sources and the filtering criteria applied to construct a high-quality dataset. A significant focus is placed on the multi-modal feature engineering process, detailing how each card is transformed into a rich, composite vector representation that serves as the foundational data structure for all subsequent models.

Section 3.2 addresses the critical challenge of creating meaningful text embeddings from the specialized language of *MTG* cards with a three-stage training methodology. For this task, a domain-adapted foundation model is created, along with an expert card role classifier. These two models provide key features for the final card representation.

Section 3.3 describes the training and configuration of the core CPR model. This chapter details the specific architecture of the card and deck encoders, which are later trained with multiple training configurations, varying the training dataset, loss function (Triplet Margin vs. InfoNCE), and training duration.

Section 3.4 details the implementation of the final recommendation engine. It describes the construction of a persistent vector database for efficient similarity search, then elaborates on the design of the retrieval system, including the implementation of a re-ranking strategy to balance synergistic relevance with prompt-guided intent.

Section 3.5 describes the implementation of an interactive online demo that allows to gather feedback on the performance of different model configurations, provided by human experts on out-of-sample decks.

Chapter 4 presents the qualitative evaluation of the system. Collected data are analyzed and results provide a comparative assessment of the different trained models, offering insights into the strengths and weaknesses of the CPR framework for this task.

Finally, **chapter 5** discusses the limitations of the current system and outlines promising directions for future research, like the potential for leveraging the collected user feedback to further fine-tune the models.

Chapter 2

Background

In this chapter, an overview of the background needed to understand this thesis is given. The game domain of *Magic: The Gathering* is summarized, then other, more technical fields of interest for this thesis are also briefly introduced, namely Natural Language Processing, Information Retrieval and Contrastive Learning.

2.1 Magic: The Gathering

Magic: The Gathering, commonly abbreviated as *Magic* or *MTG*, is a strategic collectible card game created by Richard Garfield in 1993 and published by Wizards of the Coast[1]. The game is played both as a physical tabletop game and in a digital format on the *MTG Arena* platform. More than a single game, *Magic* is a system defined by a shared, comprehensive ruleset and an ever-expanding pool of unique components (cards). This system supports numerous formats, each providing distinct rules for deck construction and gameplay. The number of unique cards at the time of writing exceeds 31 000[2], creating a vast and continually evolving strategic landscape.

2.1.1 Rules and Game Complexity

From a game-theoretic perspective, *Magic: The Gathering* is an asymmetric, stochastic, zero-sum game of imperfect information. Games are typically played between two or more players, each with their own customized deck of cards. The game state is partially observable; elements such as the battlefield (cards in play) and the graveyard (discarded or destroyed cards) are public information. However, crucial components remain hidden, including the cards in each player's hand and the composition and order of their library (the deck from which they draw cards). This asymmetry of information is a core source of the game's strategic complexity.

Gameplay is sequential and turn-based. A single turn is composed of multiple distinct

[1] `https://company.wizards.com/`

[2] `https://scryfall.com/search?q=-is%3Areprint`

phases, within which each player can take numerous actions, which are influenced by a great number of factors, like:

- Resources available to each player in the form of cards in hand, mana sources and effects that can be activated or triggered on board;
- Which cards each player have on their board or in graveyards and how they can be interacted with;
- What consequences can arise following a decision based on the state of the game, under assumptions regarding the unknown variables (e.g.: which cards opponents have in their hands), and wether a suboptimal decision can lead to higher chances of winning the game in the long run.

While most actions are restricted to a player's own turn, a significant subset of cards can be played during an opponent's turn. This "action-stacking" capability, combined with the stochastic nature of the card draw and the multitude of subjective decisions made within a single turn, results in an exceptionally large branching factor, making the game computationally intractable for exhaustive search algorithms.

Furthermore, each *Magic* card possesses a unique rules text that requires a semantic understanding of the game's specialized lexicon. The interactions between cards are often non-obvious and dynamic, meaning the function of a card can change dramatically based on the context of other cards in play. The continuous introduction of new cards and mechanics further emphasizes this complexity, constantly expanding the set of possible synergies and strategic interactions.

2.2 The Commander Format

While *Magic* offers many ways to play, this thesis focuses on its most popular multiplayer format: Commander, also known as Elder Dragon Highlander (EDH). Commander distinguishes itself with a unique set of rules that allow for a different style of play compared to traditional one-on-one formats:

- **Player Count:** A typical game involves four players in a free-for-all setting.
- **Deck Size and Singleton Rule:** Decks are composed of exactly 100 cards, and with the exception of basic lands, no two cards in a deck may share the same name. This greatly increases the variance of games.
- **The Commander:** Each deck is built around a specific legendary creature designated as its "Commander". The Commander begins the game in a special zone (the Command Zone) and can be cast from it repeatedly throughout the game, albeit at an increasing mana cost.
- **Color Identity:** The cards in a deck may only contain mana symbols that also appear on its designated Commander. This rule, known as "color identity", provides a strong thematic and mechanical constraint on deck construction.

These rules collectively encourage the creation of decks with strong internal synergies and long-term game plans, making Commander an ideal domain for studying the concept of synergistic deck construction.

2.2.1 Magic Cards

Figure 2.2.1: The structure of a *Magic: The Gathering* card, highlighting its multi-modal data components. Attributes include natural language (name, Oracle text), integer values (mana value, power/toughness), categorical features (type line), and card artwork.

The primary challenge in representing an *MTG* card for machine learning is the effective vectorization of its heterogeneous, semi-structured data. As illustrated in Figure 2.2.1, a single card contains multiple distinct data modalities that must be encoded into a unified representation. The goal is to produce a dense numerical vector, or embedding, that captures the card's functional and semantic characteristics.

The representation pipeline used in this work is detailed in section 3.1, but it builds upon the approach proposed by Bertram et al. [1]. They found that a concatenation of hand-coded features, image embeddings, and metadata yielded a representation that generalized well to unseen cards. This thesis adapts that methodology for the Commander format, accounting for the differences in information available between the two different formats. For instance, the primary informational signal that a card art provides to a model is often highly correlated with the card's color, a feature that this work handles explicitly through metadata filtering.

2.2.2 Deck Construction and Evaluation

Choosing which 99 cards to include alongside a Commander, a process known as deck building, is a complex optimization task. The power of a deck is not merely the sum of the individual strengths of its cards but is an emergent property derived from the synergistic interactions between them. While strategic archetypes (e.g., aggressive "aggro", defensive "control", or combination-focused "combo") exist and can have favorable or unfavorable matchups, the rigid "meta-game" dynamic often seen in competitive formats is significantly mitigated in Commander. The four players, singleton nature of the format introduces immense variance, and along with social dynamics - known as "politics" - they shift the deck building focus from narrow counter-strategies towards decks with resilient and powerful internal engines.

This work leverages a large dataset of human-constructed decklists to train the CPR model. This approach is based on the assumption that these publicly available decks are generally "well-built". This concept is difficult to formalize, as deck performance is highly contextual, varying between different playgroups and depending on player skill. Assigning a static power level to any given deck is an arduous task, which Wizards of the Coast tried to tackle raising critiques from the community[3]. While some research has begun to formalize the concept of synergy in games [11], this work operates on the reasonable assumption that the collective wisdom of thousands of players provides a strong and valid signal for identifying synergistic card relationships.

2.3 Natural Language Processing

Natural Language Processing (NLP) is a subfield of artificial intelligence focused on enabling computers to understand, interpret, and generate human language. In the context of this thesis, NLP is the critical bridge between the unstructured text on a *Magic* card and a structured, numerical representation that a machine learning model can process.

2.3.1 Text Data Preprocessing

Raw text from various sources — such as card Oracle text, game rules, and scraped articles — is inherently noisy and inconsistent. Before being presented to a language model, this text must undergo a rigorous preprocessing pipeline to normalize it into a canonical format. This involves several steps: removing irrelevant artifacts (e.g., HTML tags, emojis), standardizing punctuation and whitespace, and converting domain-specific symbols into natural language. A crucial step is tokenization, where the cleaned text is broken down into a sequence of smaller units, or tokens, which correspond to entries in a model's vocabulary. This sequence of tokens forms the fundamental input for the language models used in this work.

[3] `https://magic.wizards.com/en/news/announcements/introducing-commander-brackets-beta`

2.3.2 Natural Language Inference

Natural Language Inference (NLI) [12] is a task in NLP where a model needs to reason about the relationship between two sentences. Given a "premise" sentence and a "hypothesis" sentence, the model must determine if the relationship is one of:

- **Entailment:** The premise logically implies the hypothesis is true.
- **Contradiction:** The premise logically implies the hypothesis is false.
- **Neutral:** The two sentences are not logically related.

Models pre-trained on large-scale NLI datasets have demonstrated a powerful capability for zero-shot text classification. By framing a classification label as a hypothesis (e.g., "This card's function is Card Draw"), the model can determine the entailment probability between the premise (a card's text) and each potential hypothesis (the labels) without having been explicitly trained on that specific card or label. This technique is used in section 3.2 to generate a pseudo-labeled dataset used to train the expert card role classifier.

2.3.3 Knowledge Distillation

Knowledge Distillation is a model compression and transfer learning technique where knowledge from a large, powerful "teacher" model is transferred to a smaller, more efficient "student" model [13]. This is particularly useful when the student needs to be both accurate and computationally inexpensive for its target task.

The process works by training the student model on a dual objective. First, the student is trained to match the soft probability outputs of the teacher model for a given input. This is typically achieved by minimizing the Kullback-Leibler (KL) divergence between the student's and teacher's probability distributions, often softened using a "temperature" scaling factor. This soft loss encourages the student to learn the teacher's "reasoning process" and the relationships it has recognized between classes.

The student is often simultaneously trained on a standard hard loss (like cross-entropy) against the ground-truth labels. The final training objective is a weighted combination of these soft and hard losses. This methodology is leveraged in section 3.2 to train an expert, domain-adapted card role classifier (the student) by distilling knowledge from a large, general-purpose NLI model (the teacher).

2.4 Information Retrieval

Information Retrieval (IR) can be defined as finding material of an unstructured nature that satisfies an information need from within large collections. An Information Retrieval System must interpret the information need of the user and estimate the relevance of documents with relation to it. Modern IR has moved beyond traditional keyword-based methods to semantic, vector-based approaches.

2.4.1 Data Acquisition: APIs and Web Scraping

The primary methods to automatize data acquisition from web sources are Application Programming Interfaces (APIs) and direct web scraping. An API provides a structured, predictable, and officially supported method for a program to request and receive data. This is the preferred method, as it is stable and sanctioned by the data provider.

However, in many real-world scenarios, a public API may be undocumented or may not exist at all. In the case of undocumented API, it is possible to reverse-engineer the API's endpoints and parameters by observing the network traffic generated by the website in a browser.

When no API is available, web scraping becomes the necessary alternative. This technique involves the automated extraction of data directly from the HTML or XML content of web pages. It typically works by making an HTTP request to a target URL to fetch the raw page content, and then parsing this content to extract the desired information. For this thesis, the Python library `requests` [14] was used for HTTP requests, and `BeautifulSoup` [15] for parsing. To ensure this process was both efficient and respectful of server resources, this work prioritized parsing publicly available sitemaps. These XML files offer a structured list of all crawlable URLs, allowing for comprehensive data collection without fragile page-by-page crawling. This methodology is employed in section 3.1 to construct the natural language text corpus.

2.4.2 Vector Databases

A vector database is a specialized system designed for the efficient storage and retrieval of high-dimensional vector embeddings, especially useful for storing and retrieval of vectors for AI applications. It utilizes algorithms like Hierarchical Navigable Small Worlds (HNSW) [16] to perform Approximate Nearest Neighbor searches, based on the geometric properties of stored vectors. This allows it to find the vectors in a collection that are closest (most similar) to a given query vector with extremely low latency, even across collections containing millions of documents. This technology is the backbone of the recommendation system, enabling real-time similarity searches on the generated card embeddings.

2.4.3 Maximal Marginal Relevance

A common failure mode in simple similarity search is result redundancy, as a query might return a ranked list containing identical results, which provides little value to the user. The algorithm of Maximal Marginal Relevance (MMR) [17] addresses this problem, reranking results to optimize for both relevance and diversity in the final set of results.

The principle behind MMR is to iteratively build a result set by selecting items that represent the best trade-off between two competing objectives: being highly relevant to the original query and being novel with respect to the items already selected. The algorithm is a greedy process:

1. A large pool of candidate documents is retrieved using an initial similarity search.

2. The most relevant document to the query is selected and added to the (initially empty) result set.

3. The algorithm evaluates all remaining candidates and selects the one that maximizes the MMR score. This score is a linear combination of the candidate's relevance to the query and its dissimilarity to the already-selected results.

4. Point 3 is repeated until the desired number of recommendations is reached.

Formally, at each step, the MMR algorithm selects the next document from the pool of unselected candidates ($R \setminus S$) that maximizes the following expression:

$$\underset{D_i \in R \setminus S}{\operatorname{argmax}} \left[\lambda \cdot \operatorname{sim}(D_i, Q) - (1 - \lambda) \cdot \max_{D_j \in S} \operatorname{sim}(D_i, D_j) \right] \tag{2.1}$$

where:

- Q is the original query vector.
- R is the set of all candidate documents retrieved in the initial search.
- S is the subset of documents already selected for the final result list.
- D_i is a candidate document being scored, and D_j is a document already in S.
- $\operatorname{sim}(u, v)$ is a similarity function, such as cosine similarity, between two vectors.
- λ is a hyperparameter in the range $[0, 1]$ that controls the trade-off between relevance and diversity.

The hyperparameter λ allows for fine-tuning the output. A value of $\lambda = 1$ reduces the algorithm to a standard relevance-based ranking, ignoring diversity. Conversely, a value of $\lambda = 0$ would produce a maximally diverse set of results without regard for the original query.

2.5 Learning From Comparisons: Contrastive Learning

In the field of Metric Learning, Contrastive Learning is a self-supervised machine learning technique used to learn a structured embedding space without explicit labels. Instead of learning to predict a label for a single data point, a contrastive model learns to differentiate between similar and dissimilar pairs of data points. By optimizing a loss function that pulls positive pairs closer together and pushes negative pairs farther apart, the model learns a representation where where semantic similarity is reflected by spatial proximity in the embedding space.

2.5.1 Siamese Networks

A Siamese network [18] is a neural network architecture well-suited for tasks such as similarity learning and preference ranking, where the goal is to compare two or more inputs rather than classify a single one. The architecture consists of two or more identical networks that share their entire set of weights. During a forward pass, each network processes one of the inputs to be compared.

The core principle of a Siamese network is the **weight sharing** paradigm. Since networks are identical, they learn a single, consistent mapping function, $f(\cdot)$, which projects any given input into a common, low-dimensional embedding space. The objective of the training process is to structure this space so that the distance between the vector representations of two inputs becomes a meaningful measure of their semantic similarity. This ensures that the learned similarity metric is symmetric, so the computed distance between item A and item B is identical to the distance between B and A.

Siamese networks have become a common tool in identification tasks due to their one-shot capabilities [19] that allow them to learn with limited examples per class, allowing for meaningful generalization to unseen items. Formulating the training task with comparisons allows models to learn meaningful embeddings, which can be used to classify inputs. Training Siamese networks requires loss functions specific for the task, and the ones used in this work are described in the following section.

2.5.2 Triplet Margin Loss and InfoNCE Loss

The objective of a contrastive model is defined by its loss function, which guides the optimization process by quantifying the "quality" of the learned embedding space. This work explores two prominent contrastive loss functions: the Triplet Margin Loss and the InfoNCE Loss.

Triplet Margin Loss

The Triplet Margin Loss is a classic metric learning objective that operates on a triplet of data points: an **anchor** (A), a **positive** sample (P) that is similar to the anchor, and a **negative** sample (N) that is dissimilar. In the context of this thesis, the anchor is a partial deck embedding, the positive is a synergistic card embedding, and the negative is a non-synergistic card embedding.

The goal of the loss function is to ensure that the distance between the anchor and the positive sample in the embedding space is smaller than the distance between the anchor and the negative sample by at least a certain margin, α. This can be visualized as moving positive pairs closer together while pushing negative pairs farther apart. Formally, the Triplet Margin Loss is defined as:

$$\mathcal{L}(A, P, N) = \max\left(d(f(A), f(P)) - d(f(A), f(N)) + \alpha, 0\right) \tag{2.2}$$

where:

- $f(\cdot)$ is the embedding function (the neural network) that maps an input to a vector in the embedding space.
- $d(x, y)$ is a distance function, typically the squared Euclidean distance $\|x - y\|_2^2$.
- α is a hyperparameter representing the margin.

The $\max(\cdot, 0)$ component ensures that no loss is incurred if the negative sample is already farther from the anchor than the positive sample by at least the margin α. A key challenge with this approach is the selection of informative hard negatives [20, 21]. If negative samples are chosen randomly, the condition in Equation 2.2 is often satisfied early in training, causing the loss to become zero and learning to stagnate. To mitigate this, this work employs **in-batch negative mining**, where for a given anchor-positive pair, other positive samples within the same mini-batch are considered as potential hard negatives, ensuring the model is continually presented with challenging examples.

InfoNCE Loss

The InfoNCE (Noise-Contrastive Estimation) loss [22] offers a data-efficient alternative by reframing the contrastive task as a probabilistic classification problem. For a given anchor d, the model's task is to correctly identify its corresponding positive sample or key k^+ from a set of $N - 1$ negative samples $\{k_i^-\}$.

In this framework, all other positive samples within a mini-batch are utilized as negative samples for a given anchor-positive pair, while alternative methods are proposed in chapter 5. The similarity between the query and each key is computed, typically using the dot product. These similarity scores are then scaled by a temperature hyperparameter, τ, and transformed into a probability distribution using the softmax function. The InfoNCE loss is computed as the cross-entropy loss for this multi-class classification problem. Formally, it is defined as:

$$\mathcal{L}_{NCE} = -\log \frac{\exp(\text{sim}(f(d), f(k^+))/\tau)}{\exp(\text{sim}(f(d), f(k^+))/\tau) + \sum_{i=1}^{N-1} \exp(\text{sim}(f(d), f(k_i^-))/\tau)} \tag{2.3}$$

where:

- $\text{sim}(u, v)$ is the cosine similarity function, computed as $(u \cdot v)/(\|u\|\|v\|)$
- τ is the temperature hyperparameter, which controls the sharpness of the probability distribution. A lower temperature makes the distribution sharper, forcing the model to pay more attention to hard negatives, while a higher temperature softens the distribution.

By contrasting one positive pair against many negative samples simultaneously, InfoNCE leverages the information within a batch more effectively than Triplet Loss, often leading to faster convergence and a more structured embedding space. Both of these approaches are implemented and evaluated as distinct training configurations, and the results can be found in chapter 4.

2.5.3 The Contextual Preference Ranking Framework

A core challenge in many recommendation and sequential decision-making tasks can be formalized as the **Set Addition Problem**. Given a context set of items, D, the goal is to determine which candidate item, c, is the best single addition to that set. The quality of an addition is not based on the intrinsic value of the item c alone, but on the quality of the resulting set, $D \cup \{c\}$. This is particularly relevant in domains like *Magic: The Gathering*, where the value of a card is defined by its synergistic interactions with the other cards in a deck.

To address this challenge, this work adopts the **Contextual Preference Ranking** (CPR) framework, as introduced and formalized by Bertram et al. [8, 1]. This represents a significant conceptual shift from traditional recommender systems, as it explicitly incorporates the surrounding context to model conditional preferences. Instead of directly predicting a score for the quality of the set $D \cup \{c\}$, which can be a difficult and poorly-defined objective, CPR reframes the task as learning a pairwise preference relation.

Given a context D (an incomplete deck), a "positive" item c^+ (a synergistic card), and a "negative" item c^- (a non-synergistic card), the framework learns to model the preference for c^+ over c^- as a completion for the context. This preference is formally expressed as:

$$c^+ \succ_D c^- \tag{2.4}$$

where $\succ_D$ denotes a preference conditioned on the given context D. This formulation moves beyond simple similarity: the question is not merely whether the embedding of D is closer to $f(c^+)$ than $f(c^-)$, but which addition yields a superior resulting set.

To learn this preference relation, CPR employs an embedding-based approach. A Siamese network learns embedding functions $f(\cdot)$ that project both the context D and the candidate items into a shared vector space, where the preference is modeled as a function of the relative similarity scores between the context embedding and the embeddings of the candidates. Bertram et al. propose optimizing this relationship using a contrastive loss, which enforces a higher similarity score between the context and the positive item than between the context and the negative item.

Chapter 3

Methodology and System Implementation

This chapter presents the complete technical implementation of the proposed recommender system, from data acquisition to final deployment. The discussion begins with the foundational methods for data retrieval, filtering, and feature engineering used to construct the card and deck datasets. Subsequently, it delves into the architectural design and end-to-end training of the core Contextual Preference Ranking model. The chapter concludes by describing the integration of the trained models into a vector database and their deployment within an interactive online demo designed for user evaluation. The full source code is publicly available on GitHub [10].

3.1 Data Retrieval, Filtering, and Representation

This section presents how data is processed after being retrieved from the main sources used throughout this work:

- Data concerning single cards is retrieved from *Scryfall* [23] and is used to create a mapping from cards (identified with their oracle id) to their gameplay information. This mapping is necessary to create card representations and to later add metadata to the vector database, described in section 3.4.
- Decklists are retrieved from *Archidekt* [24] to build a dataset used to train the main CPR model, described in section 3.3.
- The text corpus used for fine tuning in section 3.2 is assembled using *MTG* official rules[1], Oracle texts and game related articles retrieved from *EDHREC* [9].

The primary challenge in representing an *MTG* card - the building block for training datasets - lies in the effective vectorization of its heterogeneous, semi-structured data. The goal is to produce a dense numerical vector, or embedding, that captures the card's

[1] `https://magic.wizards.com/en/rules`

functional and semantic characteristics in a way that is meaningful for downstream machine learning tasks.

The **card representation** pipeline used in this work is adapted from the methodology of Bertram et al. [1], who demonstrated that concatenating hand-crafted features, image embeddings, and metadata produced representations that generalized effectively to unseen cards. While this approach was well-suited to the context they examined, it requires adaptation for the EDH setting. In particular, obtaining consistent and meaningful metadata for the majority of legal cards is impractical, and the relevance of image features in their study was likely due to their strong correlation with card color, a factor of central importance in draft format, which is handled separately in this work.

3.1.1 Card Data

Scryfall [23] is the most popular card database and search engine for *MTG* cards, allowing for fast card searches based on a wide range of filters and keywords. It is used by many *MTG* related services thanks to its free and documented API[2].

In this work, *Scryfall*'s API[3] is used to download card bulk data, which contains information about all cards present in their database. This downloaded dataset has been filtered to remove non legal cards in EDH format and to only preserve relevant information, while also preprocessing many fields, especially those containing text. The Specific filters applied and fields kept can be found in `preprocess_cards.py`[4], specifically in the function `filter_data`.

After this process, a dictionary of unique cards associated with key information is created, which becomes the base for constructing meaningful card representations.

3.1.2 Decklists and Datasets

As opposed to the draft setting, there is no dataset containing ready for use EDH decklists. For this reason, this dataset was built from scratch using the API provided by *Archidekt* [24].

Archidekt [24] is a visual *MTG* deck builder which provides a public, although undocumented API[5] to access all public decks on its platform. After reverse-engeneering its rules, it becomes an incredibly useful tool to retrieve the big ammount of decklists necessary to render viable a deep learning approach to learn how decks are built.

The decklists retrieved this way have their cards filtered, removing the ones that are not present in the previously created dictionary of relevant cards. They are later used to create two datasets for later use: one containing all decks retrieved (around 90 000 decks), and another one in which decks are balanced using color identity and tags, thus

[2] `https://scryfall.com/docs/api`

[3] `https://scryfall.com/docs/api`

[4] `https://github.com/lollobrollo/Card-Recommender-System/blob/main/preprocess_cards.py`

[5] `https://archidekt.com/forum/thread/40353/1`

reducing the instances of the most prevalent decks in the collection, resulting in a dataset with around a third of the original instances. Details on this process can be found in `edh_scraper.py`[6]

3.1.3 Card and Deck Representations

Card representations are constructed extracting features from the Scryfall [23] dataset, including numerical and categorical data (e.g.: mana cost, color identity) and, most critically, the card's Oracle text.

The oracle text vector representation is generated using a specialized language model which is described in more detail in section 3.2. In that same section is described another language model trained to provide a vector representation that encodes the "role" a card fits in, based on its Oracle text. The vector produced this way is also added to the card representation.

Numerical variables are represented with their value, while most of the other variables are treated as categorical and encoded using one-hot or multi-hot encodings. Details on this process can be found in the function `build_card_representations`, from the script `preprocess_cards.py`[7].

The multi-hot encodings of card type and keywords resulted in large and sparse vectors, thus they are added to the representations of cards after being compressed by a specialized network, trained end-to-end with the main CPR pipeline, as described in section 3.3.

Following the methodology of Bertram et al. [1], a deck is represented as a sequence of its constituent card vectors, which are constructed as defined above.

3.1.4 Composite Corpus for LLM fine-tuning

In order to train a Large Language Model specialized in *Magic*, as described in section 3.2, a large corpus of texts needs to be retrieved. This is done by extracting rules and definitions from the official rule book of the game[8], to provide the model with knowledge about game mechanics and game-specific definitions. These paragraphs are then expanded with Oracle texts, to give the model an understanding of how an Oracle text can be structured. The corpus aggregated this way resulted insufficient in creating meaningful embeddings, so it was further expanded with *MTG* related articles scraped from *EDHREC* [9].

EDHREC [9] is a deck analysis and recommendation tool for the Commander format of *MTG*. Incidentally, it also hosts numerous articles about the game. These articles expose the language model to synonyms, metaphors, and slang that players might use when querying the system. Articles are retrieved using *EDHREC* [9] sitemap index[9] and were extensively filtered to ensure the quality of the resulting corpus. Details on the filters

[6] `https://github.com/lollobrollo/Card-Recommender-System/blob/main/edh_scraper.py`

[7] `https://github.com/lollobrollo/Card-Recommender-System/blob/main/preprocess_cards.py`

[8] `https://magic.wizards.com/en/rules`

[9] `https://edhrec.com/articles/sitemap_index.xml`

applied can be found in the script `edh_scraper.py`[10]. To ensure consistency of text data, rules, oracle texts and article paragraphs are all preprocessed with the same function before the model sees them, both during training and inference. The same logic is also applied on user prompts in the final recommender system, introduced in section 3.4.

3.2 Oracle Text Embedding

To capture the functional semantics of a card's Oracle text, which is the primary determinant of its in-game role, a three-stage pipeline was developed to produce expert text embeddings. This process involves creating a domain-specific foundation model, generating a pseudo-labeled dataset for the target task, and then fine-tuning a final, specialized model via knowledge distillation. This process produces two distinct models: an expert in Oracle text embeddings, and an expert in classifying cards into hand-picked roles.

3.2.1 Domain-Adaptive Pre-training

A pretrained language model (distilbert-base-uncased) [25] is fine-tuned using a Masked Language Modelling (MLM) objective [26] on the composite corpus described in subsection 3.1.4. The tokenizer used by this model is extended with context specific jargon extracted from the corpus. A light model is used because fine-tuning is computationally expensive and later inference needs to be fast. This step produces a model that understands vocabulary and semantics from the game and is able to create meaningful text embeddings of Oracle texts. This model will also be used to create meaningful prompt embeddings in section 3.4. When used to create sentence embeddings, it computes embeddings for each token present in the prompt, then averages the results across each embedding to create a single vector that represents the given text. This training lasts for five epochs over a corpus of around 30 000 paragraphs.

3.2.2 Semi-Supervised Pseudo-Labeling

A powerful, pretrained Natural Language Inference (NLI) model (facebook/bart-large-mnli) [27, 12] is used in a Zero-Shot Classification setting [28] to generate soft probability scores against a set of gameplay roles previously defined. This set of roles is composed of sixteen hand-picked buckets in which cards can be assigned into most commonly. The model is allowed to associate a single card with multiple roles, as a card can be designed to serve multiple purposes. Labels with scores below a certain threshold have their scores lowered further, in order to reduce possible sources of noise during training in the next step.

[10] `https://github.com/lollobrollo/Card-Recommender-System/blob/main/edh_scraper.py`

3.2.3 Task-Specific Fine-tuning via Knowledge Distillation

In the last stage, the domain expert model is fine-tuned on the pseudo-labeled dataset obtained from the previous step. The training objective combines two complementary components:

- A knowledge distillation term, where the student model is optimized to reproduce the probability distributions of the teacher model using Kullback–Leibler divergence with temperature scaling [13];
- A binary cross-entropy loss on the teacher-provided labels that enforces absolute similarity between corresponding labels.

The final loss is expressed as a weighted combination of these two terms, controlled by the hyperparameter α. The outcome of this stage is an expert Card Role Classifier, whose output logits are used as a functionally-aware embedding of the card's Oracle text, which is concatenated to the card's vector representation.

3.3 CPR Model Definition and Training

The Neural Network that learns how to expand a deck can be considered the core of this work, and it is adapted from the work of Bertram et al [1]. As represented in Figure 3.3.1, it is composed of two networks that encode cards and decks separately, and a Siamese network that learns to project these embeddings into a common space. Training these networks together means that the two encoder towers learn to create the best representations for the Siamese network to learn relations between cards and decks. These networks are trained end-to-end together with a feature encoder, whose task is to create dense embeddings of sparse type and keywords multi-hot encodings, in order to complement card representations before they are processed by the two encoding towers.

3.3.1 Network Pipeline Configuration

The **feature encoder** provides two simple linear layers, specialized in encoding card type vectors and keyword vectors respectively. The module uses both layers to encode their respective input into a 64 dimension vector, and returns these two representations in a single concatenated tensor.

The **card encoder network** has the same architecture used in the work of Bertram et al. [1], being composed by a Multi-Layered Perceptron with four hidden layers of 1024 neurons each, connected by batch normalization, ELU activation and dropout.

The **deck encoder network** treats its input as a sequence of cards, so it uses six hidden layers with 1D Convolution, Group Normalization, Max Pooling 1D and ELU activations. The output of the convolutional layers is passed trough an Adaptive Max Pool 1D in order to be flattened, and then a last fully connected linear layer projects the output in a lower dimension. In the paper that inspired this architecture [1] Batch Normalization is used,

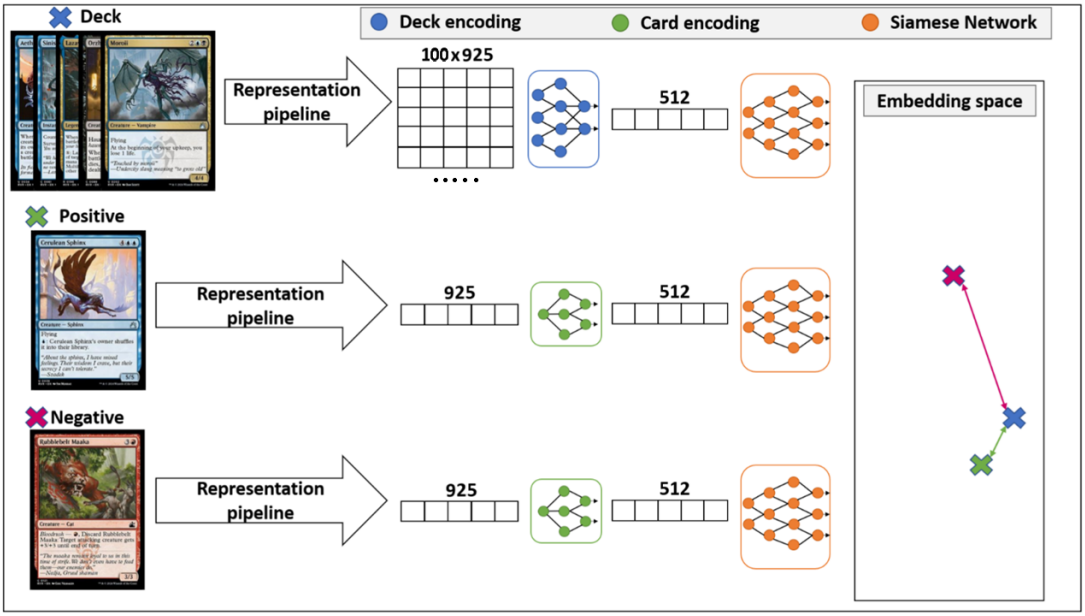

Figure 3.3.1: Overview of the pipeline during the training process, adapted from Bertram et al. [1].

but with small batch sizes it can compute noisy estimates of the parameters of the true data distribution; for this reason Group Normalization is preferred in this work.

The convolutional block is separable in two main sections. The first one follows the design principle of architectures like VGGNet [29] and ResNet [30], learns increasingly complex patterns by means of the increase in the number of channels. The second section, inspired by the decoder paths in encoder-decoder networks used in U-Net, symmetrically reduces the number of channels, learning to distill the abstract features learned in a low dimension. The number of channels is increased compared to the original work [1] since there are many more patterns that can be found in the EDH setting compared to Draft.

Both networks encode their input in a space with 512 dimensions, and they both feed into the same **Siamese network**. This network uses five hidden layers with 512 neurons, connected by normalization, dropout and ELU activation. The last layer uses the hyperbolic tangent function as the activation function, creating a final embedding space of dimension 512 where each value is squashed in the range [-1,1].

3.3.2 Dataset and Training Process

In this section, the pipeline training process is described, separating the two main training approaches in two subsections: training using Triplet Loss and training using InfoNCE Loss. In both cases, the model is trained on the two distinct datasets introduced in subsection 3.1.2. Another distinction is made regarding the number of training epochs. In chapter 4, the results of these different training settings are briefly compared.

Triplet Loss Approach

During training, the model expects batches of triplets of the form [*anchor, positive, negative*], where the anchor is a partial deck, the positive is a card that expands it well and the negative is a card that is worse at expanding the deck compared to the positive.

Since decks within a batch may vary in size, a custom collate function is implemented to pad the anchor sequences, ensuring dimensional compatibility. As discussed in subsection 3.1.3, the complete card representations have to be computed during a training step. For this reason, the Dataset is designed to return tensors containing precomputed, partial representations for all cards in the anchor decks and the positives, along with all their corresponding multi-hot encodings for card types and keywords, which will be handled in the step function to create a complete representation for all cards in the batch. The negative sample is selected during a training step, as explained below.

These triplets are generated and provided to the model via a custom Dataset and a custom step function. The Dataset is initialized with a list of decklists (either the complete set or a diversified subset) and produces, for each sample, an anchor, constructed as a random subset of a single deck, and a positive example, which is a card sampled from those excluded when selecting the anchor.

Knowing that the strategy for selecting the negative example has a significant impact on the stability and convergence of the training [20, 21], the negative sample is not chosen randomly. Instead, it is selected through distance-weighted sampling within the batch, favoring harder negatives.

Within the step function, which receives a batch of anchors and positives, cosine similarity is computed for all normalized embeddings of cards and decks in the batch. For each anchor-positive pair, the negative example is sampled probabilistically, with a higher likelihood for samples that are more similar to the positive, thereby ensuring harder negatives are prioritized while maintaining diversity.

The learning rate is controlled by a linear scheduler with an initial warmup phase. In this phase, the rate is gradually increased from zero to its maximum value over a predefined number of steps, mitigating instability due to the high variance of gradients at random initialization. After reaching the maximum learning rate, the scheduler decreases it linearly until training completion, ensuring a smoother optimization trajectory and promoting stable convergence.

This pipeline is trained twice, the first time for 20 epochs, and the second time for 200 epochs. Given the complex nature of deck building in EDH, each deck can be reused many times thanks to the shuffling of the anchor and the different positive examples.

InfoNCE Loss Approach

The training pipeline presented for the Triplet Margin Loss can be adapted with minor changes to employ the InfoNCE loss, which benefits from leveraging a larger number of negative samples compared to the triplet loss.

The main difference lies in the training step function. Card embeddings and the similarity

matrix are computed in the same manner, but explicit negative sampling is unnecessary. Instead, the cross-entropy loss directly operates on the similarity matrix, treating it as logits for contrastive learning.

3.4 Recommendation Engine and Retrieval System

Following the training of the embedding models detailed in section 3.3, a vector database is constructed to facilitate efficient card retrieval. The system utilizes a persistent ChromaDB [31] client. Within this database, a distinct collection is populated for each variant of the trained Contextual Preference Ranking (CPR) pipeline.

Each document within a collection represents a unique *Magic: The Gathering* card and is composed of two primary components: its high-dimensional vector embedding, as generated by the corresponding model, and a set of structured metadata. This metadata, extracted during the preprocessing phase, includes critical game attributes such as color identity, converted mana cost (CMC), card type, and gameplay keywords, which can be used to filter the results of queries.

3.4.1 The Multi-Stage Retrieval Pipeline

The recommendation process is not a single database query but rather a multi-stage pipeline designed to balance synergy, user intent, and result diversity. The pipeline dynamically adapts its strategy based on whether a user provides a natural-language prompt.

Context Vector Generation

The process begins with the generation of a context vector for the user's partial deck. The deck is converted into its corresponging vector representation, and it is later processed through the trained CPR model's deck tower to produce a single 512-dimensional embedding, $f(D)$. This vector represents the deck's core strategy and serves as the baseline for all subsequent similarity calculations.

Prompt-Guided Query Modification

If a user provides a prompt specifying what type of card they are searching for, the system employs a **vector offset method** to translate this textual instruction into a directional modification of the query. The prompt is processed by the domain-adapted language and role classifier models to create a "pure prompt" representation, where semantic and functional features are placed within a neutral card scaffold. This representation is then passed through the CPR model's card tower to produce a 512-dimensional **direction vector**, v_{prompt}. This vector represents the prompt's isolated intent within the synergy space.

Candidate Pool Generation

To ensure a rich and varied set of potential recommendations, the system first generates a large candidate pool. The method of generation depends on the presence of a user prompt.

With a Prompt: A multiple querying strategy is used to explore the synergy-to-intent spectrum. The final query vector, Q_{final}, is a normalized linear combination of the deck's context vector and the prompt's direction vector:

$$Q_{final} = \frac{f(D) + \alpha \cdot v_{prompt}}{\|f(D) + \alpha \cdot v_{prompt}\|_2} \tag{3.1}$$

The system performs multiple parallel queries to the vector database using a range of values for the scalar weight α. The results from each query are aggregated into a single, de-duplicated candidate pool. This technique ensures that the pool contains cards that are purely synergistic ($\alpha = 0.0$), purely prompt-relevant (α is large), and a balanced mix in between.

Without a Prompt: In a synergy-focused search, the system performs a single query using the deck embedding $f(D)$ to retrieve a candidate pool significantly larger than the final number of requested recommendations.

Filtering and Re-Ranking

Before the final ranking, the candidate pool is filtered to remove any cards that are already present in the user's input deck, ensuring that all suggestions are novel additions.

The refined candidate pool is then re-ranked using one of two **Maximal Marginal Relevance (MMR)** [17] strategies to ensure the final output is both relevant and diverse. Both methods aim to optimize the trade-off between relevance to the user's query and novelty with respect to already selected items.

The first method is the classical, deterministic MMR algorithm. At each step, it selects the single candidate from the remaining pool that has the highest MMR score. This approach will always yield the exact same set of recommendations for a given query and candidate pool.

To introduce variety in the results, a second, non-deterministic method was implemented. At each step of the iteration, this stochastic re-ranking approach performs the following:

1. It calculates the MMR scores for all remaining candidates.
2. It identifies a small subset of the top-m candidates.
3. It applies a softmax function to the scores of these top-m candidates, converting them into a probability distribution.
4. It samples the next card to be added to the results from this distribution, and removes it from the candidate pool.

This probabilistic selection ensures that while the highest-scoring cards are heavily favored, other strong but slightly different candidates are given a chance to appear in the final list. This method maintains a high standard of relevance while allowing the system to present a varied set of suggestions across multiple identical queries.

3.5 Online Demo

Evaluating the performance of a recommender system for a domain as complex and subjective as *Magic: The Gathering* presents a significative challenge. The computational intractability of the game and the context-dependent nature of card synergy prevent the use of simple, automated evaluation metrics. A "correct" recommendation cannot be determined algorithmically yet, but it is a qualitative judgment players can estimate based on strategic goals, budget, and personal playstyle. To meaningfully assess and compare the performance of the various trained models, this thesis employs a qualitative methodology based on human experts feedback. This is made possible by developing an interactive online demonstration to collect structured feedback directly from experienced human players, providing a reliable source of validation data.

3.5.1 Implementation of the Interactive Demo

The evaluation tool was built as an interactive web application using Gradio [32], an open-source Python library designed for creating user-friendly interfaces for machine learning models. The choice of Gradio was motivated by its ability to rapidly develop a functional and intuitive demo directly from the project's existing Python codebase, as well as the possibility to host and share the demo trough it. The complete implementation can be found in the script `app.py`.

The demo provides a user interface where players can input the necessary parameters to generate a set of recommendations:

1. **An Archidekt Deck URL:** The user provides a link to a public deck present in the Archidekt [24] platform. The system is designed to parse this URL and extract the deck's unique identifier.
2. **An Optional Text Prompt:** The user may provide a natural-language prompt to guide the search towards a specific functional need (e.g., "find some card draw" or "I need more removal"). If left blank, the system performs a synergy-only search.
3. **Model Version Selection:** A dropdown menu allows the user to select which of the eight trained CPR model variants should be used for the query. This feature is crucial for enabling a direct, side-by-side comparison of the different training configurations.

Upon submission, the system executes the multi-stage retrieval pipeline described in section 3.4, retrieving the top 10 recommended cards. The results are then displayed to the user as a gallery of card images, providing an immediate and visually clear output.

3.5.2 Feedback Collection Mechanism

After the recommendations are displayed, a dedicated feedback interface becomes visible within the application. For each of the 10 recommended cards, the card's name is displayed above a slider, allowing the user to provide a rating on a 5-point scale, ranging from 1 (a poor or irrelevant suggestion) to 5 (an excellent and highly synergistic suggestion).

When the user submits this feedback, a structured data record is generated and appended as a new line to a persistent log file on the host machine. Each record is a JSON object containing a comprehensive set of metadata about the query, including a timestamp, the specific model version used, the deck ID, its color identity, the user's prompt, and a list of the recommended cards paired with their user-assigned ratings. This structured and automated data collection method creates the valuable dataset that forms the basis for the analysis presented in the following chapter.

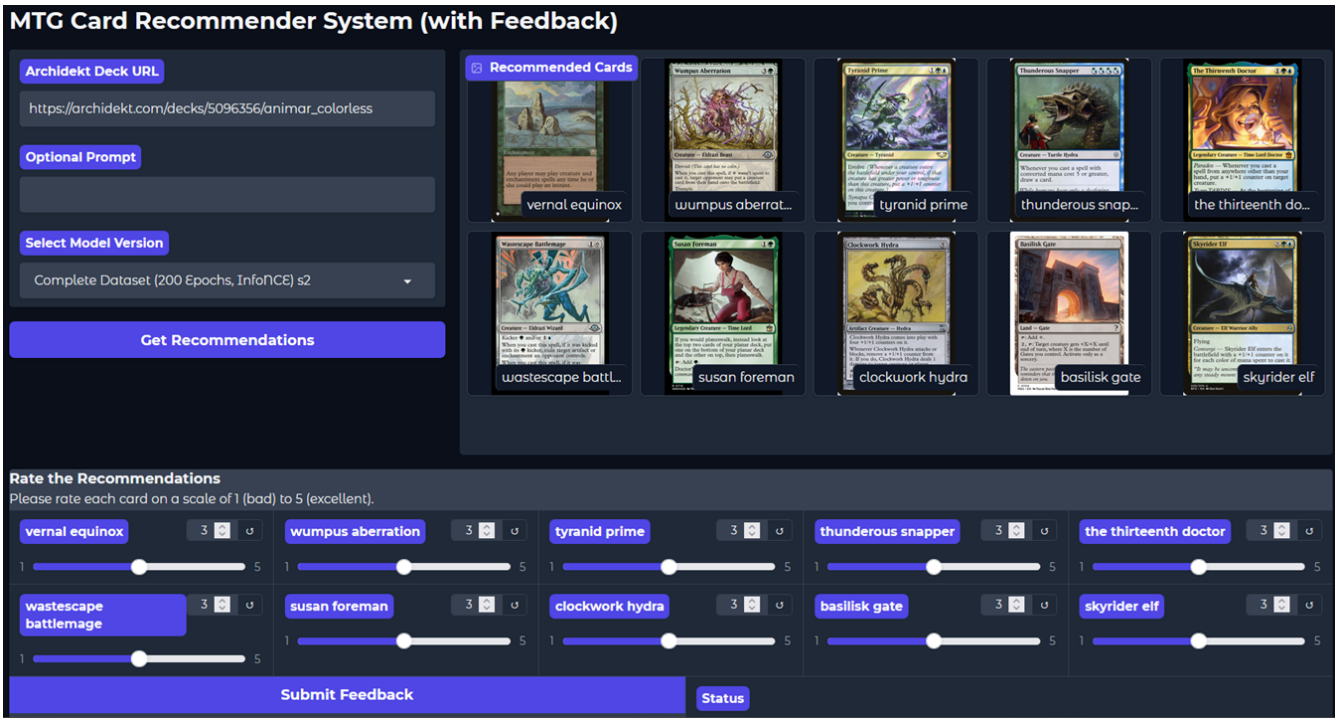

Figure 3.5.2: Example of the working demo: users provide the link to their deck, choose the model to be used and get relevant recommendations. It is also possible to rank results and submit the feedback.

Chapter 4

Model Evaluation and Results

This chapter presents a detailed analysis of the user feedback collected via the interactive online demo described in section 3.5. The primary goal of this evaluation is to assess the performance of the eight distinct Contextual Preference Ranking models developed for this work. The analysis focuses on understanding the impact of three key training variables: the dataset used (Diversified vs. Complete), the loss function (Triplet Margin vs. InfoNCE), and the number of training epochs (20 vs. 200). Re-ranking of intermediate results is done using the deterministic version of Maximal Marginal Relevance.

4.1 Analysis of User Feedback

The feedback, comprising 470 individual card ratings, was processed and aggregated to derive statistics for each model variant. The primary metric for evaluation is the **average user rating** on a scale of 1 to 5. The distribution of these ratings is also examined to examine the distribution of the recommendations.

Figure 4.1.1 provides a comprehensive visualization of these results. The grid layout displays four subplots containing the rating distribution for a specific combination of dataset and loss function, while the color hue distinguishes between training durations.

4.1.1 Results Discussion

Several distinct and, in some cases, surprising trends emerged from the analysis of the user feedback data.

Impact of Training Duration and Dataset Interaction

The most significant finding is the strong interaction effect between the training duration and the dataset used. For models trained on the larger **Complete Dataset**, increasing the training from 20 to 200 epochs resulted in a substantial improvement in performance. The average rating for the Triplet model rose from 1.93 to 2.45, and for the InfoNCE

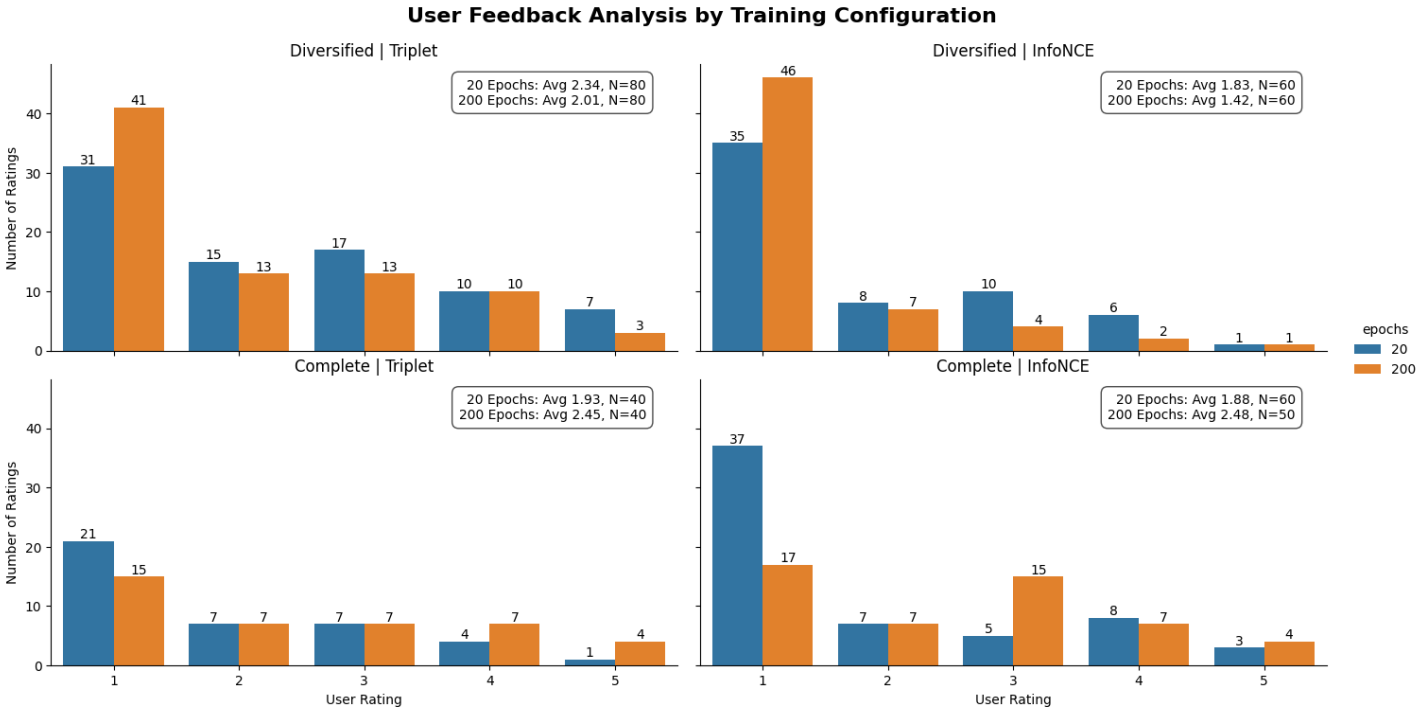

Figure 4.1.1: User feedback analysis conditioned by training configuration, using deterministic MMR. Each subplot shows the rating distribution (1=bad, 5=excellent) for a specific dataset and loss function. The color hue distinguishes between models trained for 20 epochs (blue) and 200 epochs (orange). The text box within each plot provides the average rating and the total number of ratings (N) for each configuration.

model from 1.88 to 2.48. This is visible in the bottom row of Figure 4.1.1, where the 200 epoch (orange) distributions are visibly shifted towards higher ratings compared to their 20 epoch (blue) counterparts.

Conversely, for models trained on the smaller **Diversified Dataset**, a longer training duration was detrimental. The average rating for the Triplet model decreased from 2.34 to 2.01, and the InfoNCE model's performance collapsed from an average of 1.83 to 1.42. This suggests that while the diversified dataset may contain cleaner information, its smaller size makes the model prone to overfitting during extended training, causing it to learn patterns that do not generalize well to real-world user decks. The larger, more varied **Complete Dataset**, despite containing overrepresented color combinations, provides a more robust foundation for learning, benefiting significantly from a longer training schedule.

Impact of Loss Function

The choice of loss function also had a noticeable impact on performance, which was again dependent on the dataset. On the smaller Diversified dataset, the **Triplet Margin Loss** consistently outperformed the InfoNCE loss. However, on the larger Complete dataset, the **InfoNCE loss** achieved a marginally higher average rating after 200 epochs (2.48 vs. 2.45), suggesting its effectiveness at leveraging the larger number of in-batch negatives available in a bigger dataset.

More results on Best Performing Models

The models that performed best during the previous test are compared further with new data. Testers asked for more variance in the retrieved cards, so the deterministic MMR function in this stage is substituted with its stochastic version.

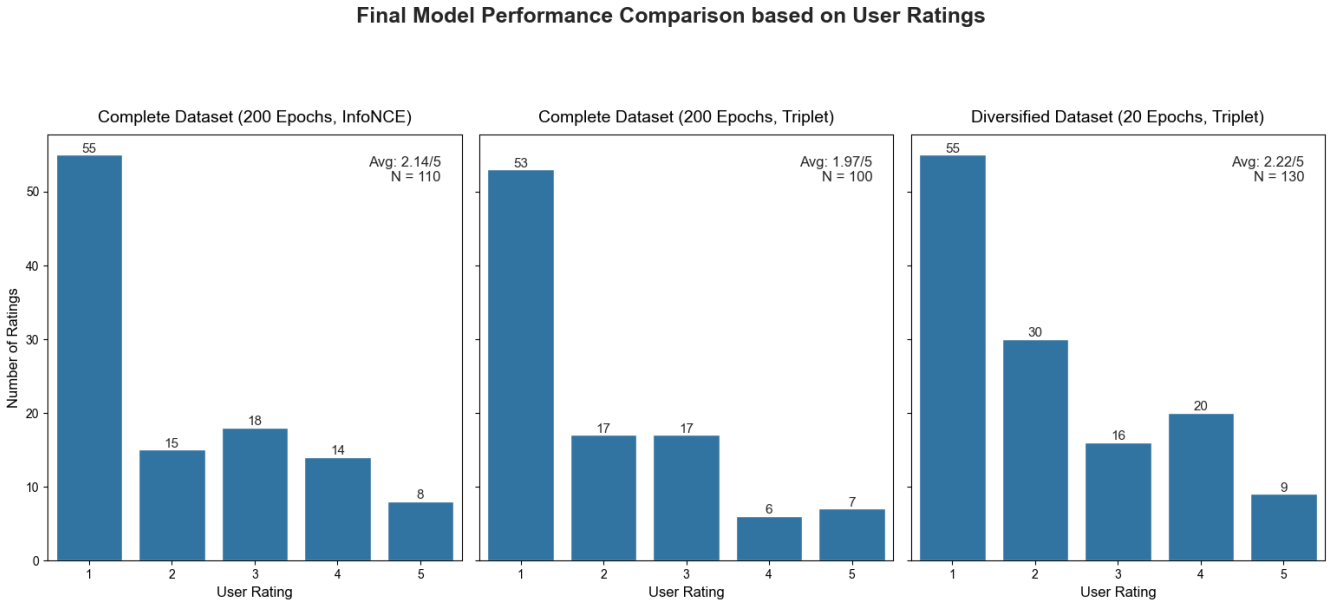

Figure 4.1.2: User feedback analysis on the top 3 models, using stochastic MMR.

Looking at Figure 4.1.2, evaluations given to the retrieved cards are concentrated on the lower ranks, meaning that many cards suggested by the system were distant from the context of the provided deck. Given this perspective, it is mandatory to ask why this is the case: the following list encapsules some possible explanations.

1. The size of the complete dataset, while extensive by itself, still was not sufficient to fully depict the plethora of ever expanding strategies available in *MTG*, resulting in lower precision from the model;

2. The higher complexity of some deck builds is not captured well by the system, resulting in poor performance especially for decks with a richer color identity;

3. The time at disposal to design, implement, deploy and test the full contraption fell short from expectations as implementation and gathering of human feedback proved more time consuming than anticipated.

Interestingly, testers observed that several models frequently suggested "Sliver" cards, despite no Sliver decks being included in the evaluation. This behavior is likely caused by the structure of the embedding space, where Sliver cards are embedded tightly due to their highly synergistic design. In Figure 4.1.3, the previous results are reconsidered after filtering out 38 Sliver-related suggestions.

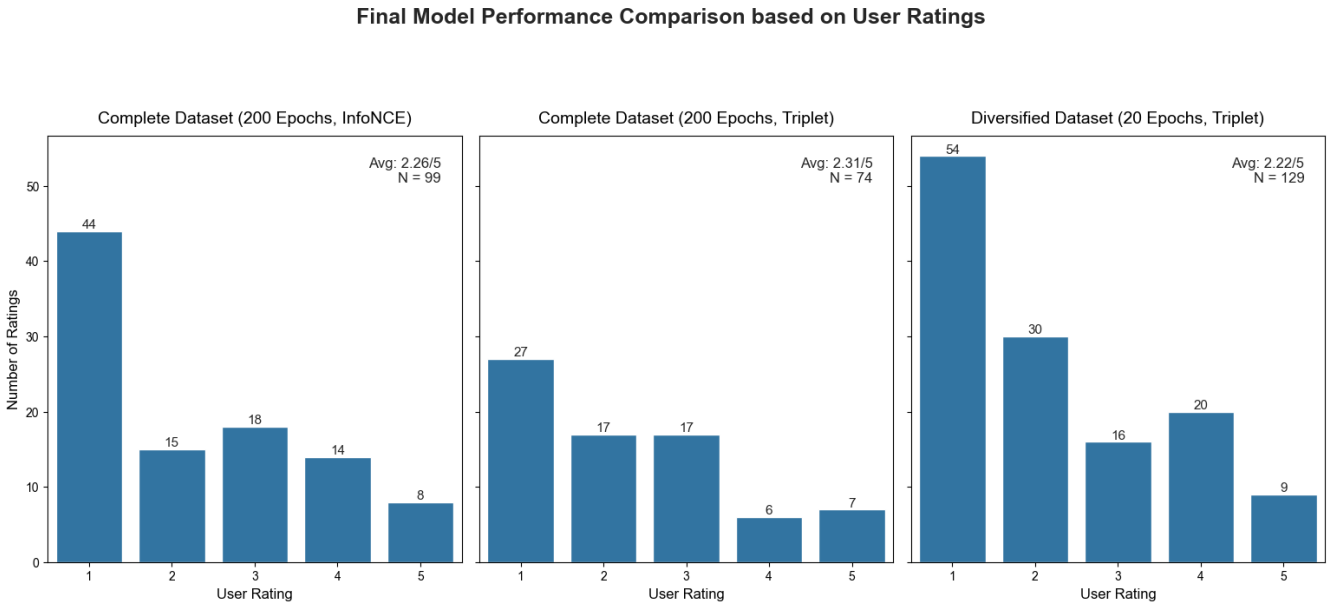

Figure 4.1.3: User feedback analysis on the top 3 models, using stochastic MMR, omitting Sliver results.

Removing sliver suggestions led to a modest improvement in the average ratings across the board. Notably, the analysis revealed that this failure mode was predominantly associated with models trained using the Triplet Margin Loss objective. The best-performing InfoNCE model, by contrast, produced only a single Sliver suggestion in the entire evaluation of non-Slver related decks. This indicates that InfoNCE provides a stronger implicit regularization effect than Triplet Loss, leading to more robust generalization.

4.1.2 Performance of Prompt-Guided Search

A qualitative analysis of the feedback logs revealed a consistent limitation across all model configurations: the performance of prompt-guided searches. While the vector offset method is theoretically sound [33], user feedback indicated that recommendations generated from non-empty prompts were not relevant to the users' specific interests.

This suggests that while the domain-adapted foundation model has a general grasp of *MTG* semantics, it lacks the understanding necessary to translate user requests into effective directional vectors within the CPR model's synergy space. The model appears to struggle with the ambiguity of natural language, often returning cards that are not relevant for the given prompt. Improving the efficacy of prompt-guided search is therefore a primary area for future work, as detailed in chapter 5.

4.1.3 Conclusions

Based on the collected user feedback, the best performing model configuration was the one trained on the **Complete Dataset for 200 Epochs using the InfoNCE loss function**. This model achieved the highest average user rating in the testing sessions, and demonstrated the most significant performance improvement with extended training. This indicates that a large, diverse dataset combined with a data-efficient contrastive loss function and a sufficient training duration is the most effective combination for learning the complex, synergistic relationships inherent to the Commander format. While this suggests that Contextual Preference Learning framework can be used to model player choices in the EDH format, the current configuration seems insufficient to bring out the best results possible from the CPR framework, when considering the user rankings alone.

While the aggregate average ratings for recommendations were modest, **testers consistently expressed satisfaction with the cards proposed by the system**. This suggests that the perceived utility of the recommendations is not dependent on the quality of all given suggestions, but rather on the **presence of a few highly impactful discoveries within the result set**. From this perspective, the models can be considered effective considering that each query returns some relevant suggestions to the users.

Having said that, the current configuration heavily relies on the CPR model to return meaningful suggestions to the user, as the focus was on observing the generalization capabilities of the CPR framework. For this reason, as stated in section 5.3, the quality of the recommendations can be greatly improved by working on the retrieval system, knowing that while unsatisfactory, the knowledge learned by the CPR model is far better than picking random cards.

Chapter 5

Future Improvements and Directions

While the system presented in this thesis demonstrates the viability of the Contextual Preference Ranking framework for Commander card recommendations, it also shows much room for improvement. This chapter outlines several directions for future research and development, categorized into data-centric refinements, advancements in the core model and training methodology, and retrieval system optimizations.

5.1 Data-Centric Refinements

Results show that the best performing model can improve simply by increasing the dataset size, but the quality of the training data also plays a major role for the model's performance. The following improvements focus on the interpretation of input data to get the most out of the best samples.

5.1.1 Deck Quality Weighting and Ranking

The current approach assumes that all decklists in the training corpus are of equal quality, ignoring the difficulty in assessing the "goodness" of a deck. Future work could move beyond this assumption by implementing a deck weighting scheme. Metrics such as those explored by Salubrious Snail's EDHRECrec project[1] - namely **Synergy** (measuring thematic focus) and **Entropy** (measuring strategic diversity) - could be used to quantify a deck's quality. A deck with high synergy and low entropy could be considered a strong training signal and be up-weighted during training. Conversely, a high-entropy "good stuff" deck might be down-weighted. By weighting the contribution of each deck to the training loss, the model could learn to prioritize more coherent and synergistic strategies, mitigating the influence of sub-optimal or sparse decklists.

[1] `https://www.salubrioussnail.com/edhrecrec`

5.1.2 Balancing for Niche Strategies

The training data is naturally biased towards popular archetypes, which appear more frequently in the retrieved decklists. This can result in the model being less knowledgeable about niche but viable strategies (e.g., "Snow tribal", "Enchantment Reanimator"). To address this, the training process could be re-balanced, for example oversampling decks with a high "Hipster Card" score, another indicator introduced in EDHRECrec which measures how many cards with a low overall play rate are present in a specific deck. This would ensure the model is more frequently exposed to underrepresented synergies, improving its ability to make interesting recommendations for less common commanders.

5.2 Model and Training Enhancements

Advancements in the model architecture and training process can surely yield significant improvements in terms of performance and quality of the results.

5.2.1 Upgraded Language Models

This work deliberately utilized open-source and relatively small language models like DistilBERT [25] to ensure the project was computationally tractable on an average computer. However, leveraging larger, more sophisticated models for text embedding and prompt interpretation would almost certainly improve performance, as state-of-the-art models possess a more profound understanding of language. Using such models to generate the semantic and role features for cards and to interpret user prompts would likely lead to a significant improvement in the quality of card embeddings.

5.2.2 Reinforcement Learning from Human Feedback

The online demo described in section 3.5 can be used to gather a continuous stream of human-generated preference data. This data is an ideal resource for implementing Reinforcement Learning from Human Feedback (RLHF) [34].

In order to implement this approach, collected human feedback would be used to train a reward model, which should be able to associate human preferences with a numerical reward signal. This reward model would then be used to fine-tune the CPR pipeline via policy optimization, paying attention to how much the reward model should be allowed to change the pretrained model policy. This would optimize the recommender system to directly generate cards that it predicts will be highly rated by human players.

5.2.3 Advanced Contrastive Loss Formulations

The current implementation utilizes a standard in-batch InfoNCE Loss which, while effective, operates on the assumption that all other samples in a batch are true negatives.

Recent research has highlighted key improvements to go beyond this formulation.

Papers like the work by Wu et al. [35] and Rusak et al. [36] propose improvements in how the InfoNCE Loss is used, respectively addressing the issue of sampling bias due to false negatives - samples treated as negative examples that are semantically similar to the anchor - and exploring how contrastive learning objectives depend strongly on hyperparameter choices and dataset characteristics. The introduction of this improved Loss in the training cycle could result in an improvement of the embeddings quality.

5.3 Retrieval and Re-ranking Enhancements

The current system employs a dense vector retrieval pipeline followed by MMR re-ranking. While effective, its precision and flexibility could be significantly improved by incorporating a multi-stage retrieval and re-ranking techniques.

5.3.1 Hybrid Retrieval with Sparse and Dense Systems

The current system relies on dense, embedding-based retrieval, which excels at capturing semantic and synergistic meaning but can fail when queries contain specific, literal keywords, for example referring to a specific card ability. To address this limitation, a more robust approach is to implement a **hybrid retrieval system** [37] that combines the strengths of both dense and sparse methods.

This architecture would augment the dense vector search with a parallel sparse retrieval engine based on a lexical matching algorithm like Okapi BM25 [38]. Sparse retrieval operates on an inverted index of terms and excels at exact keyword matching. To construct the document corpus for such a system, each card needs to be represented as a "bag of words" containing all of its associated text, including its name, type line, explicit keywords, and full Oracle text. For this representation to be effective, standard natural language processing techniques such as stop-word removal, stemming, and lemmatization can be applied to normalize the text and improve the accuracy of term matching.

The final step in a hybrid system is to combine the ranked lists generated by the dense and sparse retrievers using a rank fusion algorithm, which calculates a combined score for each document based on its position in both lists. The resulting fused ranking produces a candidate pool that benefits from the semantic understanding of the dense model and the keyword precision of the sparse model, leading to more accurate and reliable search results.

5.3.2 Query Expansion

User prompts are often short and can be ambiguous. Query Expansion techniques aim to enrich the initial query to better match the user's underlying intent, thereby improving recall. Two prominent methods could be applied:

1. **Pseudo-Relevance Feedback:** This classic IR technique, based on the Rocchio algorithm, uses the initial search results to refine the query. The system would first perform a search with the user's prompt, assume the top-k results are relevant, and then shift the original query vector towards the centroid of these results. This new, refined vector is then used for a second, more accurate search.

2. **Generative Query Expansion:** Modern language models can be used to reformulate or expand the query. For instance, a model could be prompted to generate synonyms and related strategic concepts (e.g., "card advantage" might be expanded to include "card draw", "refill hand"). This approach would create a richer, more comprehensive query vector that is more likely to match relevant cards.

5.3.3 Advanced Metadata Filtering

Finally, the system's practical utility could be enhanced with more sophisticated, user-driven filtering on card metadata. Beyond just color identity, users could be empowered to constrain the search results by:

- **Card Type:** Explicitly requesting only "creatures", "instants", or "artifacts".
- **Converted Mana Cost (CMC):** Filtering for cards within a specific mana cost range to fit their deck's curve.
- **Specific Keywords:** Requiring the presence or absence of specific keywords like "Flying" or "Trample".

Integrating these optional filters would transform the tool from a pure discovery engine into a more precise and practical deck-building assistant, allowing users to find cards that are not only synergistically relevant, but also perfectly aligned with the functional slot they are trying to fill.

Bibliography

[1] Timo Bertram, Johannes Fürnkranz, and Martin Müller. Learning with generalised card representations for "magic: The gathering". In *2024 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2024.

[2] Alex Churchill, Stella Biderman, and Austin Herrick. Magic: The gathering is turing complete. *arXiv preprint arXiv:1904.09828*, 2019.

[3] Krishnendu Chatterjee and Rasmus Ibsen-Jensen. The complexity of deciding legality of a single step of magic: the gathering. In *Proceedings of the Twenty-Second European Conference on Artificial Intelligence*, ECAI'16, page 1432–1439, NLD, 2016. IOS Press.

[4] Howe Choong Yin and Alex Churchill. A Programming Language Embedded in Magic: The Gathering. In Andrei Z. Broder and Tami Tamir, editors, *12th International Conference on Fun with Algorithms (FUN 2024)*, volume 291 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[5] Chris Alvin, Michael Bowling, Shaelyn Rivers-Green, Deric Siglin, and Lori Alvin. Toward a competitive agent framework for magic: The gathering. In *The International FLAIRS Conference Proceedings*, volume 34, 2021.

[6] Henry N Ward, Bobby Mills, Daniel J Brooks, Dan Troha, and Arseny S Khakhalin. Ai solutions for drafting in magic: the gathering. In *2021 IEEE conference on games (CoG)*, pages 1–8. IEEE, 2021.

[7] Samuel Ojala. Large language models as recommender systems, 2024.

[8] Timo Bertram, Johannes Fürnkranz, and Martin Müller. Predicting human card selection in magic: The gathering with contextual preference ranking. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021.

[9] Edhrec. Accessed: 29 August 2025.

[10] lollobrollo. Card recommender system. `https://github.com/lollobrollo/Card-Recommender-System`, 2025.

[11] Joshua Kritz and Raluca Gaina. When 1+1 does not equal 2: Synergy in games. *arXiv preprint arXiv:2502.10304*, 2025.

[12] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[14] Kenneth Reitz and Contributors. Requests: Http for humans. `https://requests.readthedocs.io/`, 2025. Version 2.32.5; accessed: 16 September 2025.

[15] Leonard Richardson. Beautiful soup documentation. `https://beautiful-soup.readthedocs.io/`, 2024. Beautiful Soup 4; accessed: 16 September 2025.

[16] Yu. A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 42, pages 824–836, 2018.

[17] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[18] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 6, pages 737–744, 1993.

[19] Gregory R. Koch. Siamese neural networks for one-shot image recognition. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, 2015.

[20] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2840–2848, 2017.

[21] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[23] Scryfall. Accessed: 29 August 2025.

[24] Archidekt. Accessed: 29 August 2025.

[25] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[26] Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*, 2021.

[27] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, 2020. Association for Computational Linguistics.

[28] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *arXiv preprint arXiv:1909.00161*, 2019.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[31] Chroma, Inc. Chroma: Open-source vector database for ai applications, 2025. Version 1.0.20, `https://trychroma.com/`.

[32] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019. Accessed: 16 September 2025.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[34] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[35] Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. Rethinking infonce: How many negative samples do you need? *arXiv preprint arXiv:2105.13003*, 2021.

[36] Evgenia Rusak, Patrik Reizinger, Attila Juhos, Oliver Bringmann, Roland S Zimmermann, and Wieland Brendel. Infonce: Identifying the gap between theory and practice. *arXiv preprint arXiv:2407.00143*, 2024.

[37] Xueguang Ma, Shuguang Deng, and Ming Chen. A replication study of dense passage retriever. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2375–2381, 2021.

[38] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. *Nist Special Publication Sp*, 109:109–126, 1994.