Chapter 1

Introduction

This chapter provides a general introduction to the different topics covered in the rest of the thesis. In Section 1.1, we provide a general overview of Reinforcement learning along with multi armed bandits. Sections 1.2 gives a description about policy gradient algorithms and actor-critic methods. In section 1.3 we discuss the ranking in search systems and recommender systems. Further, in section 1.4 we describe different applications of search and recommender systems. Section 1.5 and 1.6 discuss about the challenges and motivation of our thesis. Finally, section 1.7 provides contribution of the thesis.

As the volume of data on the internet continues to grow rapidly, the challenge of providing users with the most relevant items or documents becomes increasingly crucial. Providing relevant data to users is a fundamental goal for various platforms and services, whether it's in the context of search engines, content recommendations, e-commerce, or information retrieval. Furthermore, the relevant items or documents should also be at the top of the search list as the order in which search results are displayed significantly influences whether users find what they're looking for quickly and efficiently. Ranking refers to sorting documents according to relevancy in order to locate information relevant to a search. Although this is a basic issue with information retrieval, this challenge also occurs in many other applications. With access

to the user profile we can apply it to different tasks such as in search engine, recommender system, ecommerce, travel agencies sites etc. With a user profile having access to the users age, location, sex, etc. we can sort the web pages by relevance. Similiarly we can build a user profile with purchase history of item or rating profiles of the user and provide recommendation to the new user.

Machine learning is an area of artificial intelligence (AI) and computer science that uses data and algorithms to approximate how humans learn, gradually improving its accuracy. Machine learning is a critical component of the rapidly expanding discipline of data science. Machine learning employs a variety of algorithms to construct mathematical models and make predictions based on past data or information. It is being utilised for different applications such as image identification, audio recognition, email filtering, search engine, recommender systems, and many others. Machine learning can be classified into three types: Supervised learning, Unsupervised learning and Reinforcement learning. Supervised learning is a form of machine learning method in which we feed sample labelled data to the machine learning system in order to train it, and it predicts the output based on that. The system builds a model using labelled data to interpret the datasets and learn about each data. After training and processing, the model is tested by supplying sample data to see if it predicts the precise output or not. In unsupervised learning the model is trained given a set of unlabeled, classified, or categorised data, and the algorithm is expected to act on the data without supervision. Unsupervised learning attempts to restructure input data into new features or groups of objects with similar patterns. The third type is Reinforcement learning that performs the learning without any explicit labels, however it treats the feedback as some form of explicit data and thus is referred to as the semi supervised learning.

1.1 Reinforcement learning

Reinforcement learning(RL) is a machine learning technique where the agents learns to interact in an environment by taking actions based on the feedback received from

environment. Reinforcement learning is among the three frameworks of machine learning along with supervised and unsupervised learning. While there are no explicit labels for agents in reinforcement learning like in supervised algorithms, it treats feedbacks obtained from the environment in terms of reward as implicit labels. In reinforcement learning, the agent learns the policy that can maximize long-term cumulative reward through interacting with the environment [32]. In the context of Reinforcement Learning (RL), Markov Decision Processes (MDPs) play a central role as a formalism for modeling sequential decision-making problems. MDP consists of a tuple of five elements (S, A, P, R, γ) , where S is the set of states, A is the set of action, $T(s_{t+1}|s_t, a) : S \times A \times S \to \mathbb{R}$ is the transition probability of reaching state s_{t+1} after executing action a on state s_t. , $R(s, a) : S \times A \to \mathbb{R}$ is the immediate reward after executing action a from state s_{t+1} , and γ is the discount factor. The discount factor is a measure of how much does the agent weighs the immediate rewards relative to future rewards. It's value is in range 0-1, where 0 means agent only cares about the immediate reward and 1 if agents includes all the future rewards.

A policy in MDP defines mapping of states to actions, deciding what action to choose in current state. Solving an MDP typically refers to finding a policy $\pi : S \to A$ such that from any given state s, executing action $\pi(s)$ and then acting optimally(best actions). The goal is to learn the optimal policy π^* , i.e. the policy that gives the maximum reward in the episode by choosing the actions that maximizes the return of the current state. The reinforcement learning framework is shown in Figure 1.1 depicting an RL agent receiving the state from the environment and generates an action. The envirionment further provides the feedback to the action taken by the agent. The episode in Reinforcement learning is a sequence where agent interacts with an environment with an intitial state and a terminal state. The agent start from initial state takes the action and then receives the reward. Finally the agent stops at the terminal state. A trajectory could be any arbitrary sampled sequence of s, a, r from an episode.

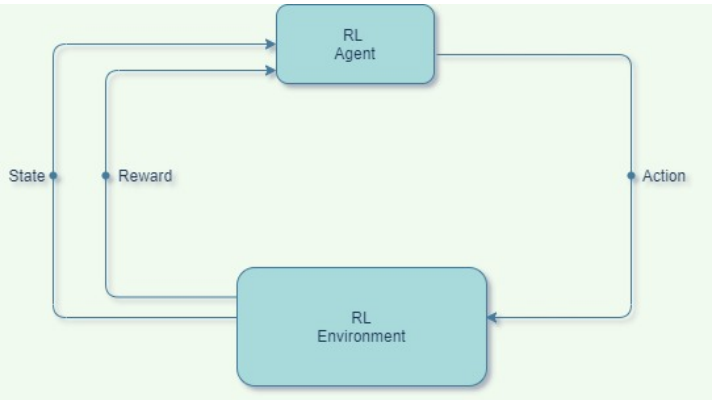

Figure 1.1: Reinforcement learning framework

In RL, the goal of the agent is to maximize the expected cumulative reward. So, the agent needs to find an optimal policy that resembles the optimal strategy for the agent to act in an environment. A policy is generally a function that outputs an action to take, given the current state of the environment outputs an action and can either be stochastic or deterministic.

For example, consider the Cartpole game in which a pole is attached by a joint to a cart, which moves along a frictionless track. The state consists of cart position, cart velocity, pole angle, and pole velocity at the tip. The system is controlled by applying a force of +1 or -1 to the cart (moving left or right). The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The game is divided into terms of episodes, each containing a specific number of steps. The episode ends when the pole is more than 15 degrees from vertical or the cart moves more than 2.4 units from the center. The goal of the game is to have the cumulative reward as high as possible. The RL agent explores the environment all by itself by playing the game multiple times until it learns the correct way to play the game and learns to balance the pole. The agent learns which actions leads to a higher reward in the corresponding state based upon the feedback from the environment and thus gradually learns the optimal action in a state that stabilises the pole. Thus, RL

model deals with learning of the optimal behaviour based on interaction with the environment based on feedback mechanism without any explicit labeled data.

Reinforcement learning methods are generally categorized as value based or policy based methods. Value based approaches learn value functions or Q functions in order to learn the value of states (or acquire an estimate for the value of states) and actions. After that, it employs policy extraction to obtain a policy for making decisions. The value based methods aim to find the value of each state , which can be obtained through the Bellman equation, $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$, where $v_\pi(s)$ denotes the value of the state s in policy π, $v_\pi(s')$ denotes the value of the state s' in policy π, $\pi(a|s)$ denotes probability of choosing action a in state s, $p(s', r|s, a)$ denotes state transition probability, r represents the reward at state s, and γ denotes the discount factor. The state value function is the expected return starting from state s following policy π is given as : $V^\pi(s) = E_\pi\{G_t|s_t = s\}$, where G_t denotes cumulative discounted reward from time step t. Further, $Q^\pi(s, a)$ denotes the state-action value function. It is the expected return starting from state s , taking action a , then following policy π, $Q^\pi(s, a) = E_\pi\{G_t|s_t = s, a_t = a\}$.

1.1.1 Temporal Difference (TD) learning

Temporal Difference (TD) learning is a popular concept in the field of artificial intelligence and reinforcement learning. It is a learning algorithm used by agents (e.g., robots, game-playing AI, etc.) to learn and improve their decision-making process based on feedback they receive from their environment. The term "temporal difference" refers to the way the agent updates its knowledge about the environment over time by making predictions and then correcting those predictions when it receives new information (feedback) i.e. developng estimates over other estimates. In TD learning, the agent learns to make decisions by interacting with an environment with the help of estimates about the states. It receives feedback in the form of rewards for its actions and updates its knowledge (values) about state-action pairs over time. The agent estimates the expected future rewards based on the observed rewards and

predictions for the next state, and it iteratively updates its value estimates using the temporal difference error. TD learning employs bootstrapping technique to update the value estimates of state-action pairs. When an agent transitions from one state to another and receives a reward, it updates the value of the previous state-action pair by incorporating the observed reward and the predicted value of the next state-action pair. This is refereed to as the temporal difference error. The popular TD learning methods include TD(0), TD (λ), etc.

1.2 Policy Gradient Algorithms

When dealing with large action spaces, such as continuous spaces with an infinite number of actions, policy-based approaches are helpful. The large sction space occurs when there are very large of actions in the environment, for instance in milions in problems such as recommender systems, web search, games etc [31]. In policy-based techniques, the agent directly learns the policy and selects an action from a probability distribution of the action space. Furthermore, unlike value-based methods, policy-based methods can learn stochastic policy, resolving the explore/exploitation conflict automatically. In reinforcement learning, the agents try to learn the optimal policy by taking the actions that maximizes the rewards. However, the traditional Policy Gradient methods suffers from high variance due to gradient estimation and a large state-action space leads to increase in sample complexity.

Let π_θ denote a policy parameterized with θ, and $J(\pi_\theta)$ denote the expected finite-horizon undiscounted return of the policy i.e., the cumulative reward agent receives from the current state to final state with a finite number of timesteps. The objective of Policy Gradient method is to find the optimal choice of θ that maximizes $J(\pi_\theta)$. We have $J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\mathbb{R}(\tau)]$, where τ is a trajectory and $R(\tau)$ denotes the reward over the trajectory.

The gradient of $J(\pi_\theta)$ is obtain from the Policy Gradient algorithm, Chapter 13, Pg.

324 [94] as,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\pi[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}], \tag{1.1}$$

where A^{π_θ} is the advantage function for the current policy, $A_t = Q(s_t, a_t) - V(s_t)$, $\pi_\theta(a_t|s_t)$, represents probability of taking action a in state s at time t. The gradient of policy performance, $\nabla_\theta J(\pi_\theta)$, is called the policy gradient, and algorithms that optimize the policy this way are called policy gradient algorithms. Policy gradient methods in general evaluate advantage function estimates based on the infinite-horizon discounted return, denoting how much better or worse taking action a in state is compared to acting according to the policy. The policy gradient algorithm works by updating policy parameters θ_k, via stochastic gradient ascent on policy performance,

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}).$$

1.2.1 Actor-critic Methods

Policy gradient algorithms optimize the policy directly rather than evaluating value functions for each state action pair. We can formulate the policy gradient objective as :

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\pi[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t], \tag{1.2}$$

where G_t denotes the return of the current trajectory.

We can extend the equation as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s_0, a_0, \ldots, s_t, a_t}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t], \tag{1.3}$$

Using the following equality $\mathbb{E}_{s_0,a_0,....,s_t,a_t}[G_t] = Q(s_t, a_t)$, we can formulate above equation as,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s_0,a_0,....,s_t,a_t}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q(s_t, a_t)] = \mathbb{E}_\pi[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q(s_t, a_t)] \tag{1.4}$$

The Q value can be learned by employing a neural network to parameterize the Q function, from which we derive the Actor-Critic algorithm, where v represents the value function of the state s and γ represents the discount factor. Actor-Critic algorithms are primarily temporal difference (TD) learning methods that represent the policy independent of the value function.

Based on the provided state, a policy function produces a probability distribution across the actions that the agent can take. A value function calculates the expected return for an agent that starts in a given state and follows a specific policy indefinitely.

The actor-critic algorithm calculates the policy gradient, policy, and value function using two distinct modules. The actor-critic framework thus consists of two models:
Actor: The actor module determines what action to take and modifies the actor parameters in the direction implied by the critic.
Critic: The critic network then evaluates whether the action taken was good or not by updating the parameters of value function. The Actor-Critic module is demonstrated in Fig. 1.2.

1.3 Ranking in Search Systems and Recommender Systems

Search and recommender systems play a key role in helping users discover relevant content, products, or information. With the overload of digital data on web, it is essential that users receive the items/documents that are most relevant to their needs.

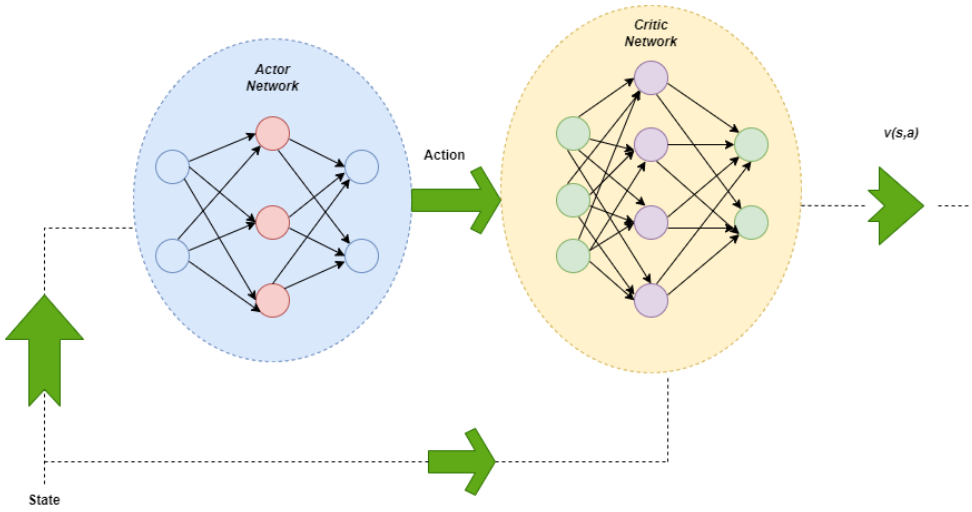

Figure 1.2: Actor Critic framework

Subsequently, not only must the search results or items be relevant, but the most relevant results should be at the top of the list. There is also significant overlap in research work for recommender systems and information retrieval where different techniques for recommender systems have been applied for ranking in information retrieval systems and vice versa [6, 87, 111, 10, 47]. There are many applications that combine both ranking and recommendation functionalities together to provide user a better experience such as in e-commerce sites like amazon, etc. Ranking is the process of arranging a list of items or examples in a specific order based on their relevance, importance, or some other criteria. Ranking is a common task in various machine learning applications, and it is used in fields such as information retrieval, recommendation systems, natural language processing, and more. Ranking in search and recommender systems is the task of determining the order in which items, such as web pages, products, or content, are presented to users based on their relevance to the user's query, preferences, or behavior. This process is crucial for search engines, e-commerce websites, content streaming services, and many other applications where user engagement and satisfaction are important. Search systems are designed to help users find specific information, documents, or resources by entering queries. These systems retrieve and rank results based on the relevance to the user's query. Displaying the most relevant items at the top of the search results,

makes it efficient for users to find the information they are looking for. Search ranking incorporates different algorithms and tools to ensure that the displayed results align with the users intent and preferences. Ranking of queries is a critical feature of search engines and information retrieval systems. When a user enters a query into a search engine, the system must rank relevant documents or items to present the most useful and valuable results at the top.

Subsequently, due to the substantial increase in digital information being accessed via the internet, recommender systems have grown in popularity. By choosing a small number of products from a huge group of options, they assist in providing the user/customer with personalised recommendations. Recommender systems are designed to predict and recommend items that users might be interested in based on their preferences and behavior. The relevance of items in recommender systems is typically determined by a combination of factors, and ranking plays a crucial role in presenting the most relevant items to users. "Top-N" recommendations in recommender systems refer to the practice of presenting a user with a list of the N most relevant items based on their preferences or historical interactions. The ranking of these items is crucial as it determines the order in which they are displayed to the user. Usually, the recommendations are made in reference to various decision-making tasks, such as choosing a product to buy, movies to watch, music selection, online news to read, etc. When a person must select an item from a possible overwhelming selection of things, recommender systems are especially helpful. Recommender systems are ubiquitous nowadays from social media websites to ecommerce product sites with millions of items. Users are given recommendations for products via recommender systems based on information that is readily available, such as past usage trends, user usage habits, and characteristics of the products themselves. The consumer can more easily access the product that best suits his needs with the help of recommendation systems. In the internet age, people search through a wide range of possibilities to find the ones that suit them the best. Typically, recommender systems draw information from a vast database corpus and present the findings to a

user in a certain domain, such as movies books, or products available on the internet.

1.3.1 Recommender systems

Recommender systems provide personalized suggestions to users for items or content such as movies, books, music, products, or other resources, based on the user's preferences, behavior, or past interactions. Recommender systems are widely used in various online platforms to enhance user experience and engagement. The recommender system problem can be mathematically stated as the function $u, i \rightarrow R$, where u represents the user, i represents item and r represents the corresponding rating assigned by the user.

Collaborative filtering and content-based systems are the two general categories of recommender systems. Another category is hybrid recommender systems that combine two or types of recommender system methods. The core idea of the collaborative filtering is that users' past preferences have a big impact on their present and future actions. Generally, the similarity between users is calculated based on their past behaviours. Following that, the target users' interest in the item is predicted based on the evaluation of the neighbours who share a high degree of similarity with the target users. Collaborative filtering methods are further classified as Memory-based or Model-based. Memory-based approaches are generally neighborhood-based methods and use rating data to find the similarity between the users or items [9, 19]. Model-based methods learn a model from user's rating history and use it for making prediction, for example, Singular Value Decomposition (SVD) based models. Item-based and user-based CF recmmendations are two different types of CF recommendations. The item-based CF recommendation looks for an item set that resembles the target project based on how similar the items are to one another. In contrast, user-based CF recommendations base their forecasts and suggestions for choosing a neighbourhood on active user knowledge about the neighbourhood.

The latent factor models are effectively used in collaborative filtering. These methods postulate that a latent user and item vector in a low dimensional space determines the user preferences on the items by discovering the latent factors that influence the users preference, like the price of items, quality. Matrix Factorization is a popular latent factor-based method introduced in Netflix competition which improved the performance of recommendation systems significantly over neighbourhood based [49]. Latent factor based methods are a class of collaborative filtering techniques. In CF user-item interaction matrix stores the ratings given by users for different items, such as movies, songs, etc. It involves decomposing the interaction matrix into the product of two rectangular matrices with smaller dimensions is known as matrix factorization. According to these methods, a latent user and item vector in a low-dimensional space determines the user preferences on the things by identifying the latent elements that affect the user's choice, such as the price and quality of the products. The goal of matrix factorization is to represent users and items in a latent space with fewer dimensions. Many different matrix factorization methods have been put forth for recommender systems since they were proposed such as Funk SVD, Asymmetric SVD, etc.

1.3.2 Search systems in Information retrieval

Search systems in information retrieval involve in retrieving relevant information from a collection of data in response to a user's query. These systems are fundamental in helping users find specific information or resources within large and diverse datasets. They are designed to assist users to discover specific information, documents, or resources by entering queries. These systems retrieve and rank results based on the relevance to the user's query. Ranking is one of the primary task in search systems and involves finding the most relevant document based on the input query. Relevance is typically determined by assessing how well a document or item matches the terms and intent of the query. The primary objective of ranking is to present search results in a way that reflects their relevance to the user's query.

Relevance models are a class of models used in information retrieval to enhance the precision and relevance of search results. They aim to address the challenge of retrieving documents that are more closely aligned with the user's information needs. Various relevance models are employed to assess the relevance of documents to a query. These models may include probabilistic models, language models, and vector space models. The Vector Space Model (VSM) is a mathematical model used in information retrieval and natural language processing to represent documents and queries as vectors in a multi-dimensional space. Documents and queries are represented as vectors in a multi-dimensional space, and the similarity between vectors (using cosine similarity or other metrics) is used to rank documents.

TF-IDF is a common weighting scheme that evaluates the importance of terms in a document relative to their frequency across a collection of documents. It helps identify documents where the query terms are significant. BM25 (Best Matching 25) is a ranking function that builds upon the TF-IDF model. It introduces term saturation and document length normalization to enhance the accuracy of ranking.

Modern search systems often leverage machine learning techniques to improve ranking and are classified as Learning to Rank models. These models learn from historical user interactions and other features to predict the relevance of documents to a specific query.

1.3.3 Learning to Rank and Top-N Recommender systems

- Learning to Rank : Learning to Rank is a machine learning approach used to train models that can rank a list of items according to their relevance or importance for a specific task. The goal of Learning to Rank is to learn a ranking function that takes a query or input and assigns a relevance score to each item, allowing them to be ranked in order of their estimated relevance. LTR is commonly applied in information retrieval, search engines, recommendation systems, and other applications where ranking items is essential to present

the most relevant results to users. Machine learning approaches are used to develop the ranking model over the input space in learning to rank.

Learning to Rank approaches create a ranking model for a variety of documents by minimising the loss function across the training data, which consists of query document pairs and judgement labels. In order to learn the ranking function during the training process, various machine learning-based algorithms have been presented that directly optimise the evaluation metric, such as Precision, NDCG, etc. Optimising the various information retrieval metrics, MAP and NDCG, respectively, support vector machine-based approaches, such as SVMMAP and SVMNDCG have been applied to the task. Ranking models typically predict a relevance score $s = f(x)$ for each input $x = (q, d)$, where q is a query and d is a document. Once we know the relevance of each document, we can sort the documents based on those scores producing ranked list. Learning to Rank approaches are classified as pointwise, pairwise, or listwise. Each approach addresses the task of ranking items (such as documents or recommendations) in response to a query, but they differ in their strategies for learning the ranking function. Pointwise solutions for learning to rank problem are often classification or regression techniques that are limited to predicting a class or relevance label for the input content. For pointwise techniques, any general classification or regression methodology could be used. The pairwise approach considers pairs of instances (query-document pairs) and focuses on learning the relative ordering of pairs while the listwise approach considers the entire ranked list of documents for each query as a single training instance.

- Top-N recommender systems : Top-N recommender systems focus on providing users with a ranked list of the most relevant items based on their preferences or historical interactions. In Top-N recommender systems, ranking is a crucial aspect of providing users with a list of the most relevant items. Their applications can be found across multiple domains ranging from online shopping, e-commerce items, video portals, etc. The goal is to order items

in a way that maximizes user satisfaction by presenting the most likely items that the user will find interesting or useful. There are multiple approaches to provide Top-N recommendation such as collaborative filtering, content based recommender systems, deep learning models based recommendations etc.

1.4 Applications of Recommender systems and Search systems

Recommender systems and search systems have numerous applications across various domains, providing users with personalized content and helping them discover relevant information. They help provide relevant items or data to the user from a very large volume of data. There is some overlap between the applications of search and recommendations, where the modern systems integrate both recommendation and search functionalities to provide a comprehensive user experience. For example, e-commerce platforms may use recommender systems to suggest products and search systems to allow users to find specific items. Here are some key applications for both types of systems:

Recommender Systems applications:

- E-commerce - As the e-commerce business expands, so will the demand for recommendation systems. The recommender systems can be effectively utilized to keep the online catalogue updated as per the user usage preferences. With millions of customers and data on their online behaviour, e-commerce companies are best placed to offer accurate suggestions for a wide range of items.

- Social Network recommender systems - To improve users experience in social network domainsites such as twitter, facebook etc., recommender systems provides users to increase social interactions with other likeminded users such as

in online friending, dating, social media communties and tags etc.

- Media - This includes recommender systems targetted at personalized recommendations for movies, songs etc. The recommender systems can learn a user profile from the ratings provided by the user for movies or songs.

- Streaming Services - Platforms like Netflix, Hulu, Spotify, and YouTube leverage recommender systems to recommend movies, TV shows, music, and videos tailored to users' preferences.

- News Aggregators - News websites and apps use recommender systems to suggest articles and news stories that align with users' preferences and reading habits.

- Travel and Hotel Recommendations - Travel platforms like TripAdvisor and hotel booking websites use recommender systems to suggest destinations, accommodations, and activities based on users' preferences.

  Other applications for recommender systems include news article recommendation, tourist sites, advertisement, banking etc.

Search Systems applications:

- Web Search Engines - Platforms like Google, Bing, and Yahoo provide users with the ability to search and retrieve information from the vast expanse of the World Wide Web.

- Search engine advertising - Search engine advertising is a form of digital advertising that allows businesses to promote their products or services by placing ads within the search engine results pages (SERPs) of popular search engines like Google, Bing, or Yahoo.

- E-commerce Search - Online marketplaces such as Amazon, eBay, and Alibaba use search systems to enable users to find products efficiently and make informed purchase decisions.

- Academic and Job Search - Platforms like Google Scholar or specialized academic databases help researchers, students, and academics discover scholarly articles, papers, and research publications. Similarly, websites like LinkedIn, Indeed, and Glassdoor employ search systems to match job seekers with relevant job postings based on skills, experience, and preferences.

- Multimedia Search - Search engines for images (e.g., Google Images), videos (e.g., YouTube), and audio content enable users to find multimedia content quickly and effectively.

- Enterprise Searchm - Within organizations, search systems help employees find relevant documents, information, and resources in corporate databases, intranets, and document repositories.

  The other applications of it are in question answering task, ocal search engines like Google Maps for finding nearby places, flight itinerary search, etc. As search has played an important role in the client experience, integrating LTR with natural language processing (NLP) techniques to understand a customer's intent and provide relevant results. Learning to Rank for flight itinerary search have been employed that allows users to search for flights and book the best vacation for them.

  Both recommender systems and search systems contribute significantly to improving user experience and information discovery in various domains. They leverage advanced algorithms and technologies to provide users with relevant and personalized results.

1.5 Major Challenges

In this section, we enlist dome key challenges for search and recommender systems :

- User Intent Understanding: Both search and recommender systems need to accurately interpret user intent. Understanding the context behind a user query or request is crucial for providing relevant results or recommendations.
- Data Quality and Sparsity: The quality of the underlying data is critical for both types of systems. Noise, inaccuracies, and sparsity in the data can lead to suboptimal results. Thus, it is required to maintain data quality and address sparsity challenges.
- Dynamic Content and Changes:

  The dynamic nature of online content poses challenges for both systems. Content is continually changing, and staying up-to-date with the latest information is crucial for relevance in both search results and recommendations.
- Scalability:

  As user bases grow and the volume of content increases, scalability becomes a common challenge. Both search and recommender systems must efficiently handle large datasets. Given the rapid increase in the quantity of web pages, searches and large amounts of user and item data, the scalability of large-scale search engines and recommender systems becomes a genuine challenge.
- User Feedback Incorporation:

  Gathering and incorporating user feedback is essential for improving the performance of both search and recommender systems. Designing mechanisms to collect feedback and adapt the system accordingly is a common challenge.
- Adaptability:

  User preferences and behaviors can evolve over time. Both search and recommender systems need to adapt to these changes to continue providing valuable

and relevant content.

- Position bias: Position bias is a common issue in search and recommender systems in which items displayed at the top of a list (e.g., search results, recommended items) are more likely to be clicked or selected than items displayed lower down the list, even if the lower ranked items are a better match for the user's preferences.

- Mutual information: Existing ranking techniques over looks the correlation between the documents and do not utilize shared information between them. By intergerating this mutual information during training, the ranking model can provide more relevant results to the user.

- High variance in existing ML models: The machine learning algorithms that have proposed for ranking task also suffer from variance and noise. In RL, variance refers to the variability or fluctuation in the estimates of quantities such as state-action values or policy gradients during the learning process. The limited number of samples can lead to high variance in the estimates, especially in high-dimensional state and action spaces.

In this thesis, we consider the challenges such as scalability, utilization of shared information, high variance in existing ranking models. We address these challenges and propose models to improve the efficiency and effectiveness of ranking in search systems and recommender systems.

1.6 Motivation

With the multifold increase of data over the internet in recent years, it has become imperative that users should be returned the items/documents that are most relevant to their needs. Today, the usage of search engine and recommender systems is widespread across various domains. They provide the users with relevant search results or items from a large corpus of data. However, the search results or items

have to be not only relevant but also most relevant results should be at the top of the list. Ranking is a primary task in web search and recommendation task that can be used to provide the most relevant results(documents or items) at the top of the search list. Search and Recommender systems are a subclass of information systems providing meaningful suggestions to the users to tackle the information overload over internet.

With the increasing size of item space and users, scalability remains a key issue for search and recommender systems. It deals with the challenge of designing efficient algorithms capable of handling large scale datasets. Scalability is crucial for handling increasing volumes of data and user requests efficiently. Further, the existing ranking methods for search over looks the correlation between the documents and thus do not utilize the shared information between them. If the ranking model is able to incorpoorate more documents similar to the relevant ones based on the shared features during learning, the performance can be improved further.

The machine learning algorithms that have proposed for these tasks also suffer from variance and noise. In RL, variance refers to the variability or fluctuation in the estimates of quantities such as state-action values or policy gradients during the learning process. RL algorithms often rely on sampling to estimate state-action values or gradients. The limited number of samples can lead to high variance in the estimates, especially when dealing with complex scenarios or high-dimensional state and action spaces. Traditional RL algorithms, such as tabular methods or linear function approximation based methods, can struggle to handle large or high-dimensional state and action spaces effectively. As the size of the data space increases, these algorithms suffer from the curse of dimensionality and lack the complexity to generalize well or learn meaningful policies. Recently, reinforcement learning methods have been applied effectively to information retrieval tasks, however traditional RL algorithms suffer from lack of complexity required when the state space becomes extremely large with millions of items. Further, noisy gradients and increase in variance is another in policy gradient RL algorithms with increasing item space. With the advent of

Deep Reinforcement learning framework, it is now possible for adapt RL to complex problems with very large action space. The issue of intractability with traditional RL approaches can be resolved with the neural network approximation.

1.7 Contribution

This thesis focuses on application of policy gradient based methods for ranking in search systems and recommendation systems. This work contributes algorithms, frameworks and provides analysis and insights into to different issues that arise in the above mentions tasks. We enlist the major contribution of our thesis in various parts as follows:

- In Chapter 3, we describe our first work "A deep actor critic reinforcement learning framework for learning to rank". Here, we propose a deep reinforcement learning based approach for ranking in web search. By combining Deep learning with the Reinforcement Learning framework, our method can learn a complex function as deep neural networks can provide significant function approximation. To the best of our knowledge, this is the first Deep RL approach applied in Learning to Rank for document retrieval. We performed experiments on the various Letor datasets for different evaulation metrics and showed that our method outperforms various state-of-the-art baselines.

- In chapter 4, we present our second work "A Multi Agent Deep Reinforcement Learning based approach for large scale Learning to Rank". We propose a Multi-Agent Deep Reinforcement Learning based framework for large-scale ranking in search. The existing Learning to Rank approaches also overlooks the correlation between the documents by ignoring the shared information between different documents. We use the Multi-Agent framework to capture the correlation between the documents by sharing the information between different agents, with the centralized critic framework having access to this global information. We performed experiments on the two large scale Microsoft

Letor datasets and showed that our method outperforms various state-of-the-art baselines.

- In chapter 5, we describe "Proximal policy optimization based hybrid recommender systems for large scale recommendations". We propose a Policy gradient-based hybrid recommender system for large scale recommendations providing the list of top items to the user, utilizing the state-of-the-art PPO algorithm to train our agent. We use the Proximal Policy Optimization approach to train our agent and modeled recommender system as a Markov Decision Process. The PPO based algorithm mitigates the high variance and leads to increase in stability. We demonstrate the effectiveness of the proposed framework on the two popular Movielens datasets : ML-1m and ML-100k, for different evaluation metrics.