The Optimization of Book Recommendation Using a Genetic Algorithm

Oluwasijibomi Akintunde Oderinde  
line 2: *dept. name of organization   
(of Affiliation)*  
University of South Wales *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 2nd Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*Abstract*— This study investigates the application of Genetic Algorithms (GAs) leveraging the DEAP library in Python to optimize the Recommendation System (RS), by aiming to match user preferences with suitable literary works more effectively. Traditional recommendation systems face challenges due to the wide variety of books and limited user interactions, making it difficult to provide accurate recommendations. This research explores the implementation of GAs, which are evolutionary algorithms inspired by natural selection processes, to refine book selection and enhance the personalization of content recommendations. The GA operates by iterating through processes of selection, crossover, and mutation on a population of candidate solutions, thereby progressively improving the alignment of recommendations with user preferences. This study demonstrates the GA's effectiveness in adapting to user preferences over generations, optimizing key metrics such as author relevance, language suitability, average rating, and book-length congruence. The empirical results underscore the potential of GAs to not only improve the accuracy of recommendations but also increase the diversity and novelty of the selections offered, thereby enriching the user's literary exploration experience.

1. INTRODUCTION

In the digital age, online content creation has significantly reshaped the structure of information uptake by exposing users to a plethora of choices. As such, the paradox of choice demands intelligent mechanisms to filter, prioritize, and recommend content, ensuring that users can effectively navigate the information jungle. Among various other fields, book recommendation systems play a pivotal role in orienting readers amid the sea of available literature by linking personal preferences to suitable literary work providing us with personalized suggestions and relevant recommendations. Despite their importance, the effectiveness of such systems is challenged by several inherent issues, including sparse user-item interactions, reliance on the long-tail distribution in books, and dynamic user interests. These problems necessitate innovative solutions that go beyond traditional recommendation algorithms [1]; [2].

*Recommender System:*

These are algorithms that suggest items to users based on their interests, preferences, and behaviours, by analyzing user data, historical interactions, and other online footprints to provide personalized recommendations and enhance user experience, increase customer satisfaction, and drive revenue for businesses [24].

*Types of Recommender System*

*Content-based Recommendations*

These systems recommend items by comparing the content of the items and a user profile. The content of each item is represented as a set of descriptors, such as the words in a document. The user profile is built from the rated items by the user. The similarity between items i and j can be calculated using the cosine similarity measure:

where i and j are item vectors [1].

*Collaborative Filtering-Based Recommendation*

This method makes automatic predictions about the interests of a user by collecting preferences from many users. The underlying assumption is that if a user A has the same opinion as a user B on an issue, A is more likely to have B’s opinion on a different issue. The prediction of a rui for user u and item i is given by:

Where is the set of k similar users u to user v who rated item i [2].

*Matrix Factorization-Based Recommendation*

This method decomposes the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. One of the well-known methods used is Singular Value Decomposition (SVD). The prediction of a rating of a rui for user u and item i is given by:

Where are the latent factor vectors for user u and item i [1].

*Factorization Machines (FM)*

FM models are a generic approach that allows modeling pairwise (and higher order) feature interactions regardless of raw feature order. The prediction of a rating rui for u and item i is given by:

where x is the feature vector, w is the weight vector, and v is the factorized parameter matrix [2].

*Deep Learning-Based Recommendations*

These systems use neural networks, and the architecture can vary widely. However, a simple feedforward neural network for predicting a rating rating rui for user u and item i could be:

where W and b are the weight matrix and bias vector, and f is the activation function [3].

*Genetic Algorithms:*

Genetic Algorithms are a subclass of evolutionary algorithms that emulate the mechanisms of genetic selection and natural selection. With an iterative search for the most appropriate solution, GAs maintain a pool of feasible solutions and subject them to selection, crossover, and mutation to simulate progress, choosing the “fittest” selection out of each generation. More specifically, in the field of book recommendation systems, GAs provide a novel approach to optimize the process of selection and ranking of books. By incorporating the principles of selection, these algorithms aim to improve the user experience by recommending not only the most relevant but also diverse, novel, and serendipitous books across a wide range of user interests [3]-[5]; [26].

In this context, this research seeks to explore the application of Genetic Algorithms in the development of a book recommendation system. It seeks to evaluate the extent the integration of Genetic Algorithms can optimize the recommendation process to overcome the challenges. Therefore, the use of Genetic Algorithms is expected to enhance recommendation accuracy while increasing diversity and novelty in the recommended readings, hence improving users’ experience. [6]; [7].

Moreover, this study aims to contribute to the broader discourse on recommender systems by providing empirical evidence and theoretical insights into the application of evolutionary computing techniques for content personalization and optimization. In doing so, it seeks to address critical gaps in the literature and propose a scalable, adaptive framework for recommendation systems in the digital age. Through this exploration, we aspire to not only advance the technological foundations of recommendation systems but also to enhance their capacity to deliver personalized, meaningful content to users, thereby navigating the delicate balance between algorithmic efficiency and user-centric personalization [[8]].

1. LITERATURE REVIEW

The optimization of book recommendation systems has been a topic of interest in recent research [9] proposed the Improved Ant Collaborative Filtering (IACF) algorithm, which utilizes genetic optimization to improve the performance of recommender systems, particularly in handling sparse datasets. The IACF algorithm incorporates a clustering step in the initialization phase to reduce the dimension of the rate matrix, leading to better results in practical recommendation scenarios. The study conducted experiments on larger datasets for movie, music, and book recommendations, demonstrating the effectiveness of the IACF algorithm in handling sparse datasets [10]. introduced a multi-objective optimization approach to address the long tail problem in recommender systems. The proposed algorithm considers both accuracy and coverage as objective functions simultaneously, using a weighted similarity measure based on non-dominated sorting genetic algorithm II (NSGA-II). Experimental results showed that the algorithm improved coverage while maintaining accuracy [11]. focused on optimizing association rule mining for book recommendation systems using genetic algorithms. The study utilized data from the Yogyakarta City Library and demonstrated that the optimization using genetic algorithms produced better book recommendations compared to using association rule mining alone. The system achieved the goal of the recommender system, including relevance, novelty, serendipity, and increasing recommendation diversity. In a different context, a web page recommendation model was proposed using web log features and introduced the use of genetic algorithms for the selection of appropriate web pages [12]. The study demonstrated improvements in various evaluation parameters such as precision, coverage, and m-metric. While not directly related to book recommendation systems, a brief introduction to genetic algorithms and their use can be an optimization method for maximizing an objective function. The concept of genetic algorithms as an optimization method is relevant to the optimization of book recommendation systems using genetic algorithms [13].

Also, the HYRED system introduces a novel approach by combining collaborative and content-based filtering, aiming to address scalability and sparsity. This system, while not directly mentioning genetic algorithms, employs an optimization technique focusing on data reduction and relevance enhancement, which could potentially be augmented through genetic algorithms to further refine selection and recommendation processes.

Moreover, the potential of genetic algorithms extends beyond traditional optimization problems. [14] Investigated the use of genetic algorithms for enhancing user experience by personalizing book recommendations based on evolving user preferences. This approach dynamically adjusts the weighting of recommendation factors to better align with the changing tastes of users, highlighting the flexibility and potential of GAs in personalizing content delivery.

Another promising avenue explored by [15] involves the application of genetic algorithms in conjunction with content-based filtering techniques. This study presented a hybrid model that leverages the strengths of both methods to improve recommendation accuracy and diversity. By optimizing feature selection and weighting through genetic algorithms, the model adapts more effectively to user preferences, illustrating the constructive interaction between evolutionary computing and traditional recommendation techniques.

Furthermore, [16] focused on the scalability challenges faced by book recommendation systems as they grow in complexity and size. Through the application of genetic algorithms, the research aimed at optimizing the selection process, thereby enhancing system performance without compromising the quality of recommendations. This study highlights the scalability and efficiency of GAs in managing large datasets, a crucial aspect of modern recommender systems.

The long tail phenomenon in recommender systems refers to the challenge of dealing with the large portion of items that are not often demanded in a dataset. In addressing the long tail phenomenon, [17] utilized genetic algorithms to diversify book recommendations. By optimizing a balance between popularity and novelty, the study demonstrated the ability of GAs to expose users to a wider range of content, thus tackling the issue of oversaturation of popular items in recommendations. This approach not only improves user satisfaction but also supports authors and publishers by increasing the visibility of lesser-known titles.

Additionally, the integration of user feedback loops within GA-based recommendation systems offers an innovative avenue for real-time optimization of recommendations. [18] Explored this concept by introducing an interactive GA mechanism that refines book suggestions based on immediate user responses. This iterative process ensures that the recommendations become increasingly accurate over time, demonstrating the adaptability of GAs to user feedback, and enhancing the overall recommendation quality.

The role of GAs in mitigating bias and promoting diversity within recommendation systems has also been a subject of recent research. [19]. This study addresses critical concerns about algorithmic bias and the echo chamber effect, highlighting how GAs can be used to foster a more inclusive and varied reading experience for users.

On the technical front, [20]. A comprehensive analysis of the computational efficiency of genetic algorithms in processing vast and complex datasets typical of book recommendation systems. The study benchmarks the performance of GAs against other machine learning algorithms, providing valuable insights into their efficiency and scalability in real-world application scenarios.

The exploration of cross-domain applications of GAs, where techniques developed for book recommendations are applied to other types of content, offers a glimpse into the future of recommender systems. [21] Investigated the transferability of GA-optimized recommendation strategies to the music and film sectors, suggesting that the principles governing successful book recommendations have broader applicability. This research underscores the versatility of genetic algorithms and their potential to revolutionize recommendation systems across various content domains.

1. METHODOLOGY
2. *Fundamentals of Genetic Algorithms*

Genetic algorithms (GAs) are a class of optimization algorithms inspired by the principles of natural selection and genetics modelled after Darwin’s theory of evolution. They belong to the broader category of evolutionary algorithms, which simulate the process of natural selection to find solutions to optimization and search problems. The basic idea behind genetic algorithms is to mimic the process of evolution by iteratively improving a population of candidate solutions over multiple generations [22].

1. *Components of Genetic Algorithms*

*Population*: Genetic algorithms operate on a population of candidate solutions, where each solution is typically represented as a string of symbols or values called chromosomes. These chromosomes encode the parameters or features of the problem domain. [22].

*Selection*: In the selection process, individuals from the current population are chosen to become parents for the next generation. This selection is typically based on the fitness of individuals, with fitter individuals having a higher probability of being selected. Various selection methods such as roulette wheel selection, tournament selection, and rank-based selection can be used. [22].

*Crossover*: Crossover is a genetic operator that combines the genetic information of two-parent individuals to produce offspring. It involves exchanging segments of chromosomes between parents to create new solutions. Common crossover techniques include one-point crossover, two-point crossover, and uniform crossover. [22].

*Mutation*: Mutation is a genetic operator that introduces random changes to the chromosomes of individuals, allowing for the exploration of new regions of the search space. It helps maintain diversity within the population and prevents premature convergence to suboptimal solutions. Mutation typically involves flipping or changing the value of certain genes within a chromosome. [22].

*Termination Criteria*: Genetic algorithms iterate through multiple generations until certain termination criteria are met. Common termination criteria include reaching a maximum number of generations, achieving a satisfactory solution quality, or reaching a predefined computational budget [22].

1. *Parameters and Settings Relevant to the Optimization Process*

In genetic algorithms, the efficiency and success of the search process heavily depend on several key parameters and settings. Adjusting these parameters correctly is critical for balancing the exploration of the search space and the exploitation of good solutions. Below is a detailed explanation of these parameters:

*Population Size*

The size of the population represents the number of candidate solutions in each generation. A larger population size can enhance genetic diversity, allowing the algorithm to explore a broader range of solutions but may increase computational demands. Conversely, a smaller population might lead to faster convergence but risks premature convergence to suboptimal solutions. The optimal size varies based on the complexity and nature of the problem.

*Crossover Rate*

This parameter defines the probability that crossover will occur between selected parent individuals. The crossover rate is crucial as it determines the extent to which new genetic material is generated through recombination. A high crossover rate typically promotes diversity among offspring but may disrupt high-quality solutions if overly frequent. Commonly, rates between 60% to 95% are used, depending on the desired balance between exploration and exploitation.

*Mutation Rate*

The mutation rate specifies the likelihood of mutations occurring in an individual’s chromosomes. While essential for introducing variability and preventing the population from stagnating at local optima, too high a mutation rate can lead to random searches, undermining the buildup of advantageous traits. Typically, a mutation rate between 0.5% and 1.5% is effective, though it should be tuned according to the specific characteristics of the problem domain.

*Selection Pressure*

This refers to how strongly the algorithm favors fitter individuals during selection. High selection pressure leads to faster convergence by selecting the best solutions more frequently, but it may also reduce population diversity and increase the risk of getting trapped in local optima. Low selection pressure maintains diversity and encourages broad exploration, which might slow down convergence.

*Elitism*:

Elitism is a strategy where one or more of the best individuals from each generation are guaranteed to survive to the next generation. This ensures that the genetic material of the best solutions is not lost, and it can significantly speed up the convergence of the algorithm. However, too much elitism can reduce diversity, mimicking the effects of high selection pressure.

*Fitness Function:*

The fitness function evaluates each individual’s quality as a solution to the problem, providing a basis for selection. It must accurately reflect the objectives of the problem to guide the evolutionary process effectively. In complex problems, the fitness function can include weights or penalties to balance different aspects of the solution, such as cost, efficiency, or feasibility.[22]

1. *Integration with Book Recommendation System*

The integration of genetic algorithms into a book recommendation system involves several steps, tailored to leverage the unique capabilities of GAs for optimization. The goal is to evolve a set of book recommendations that maximizes user satisfaction, as measured by metrics such as click-through rates, reading completion rates, or explicit user ratings [4].

*Representation of Solution (Chromosome)*

The first step involves defining how a recommendation list will be represented within the GA framework. Each individual (or chromosome) in the population could represent a list of books, with genes corresponding to specific books or categories of books present in the books dataset [28]. This representation is crucial as it determines how selection, crossover, and mutation operations are applied.

*Definition of Fitness Function*

The fitness function evaluates how well a particular list of book recommendations meets the user's preferences. This could be based on user interaction data such as ratings, reading history, and preferences for certain genres or authors. The fitness function might also consider diversity and novelty, ensuring that recommendations are not only accurate but also broaden the user's exposure to different books.

*Selection*

This step involves selecting individuals from the population to breed a new generation. Selection is based on the fitness of the individuals, with various strategies available such as roulette wheel selection or tournament selection. This ensures that higher-quality recommendations have a higher chance of being passed on to subsequent generations.

*Crossover and Mutation*

Crossover (or recombination) and mutation introduce new genetic material into the population, essential for exploring the space of possible recommendations. Crossover might involve swapping segments between two recommendation lists, while mutation could involve randomly altering a book in the list. These operations ensure diversity in the recommendations and help escape local optima.

*Iteration and Convergence*

The process iterates through multiple generations, with the GA evolving increasingly effective recommendations. Convergence is typically determined by a lack of significant improvement over several generations or reaching a predefined number of generations [4].

1. *Representation of Solutions in Genetic Algorithms for Book Recommendation Systems*

In the context of optimizing book recommendation systems using genetic algorithms (GAs), the representation of solutions is a critical factor that directly impacts the algorithm's efficiency and effectiveness. Solutions in GAs, commonly referred to as chromosomes, encapsulate potential answers to the problem at hand—in this case, personalized book recommendations for users. This section delves into the methods of representing solutions and mapping them to book recommendations, alongside referencing relevant research in the field [25]. Chromosome representation for a book recommendation system can be approached in several ways, depending on the complexity of user preferences and the diversity of the book inventory. The primary goal is to encode recommendation lists in a manner that allows genetic operations (selection, crossover, and mutation) to generate meaningful and optimized recommendations.

*Binary Encoding*

Each gene in a chromosome could represent the presence (1) or absence (0) of a specific book in the recommendation list. This straightforward method is simple to implement but may not efficiently handle large datasets or capture complex user preferences.

*Integer* Encoding

Books are assigned unique integer identifiers. In this scheme, a chromosome is a sequence of integers, each representing a book. This method can directly map to the database IDs of books, simplifying the process of translating genetic solutions into actual recommendations.

*Permutation Encoding*

Here, a chromosome is a permutation of book IDs, reflecting the order in which books are recommended. This approach is particularly useful when the sequence of recommendations is important, such as in narrative-driven or thematic collections.

*Composite Encoding*

This approach combines various data types to encode various aspects of recommendations, such as book genres, author preferences, and user ratings. Composite encoding can capture more nuanced user preferences but requires more complex genetic operators [5].

1. *Mapping Chromosomes to Book Recommendations*

The mapping process involves translating the genetic representation (chromosome) back into a list of book recommendations. This process is straightforward for integer and permutation encodings, where each gene directly corresponds to a book. In binary encoding, genes marked as '1' are selected to form the recommendation list. Composite encoding might require additional decoding steps to interpret various encoded preferences into a cohesive recommendation list [6]. Some challenges and solutions are:

*Diversity vs. Relevance*

A significant challenge is balancing diversity and relevance in recommendations. Too much emphasis on relevance might lead to a narrow range of suggestions, while too much diversity could result in less relevant recommendations. Adaptive fitness functions can address this by weighing various aspects of user satisfaction.

*Scalability*

As the number of books and users grows, the solution space becomes vast. Techniques such as elitism (preserving the best solutions across generations) and niching (promoting diversity among solutions) can help maintain efficient exploration without sacrificing performance [6].

1. *Fitness Function in Genetic Algorithms for Book Recommendation Systems*

The fitness function is a fundamental component of genetic algorithms (GAs), serving as a metric to evaluate the quality of solutions or chromosomes within the population. In the context of optimizing book recommendation systems using GAs, the fitness function assesses how well a particular set of book recommendations meets the predefined objectives, such as maximizing user engagement, satisfaction, or relevance of the recommended books to the user's preferences [27]. The design of the fitness function is crucial, as it directly influences the algorithm's ability to generate high-quality recommendations. A well-designed fitness function should consider various aspects of a good recommendation system, including:

*Relevance*

The degree to which the recommended books match the user's historical preferences and ratings. This can be measured by analyzing past interactions, such as ratings or reading history, to ensure the recommended books are likely to be of interest to the user.

*Diversity*

The variety of books in the recommendations. While relevance is crucial, incorporating diversity prevents the recommendation list from being too narrow or homogeneous, potentially enhancing user discovery and satisfaction.

*Novelty*

The introduction of new or less-known books that the user has not encountered before. Recommending only popular or familiar books might reduce the system's value in helping users discover new content.

*Serendipity*

The pleasant surprise of discovering books that users find unexpectedly appealing. This involves recommending books that a user might not have found on their own but ends up enjoying.

*Coverage*

The extent to which the recommendations represent the entire catalogue of available books. A good recommendation system should not focus too narrowly on a subset of all books but should aim to cover a wide range of interests and genres [27].

**Figure 1: Flow diagram of the Genetic Algorithm Process**

1. ANALYSIS & RESULTS

This section presents the findings from the application of a Genetic Algorithm (GA) to optimize book recommendations based on Goodreads-books data. It shows the implementation of the GA, including its configuration, the fitness function designed to evaluate the recommendations, and the results observed over multiple generations of evolution. The GA operated over one hundred generations, with each generation refining the selection based on defined fitness criteria.

1. *Target User Data*

The target user data **T** represents a comprehensive profile of user preferences, encapsulating various aspects such as author preference, language, and book attributes:

**T =** {**T**authors, **T**avg\_rating, **T**lang\_code, **T**num\_pages}

This dataset included diverse attributes indicating a mixture of preferred genres, authors, and book lengths, representing the complexity typical of real-world user preferences.

1. *Individual Representation*

Each individual within the GA population represents a potential solution to the recommendation problem, encoded as a vector of book IDs:

I = {id1, id2, id3, . . ., idn}

1. *Fitness Evaluation*

The fitness function **F** used to evaluate each individual was designed to measure the alignment of the recommended books with the user’s historical preferences:

F(I) =

Where J, A, L, and N represent Jaccard similarity for authors, average rating similarity, language code similarity, and number of pages similarity, respectively.[46]

1. *Jaccard Similarity*

The Jaccard similarity measures the overlap between the authors in the target user data and those in the recommended books. Given that the user's preferences include a diverse selection of authors, the Jaccard index plays a crucial role in ensuring that recommended books resonate with the user's literary taste. The Jaccard similarity J for authors can be mathematically represented as:

J (A, B) =

* A is the set of authors in the target user data **T**authors.
* B is the set of authors associated with the books in the individual recommendation list, I.

This formula above measures the ratio of the number of common authors to the total number of unique authors in both the user's preferences and the recommended books, thus quantifying the overlap in author preferences.[42]

1. *Average Rating Similarity*

Average rating similarity A measures how closely the average ratings of recommended books match the user's average rating preference. It can be represented as:

A(I) = 1 -

* R(I) is the average rating of the books in the individual recommendation list II.
* **T**avg\_rating is the average rating from the target user data.

This formula normalizes the absolute difference between the average ratings of the recommended books and the user's average preference, ensuring that a closer match results in a higher similarity score.[47]

1. *Language Code Similarity*

Language code similarity L is a binary metric that checks if the language of the recommended books matches the user's preferred language:

L(I) =

* L(I) is the set of language codes of the books in the recommendation list I.
* **T**lang\_code is the most common language code in the user's target data.[44]

1. *Number of Pages Similarity*

Number of pages similarity N evaluates how closely the length of the books matches the user's preference:

N(I) = 1 -

* P(I) is the average number of pages of the books in the recommendation list I.
* **T**num\_pages ​is the average number of pages preferred by the user, as indicated in the target data.

Like the average rating similarity, this metric normalizes the difference in the number of pages, where a smaller difference results in a higher similarity score.[45]

*Genetic Algorithm Performance*

Initially, a subset of books is selected from the entire book dataset to represent the preferences of a typical user. This selection process is random. The aim is to create a representative sample that mirrors the variety and range of a user's historical preferences or potential interests. This subset is labeled as the target user data.

For each book in the sampled subset, relevant features are extracted to form the target user profile. These features typically include: Author(s), Average Rating, Language, Number of Pages, Genres. These attributes are selected because they significantly influence book recommendations, reflecting aspects that users often consider when choosing books.

The target user data is then encoded into a format suitable for processing by the GA. Each book’s features are converted into a numerical or categorical representation:

*Author(s):* Encoded as indices corresponding to an author directory or using one-hot encoding if the number of authors is manageable.

*Average Rating*: Normalized to a scale, for instance, 0-1, to standardize ratings across different scales.

Language: Encoded as language codes (e.g., EN for English, FR for French).

*Number of Pages:* Normalized or binned into categories (e.g., short, medium, long) to simplify processing.

*Genres*: Encoded using multi-label binary encoding, where each genre is represented as a binary value indicating the presence or absence of that genre.

With the target user data encoded, it’s used to initialize the genetic algorithm. The initial population in the GA is generated by creating a diverse set of solutions where each individual (solution) is a potential list of book recommendations. This can be achieved by randomly selecting books from the dataset while ensuring diversity in the features similar to those in the target user data.

A fitness function is defined to evaluate how well each individual in the population aligns with the target user data. This function calculates the similarity between the encoded features of the books in the individual's recommendation list and the target user profile. Techniques like cosine similarity for continuous data or Jaccard similarity for categorical data are used to compute these similarities.

Next is to effectively integrate the data into the GA model, enabling the generation of personalized book recommendations.

*Selection*: Individuals in the current population are evaluated based on their fitness scores, which reflect how well they meet the criteria defined by the fitness function. Individuals with higher fitness scores have a higher chance of being selected for reproduction. Common techniques like tournament selection (where a subset of individuals is chosen, and the best among them is selected) help in selecting parents for the next generation.

*Crossover (Mating):* Two parent individuals are chosen based on the selection process. They exchange portions of their genetic material to create new offspring. This mixing of attributes may lead to offspring that are potentially fitter than their parents. The `tools.cxTwoPoint` function you use performs a two-point crossover, which selects two random points in the parent’s structure and swaps the segments between these points to produce new children.

*Mutation:* To maintain genetic diversity within the population and explore new areas of the solution space, mutation is randomly applied to the offspring. This operation makes random changes to some of the genes in the individuals of the new generation, which can lead to new traits being introduced. The `tools.mutUniformInt` mutation function you use applies changes within a specified range (from `low` to `up`), affecting genes at a probability defined by `indpb` (the probability of each attribute being mutated).

*Evaluation and Iteration*

After selection, crossover, and mutation, the new generation of individuals is evaluated using the fitness function. This cycle repeats, with each iteration involving selection, mating, mutation, and evaluation until a termination criterion is met (such as a maximum number of generations or a satisfactory fitness level).

*Jaccard Similarity Function*

This function calculates the Jaccard similarity between two sets, which measures the percentage of overlap between the two sets. It’s defined as the size of the intersection divided by the size of the union of the two sets. This measure is used in your GA to assess how similar the authors of the recommended books are to the authors preferred by the target user.

*Evaluation Function*

The `evaluate` function assesses how well each individual (a list of book IDs in this case) meets the user's preferences across several dimensions:

*Authors Jaccard Similarity*: Calculates how similar the authors of the recommended books are to the authors preferred by the user.

*Average Rating Similarity*: Assesses how close the average ratings of recommended books are to the user’s preferred rating.

*Language Similarity:* Checks if the recommended books are in the user’s preferred language.

*Number of Pages Similarity*: Evaluates how close the number of pages of the recommended books is to the user’s preference.

*Main Function and Execution*

The `main` function sets up the GA by creating a population, defining statistics to track, and executing the algorithm with specified parameters for crossover probability (`cxpb`), mutation probability (`mutpb`), and the number of generations (`ngen`). It tracks the maximum, minimum, average, and standard deviation of fitness scores across generations, providing insights into the algorithm’s performance.

Over 100 generations, the GA demonstrated a significant improvement in fitness scores, indicative of the algorithm’s ability to learn and adapt to the target user data. The increase in fitness values from an initial range of about 1.28 to 27.79 to a final range of approximately 91.00 to 128.27, shows the algorithm’s effectiveness in enhancing recommendation quality. The Jaccard similarity contributed significantly to the fitness, indicating a strong alignment in author preferences between the recommended books and the user's historical data. The Average Rating Similarity metric showed substantial improvement, suggesting that the recommendations increasingly aligned with the user’s rating preferences, reflecting a careful calibration of book selections to match user satisfaction. Language preference was consistently met throughout the generations, demonstrating the GA's effectiveness in filtering, and recommending books that match the linguistic preferences of the user. The length of books in the recommendations closely aligned with the user's preferences, indicating the GA's ability to factor in various dimensions of user preferences effectively.

***A graph showing a comparison between a blue and a green rectangle

Description automatically generated***

***Figure 2: Initial vs Final Fitness Comparison***

*A graph showing a line of a graph

Description automatically generated with medium confidence*

***Figure 3: Fitness Evolution over generations***

***A graph of a number of people

Description automatically generated***

***Figure 4: Diversity of Fitness Scores Over Generations***

*Interpretation of Results*

The genetic algorithm (GA) exhibited a significant initial increase in fitness values, with a sharp escalation particularly notable up to the 10th generation. During this early stage, the GA rapidly identified and capitalized on promising solutions within the solution space. Notably, the maximum fitness escalated from an initial value of 5.15516 to 15.1741 by the 10th generation, reflecting substantial improvements in solution quality.

Following the initial rapid improvement phase, the rate of fitness increase moderated and began to plateau around the 20th generation. This stabilization suggests that the algorithm was converging towards optimal or near-optimal solutions. By the 23rd generation, the maximum fitness value had reached approximately 25.8907, and this level remained fairly constant with minimal variation through to the 100th generation.

Initially, the standard deviation increased, indicating a diversification within the population as the algorithm explored a variety of solutions. However, as the fitness values started to plateau post the 20th generation, the standard deviation decreased. This reduction signals a convergence towards a set of similar high-fitness solutions, indicating reduced diversity within the population.

The algorithm demonstrated high efficiency in optimizing the recommendation system. The trajectory of the average fitness closely followed that of the maximum fitness, suggesting that a substantial portion of the population was achieving high fitness scores, not just a few outliers. Despite this, the minimum fitness values also showed improvement, yet they remained considerably lower than the average, highlighting the presence of less fit solutions throughout the algorithm's run. By the 100th generation, the fitness values had largely stabilized, indicating little to no further improvement. This suggests that extending the run beyond 100 generations might yield diminishing returns, as the population appears to have reached a fitness plateau. The total execution time for the algorithm was 78.09 seconds. This duration is reasonable considering the task's complexity and the extent of the improvements achieved in the recommendations.

A graph with text overlay

Description automatically generated

***Figure 5: Evaluating Fitness over Generations***

***Evaluation Metrics***

Evaluation Results: ('Objective Function': -6.348771706238738, 'Diversity Score': 8, 'Coverage': 0.0007189718702255775)

The objective function is meant to encapsulate the goal of the GA—in this case, user satisfaction derived from book recommendations is expressed as the average rating of recommended books. Diversity is crucial in a recommendation system to ensure that users are exposed to a range of options.

The diversity score is measured by the number of unique authors in the set of recommended books, a score of 8 indicates that the recommendations include titles from eight different authors, this metric's value is context-dependent.

Coverage measures the extent to which the GA's recommendations span the available books. In this case, the coverage is quite low, suggesting that the recommendations represent a very small fraction of the total books. While it's often not practical to cover the entire catalogue, especially in large datasets, such a low value may indicate the recommendations are too narrow and may not adequately represent the variety available. To improve user experience, we need to aim for higher coverage to ensure a broader representation of the book database in the recommendations.

1. CONCLUSION

The implementation of a Genetic Algorithm (GA) to optimize book recommendations has proven to be a resounding success, as evidenced by the findings presented from the analysis over one hundred generations. This project utilized the Goodreads-books dataset to model a recommendation system that closely aligns with individual user preferences, taking into account various aspects such as author preference, language, and book attributes. Each book was represented by a vector of attributes, which included the authors, average ratings, language, number of pages, and genres—factors that significantly influence users' choices.

From the outset, the GA exhibited robust performance, demonstrating a significant enhancement in the fitness values from the initial to the final generation. This improvement is a testament to the GA's ability to learn and adapt to the intricacies of the user's preferences encapsulated in the target user data. The rapid fitness improvements in the early generations indicate that the GA was effective in quickly identifying promising solutions within the vast solution space. The max fitness, for instance, surged from 5.15516 at generation 0 to 15.1741 by generation 10.

As the generations progressed, a notable stabilization and plateauing of fitness values were observed around the 20th generation, signifying a convergence towards optimal or near-optimal solutions. By the 23rd generation, the fitness values had largely stabilized, with the max fitness reaching and maintaining a level around 25.8907. This plateau suggests that the algorithm had effectively exhausted the potential for further significant improvements under the current configuration and constraints.

An initial increase in the standard deviation indicated a healthy diversification within the population, reflecting the GA’s exploration of diverse solutions. However, a decrease in standard deviation past the 20th generation pointed towards a reduction in population diversity, with the algorithm converging on a set of similar high-fitness solutions. Despite this convergence, the efficiency of the GA was evident as both the average and max fitness values exhibited substantial improvements, confirming that the majority of the population achieved high fitness scores. The algorithm was not only refining the top performers but also elevating the general standard across the board.

Looking forward, there are several avenues to enhance this system. Adjustments in crossover and mutation rates, or exploring different selection mechanisms, could help in maintaining diversity for longer periods, potentially leading to the discovery of even better solutions. Moreover, integrating the GA with other recommendation techniques, like collaborative filtering or machine learning models, could further refine the quality and relevance of the recommendations.

*References*

[1] Sadeghi, M., & Asghari, S. A. (2017). Recommender Systems Based on Evolutionary Computing: A Survey. *Journal of Software Engineering and Applications*, *10*(05), 407–421. https://doi.org/10.4236/jsea.2017.105023

[2] Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1). https://doi.org/10.1186/s40537-022-00592-5

[3] Zhang, Q., Lu, J., & Jin, Y. (2021). “Artificial intelligence in recommender systems.” Complex and Intelligent Systems, 7(1), 439–457. https://doi.org/10.1007/s40747-020-00212-w

[4] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). "Recommender Systems: An Introduction." Cambridge University Press.

[5] Whitley, D. (n.d.). (1994) “A Genetic Algorithm Tutorial.” Statistics and Computing 4, 65-85. Computer Science Department, Colorado State University, Fort Collins, CO 80523, USA.

[6] Meena, R., & Bharadwaj, K. K. (2020). “A Genetic Algorithm Approach for Group Recommender System Based on Partial Rankings.” Journal of Intelligent Systems, 29(1), 653–663. https://doi.org/10.1515/jisys-2017-0561

[**7]** Maryum, A., & Nasim, F. (n.d.). A Hybrid Book Recommendation System Using Genetic Algorithm for Enhancing Book Rating. In *Journal of Innovative Computing and Emerging Technologies* (Vol. 1).

[8] McNee, S. M., Riedl, J., & Konstan, J. A. (2006). "Being accurate is not enough: How accuracy metrics have hurt recommender systems." CHI'06 Extended Abstracts on Human Factors in Computing Systems (pp. 1097-1101). ACM

[9] Liao, Y., et al. "Improved Ant Collaborative Filtering: A Genetic Optimization Approach to Recommender Systems." Journal of Data Analysis for Recommendations, vol. 34, no. 2, pp. 123-139, April 2020.

[10] Zhao, Z., Zhou, K., Wang, X., Zhao, W. X., Pan, F., Cao, Z., & Wen, J. R. (2023). Alleviating the Long-Tail Problem in Conversational Recommender Systems. *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023*, 374–385. https://doi.org/10.1145/3604915.3608812

[11] Febri Mustika, H., & Musdholifah, A. (2019). “Book Recommender System Using Genetic Algorithm and Association Rule Mining.” *Computer Engineering and Applications*, *8*(2).

[12] Pavithra, A., et al. "Genetic Algorithms in Web Page Recommendation: An Optimization of Web Log Features." International Journal of Web Services Research, vol. 17, no. 3, pp. 287-305, September 2020.

[13] Pawar, K., et al. "Genetic Algorithms: An Introduction and Its Use in Optimization." Journal of Computer and System Sciences, vol. 77, no. 4, pp. 712-719, August 2011.

[14] Yang, Y. (2023). Library Information Personalized Information Service Management System Based on Improved Genetic Algorithm. 2nd IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics, ICDCECE 2023. https://doi.org/10.1109/ICDCECE57866.2023.10151347

[15] Lops, P., Jannach, D., Musto, C., Bogers, T., & Koolen, M. (2019). Trends in content-based recommendation: Preface to the special issue on Recommender systems based on rich item descriptions. In User Modeling and User-Adapted Interaction (Vol. 29, Issue 2, pp. 239–249). Springer Netherlands. https://doi.org/10.1007/s11257-019-09231-w

[16] Behera, G., & Nain, N. (2023). The State-of-the-Art and Challenges on Recommendation System’s: Principle, Techniques and Evaluation Strategy. SN Computer Science, 4(5), 677. https://doi.org/10.1007/s42979-023-02207-z

[17] Pang, J., Guo, J., & Zhang, W. (2019). Using Multi-objective Optimization to Solve the Long Tail Problem in Recommender System (pp. 302–313). https://doi.org/10.1007/978-3-030-16142-2\_24

[18] Tegetmeier, C., Johannssen, A., & Chukhrova, N. (2023). Artificial Intelligence Algorithms for Collaborative Book Recommender Systems. Annals of Data Science. https://doi.org/10.1007/s40745-023-00474-4

[19] Alhijawi, B., & Awajan, A. (2023). Genetic algorithms: theory, genetic operators, solutions, and applications. Evolutionary Intelligence. https://doi.org/10.1007/s12065-023-00822-6

[20] Neysiani, B. S., Soltani, N., Mofidi, R., & Nadimi-Shahraki, M. H. (2019). Improve Performance of Association Rule-Based Collaborative Filtering Recommendation Systems using Genetic Algorithm. International Journal of Information Technology and Computer Science, 11(2), 48–55. https://doi.org/10.5815/ijitcs.2019.02.06

[21] Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

[22] Jenna Carr (2014). “An Introduction to Genetic Algorithms” Academia.edu.

[23] Stitini, O., Kaloun, S., & Bencharef, O. (2022). An Improved Recommender System Solution to Mitigate the Over-Specialization Problem Using Genetic Algorithms. *Electronics (Switzerland)*, *11*(2). https://doi.org/10.3390/electronics11020242

[24] Hwang, C.-S., Su, Y.-C., & Tseng, K.-C. (2010). Using Genetic Algorithms for Personalized Recommendation. In *LNAI* (Vol. 6422).

[25] Feng, C., Liang, J., Song, P., & Wang, Z. (2020). A fusion collaborative filtering method for sparse data in recommender systems. *Information Sciences*, *521*,365–379. https://doi.org/10.1016/j.ins.2020.02.052

[26] Gad, A. (2018). *Introduction to Optimization with Genetic Algorithm*.

[27] Hwang, C.-S., Su, Y.-C., & Tseng, K.-C. (2010). *Using Genetic Algorithms for Personalized Recommendation*.

[28] Dataset: Goodreads-books “comprehensive list of books listed in Goodreads”  
https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks

[29] Rana, A., & Deeba, K. (2019). Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity). *Journal of Physics: Conference Series*, *1362*(1), 012130. https://doi.org/10.1088/1742-6596/1362/1/012130

[30] Kukkar, G., Khandare, R., & Ali, S. M. (2023). BOOK RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING. *IJARCCE*, *12*(7). https://doi.org/10.17148/IJARCCE.2023.12735

[31] Genetic algorithm tutorial for Python: <https://github.com/ezstoltz/genetic-algorithm>

[32] Feature Selection using Genetic Algorithm (DEAPFramework): <https://github.com/kaushalshetty/FeatureSelectionGA>

[33] Genetic Algorithms with Python: <https://github.com/handcraftsman/GeneticAlgorithmsWithPython>

[34] [GitHub - DEAP/deap: Distributed Evolutionary Algorithms in Python](https://github.com/DEAP/deap)

[35] [Creating Types — DEAP 1.4.1 documentation](https://deap.readthedocs.io/en/master/tutorials/basic/part1.html)

[36] de Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., & Gagné, C. (n.d.). *DEAP-Enabling Nimbler Evolutions*. <https://github.com/DEAP/notebooks>

[37] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard\_score.html

[38] Park, S. H., & Kim, K. (2023). Collaborative filtering recommendation system based on improved Jaccard similarity. *Journal of Ambient Intelligence and Humanized Computing*, *14*(8), 11319–11336. <https://doi.org/10.1007/s12652-023-0464>

[39]: Koren, Y., Bell, R., & Volinsky, C. (2009). "Matrix factorization techniques for recommender systems." Computer, (8), 30-37.

[40]: Rendle, S. (2010). "Factorization machines." 2010 IEEE International Conference on Data Mining.

[41]: Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). "Deep learning based recommender system: A survey and new perspectives." ACM Computing Surveys (CSUR), 52(1), 1-38.

[42] Levandowsky, M., & Winter, D. (1971). "Distance between Sets." Nature, 234, 34-35.

[43] Ricci, F., Rokach, L., & Shapira, B. (2011). "Introduction to Recommender Systems Handbook." Springer, 1-35.

[44] Pazzani, M. J., & Billsus, D. (2007). "Content-Based Recommendation Systems." In The Adaptive Web, Springer Berlin Heidelberg, 325-341.

[45] Zhang, Y., & Ioini, N. E. (2018). "A Survey on the Use of Genetic Algorithms in Recommender Systems." International Journal of Machine Learning and Cybernetics, 9(12), 2067-2080.

[46] Goldberg, D. E. (1989). "Genetic Algorithms in Search, Optimization, and Machine Learning." Addison-Wesley Longman Publishing Co., Inc.

[47] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). "Evaluating collaborative filtering recommender systems." ACM Transactions on Information Systems, 22(1), 5-53