POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A Recommender System Approach for in-silico drug discovery using HPC architectures

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Alessio Russo Introito**

Student ID: 903504
Advisor: Prof. Gianluca Palermo
Co-advisors: Ph.D. Davide Gadioli, Ph.D. Stefano Cereda
Academic Year: 2021-2022

Abstract

The Covid-19 pandemic motivated the urge for improvements in the research of new drugs. Recently, the virtualization of screening techniques has been successfully integrated into Drug Discovery pipelines to generate a fast and approximate investigation of the chemical space. In particular, advances in technology made possible the application of powerful docking methods that leverage the three-dimensional structure of the proteins to provide better analysis. However, *Molecular Docking* approaches are computationally intensive tasks, usually deployed in HPC environments to enable large scale of concurrency.
In this thesis, we adopt classical Recommender Systems' models to prioritize the evaluation of the most promising protein-ligand complexes that could be formed, which should be further analyzed via in-laboratory experimentation. Since no dataset is currently available to feed the models, an in-house docking pipeline has been built, thanks to the collaboration with IT4Innovations' Super-Computing Center. The results of the experiment conducted show that our solution can efficiently exploit a set of evaluated protein-ligand interactions to *re-rank* the sequence of molecules to be visited when a new protein is taken into account. This optimization can enhance the time-to-solution in drug discovery and thus amortize the costs of execution.

Keywords: virtual screening, molecular docking, recommender systems, hpc

Sommario

La pandemia di Covid-19 ha motivato l'urgenza di miglioramenti nella ricerca di nuovi farmaci. Recentemente, la virtualizzazione delle tecniche di screening è stata integrata con successo nelle pipeline di Drug Discovery per generare un'indagine rapida e approssimativa dello spazio chimico. In particolare, i progressi tecnologici hanno reso possibile l'applicazione di potenti metodi di docking che sfruttano la struttura tridimensionale delle proteine per fornire una migliore analisi. Tuttavia, gli approcci Molecular Docking richiedono attività computazionalmente intense, implementate solitamente in ambienti HPC per consentire una concorrenza su larga scala.
In questa tesi adottiamo i modelli classici di Recommender Systems per dare priorità alla valutazione dei complessi proteina-ligando più promettenti che potrebbero formarsi, i quali dovranno essere ulteriormente analizzati tramite una sperimentazione in laboratorio. Poiché al momento non è disponibile alcun dataset per il training dei modelli, è stata costruita una pipeline di docking, resa possibile grazie alla collaborazione con il Centro di Super-Computing di IT4Innovations. I risultati dell'esperimento condotto mostrano che la nostra soluzione può sfruttare in modo efficiente un insieme di interazioni proteina-ligando, precedentemente valutate, per ri-ordinare la sequenza di molecole da visitare quando una nuova proteina viene presa in considerazione. Questa ottimizzazione può migliorare il tempo necessario impiegato nella scoperta di nuovi farmaci e quindi ammortizzare i costi di esecuzione.

Contents

Introduction

Drug development still represents one of the main challenges in the pharmaceutical industry where large capital are invested to improve the generation of new candidate medications. The cost of research and development to bring a single drug on the market has grown over the years, up to an average estimated cost of $2.8 billion[64] per drug. This expenditure is employed to support the entire process of drug approval (including the cost of failures), which consists of a lengthy and complex pipeline of execution that needs an average time of 10-15 years to be completed. Indeed, multiple phases are operated sequentially starting from the *discovery* of potential chemical entities and followed by a deeper evaluation of the latter during *pre-clinical* and *clinical* testing phases before reaching the market. Despite the expensiveness and the laboriousness of the process, the rate of new therapeutic medication discovery stays low, since less than 12% of clinical trials eventually evolve into an approved medicine[18].

In particular, in the early stages of the pipeline, the *drug discovery* phase plays a fundamental role in the identification of new potential drugs. In fact, given a large library populated by chemical compounds, it aims at detecting those elements that show a certain affinity with the target molecule they bind on. The target is generally interpreted as a large macro-molecule such as a protein, whereas the evaluated compounds (called *ligands*) are structurally much smaller in size. Drug discovery phase should fulfill two important requirements: since the chemical libraries can contain billions of molecules, their screening should be fast enough to meet the time requirements; in addition, it should be able to correctly filter out the compounds that don't provide any chemical activity with the target protein.

To perform the screening of chemical libraries, an *high-throughput screening* (HTS) is generally applied, characterized by parallel execution of thousands of in-vitro trials via robotics automation. In modern solutions, an additional method, named *Virtual Screening*, is typically placed before the in-laboratory HTS. Virtual Screening emphasizes the use of computer-aided simulations to virtually evaluate the binding force of a protein-ligand pair. This approach allows scaling the analysis to billions of compounds leveraging the parallelism of High-Performance-Computing infrastructures, where approximated so-

lutions are preferred over detailed ones to guarantee "fast" enough evaluations.
Among the virtual screening methods, in recent years *Molecular Docking* has gained lots of interest in both pharmaceutical and academic fields as a powerful technique that exploits a three-dimensional representation of a protein to measure to what extent the latter is compatible with a ligand, studying the forces that regulate their interaction. In particular, docking algorithms *search* in the three-dimensional space for the reciprocal best orientation and conformation of the protein-ligand pair in order to evaluate the binding affinity. Due to the enormous amount of possible ways to combine two molecules in the three-dimensional space, Molecular Docking techniques become computationally intensive tasks, especially if the evaluation is extended to datasets with billions of chemical elements. Once the chemical library is completely screened, only the top performing protein-ligand complexes are allowed to access the next stage of the drug development pipeline.

Thesis Motivation

Given a new protein, molecular docking applications aim at screening the entire dataset of molecules to find the best chemical complexes. However, docking methods are computationally intensive and analyzing databases with millions of compounds can be time-consuming and expensive considering the cost related to HPC infrastructure. During the simulation, no particular order of the ligands is taken into account for the evaluation, such that the elements are picked randomly from the database. This approach could impact the expected result of the algorithm especially when the simulation is subjected to fixed computational availability that limits the amount of estimations a user can perform. Moreover, the hugeness of chemical space makes it impossible to test all the molecules, thus choosing only a relevant subset could greatly improve the overall performance of the drug development pipeline.

Thesis Contribution

In this thesis, we compare the random selection performed by state-of-the-art methods with respect to a precise sorting of the ligands in the chemical library. In particular, we focus on the prioritization of those molecules that are likely to be the best fit for the protein in account. The entire context can be interpreted as a Recommendation Systems (RS) problem where a set of *items* is recommended to a *user* based on the analysis of its past interactions. In this chemical scenario, the goal is to "recommend" a collection of the most promising molecules to a protein, so that the probability to find, in the early stages of the simulation, the protein-ligand pairs with the highest binding affinity increases. To extract those suggestions, we leverage a collection of affinity scores that derive from

previous evaluations of protein-ligand pairs. In such a way, the most likely candidates are evaluated first, while the others are postponed, guaranteeing a sorted sequence of molecules to be submitted. To validate our work, we are inspired by a real-case scenario where a *new protein* is taken into account to be tested against a library of compounds; given the wide size of the libraries, the evaluations are made in batches, following an iterative process until all protein-ligand pairs have been considered. For this purpose, we focus on a custom *cross-validation* technique in which each fold is composed of a singular protein and subjected to multiple rounds of recommendations.

Thesis Outline

This work has been structured in different chapters. Chapter 1 provides an introduction to the Virtual Screening techniques and the state-of-the-art methods in Molecular Docking field. Chapter 2 highlights the main features of High-Performance-Computing architectures and the common methodologies to handle the computation in a distributed environment. Chapter 3 briefly describes the main concepts to face Recommender Systems problems, while Chapter 4 details the proposed approach along with the RS models we adopted and the generation of the data used to train them. Eventually, Chapter 5 shows the results of the models and how their quality has been measured with proper evaluation metrics.

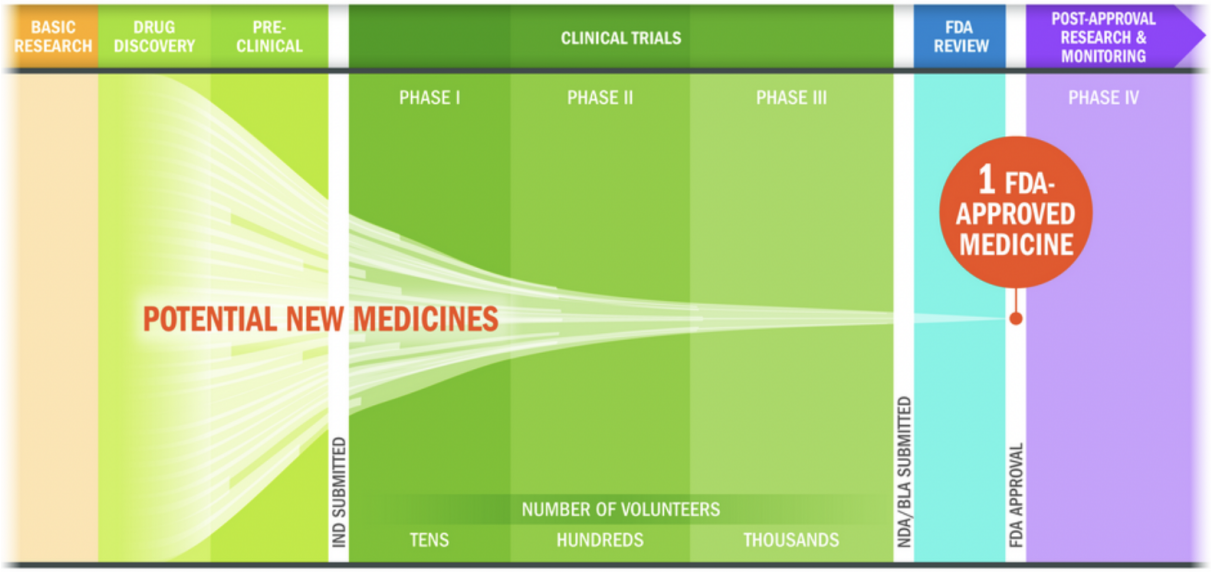


Figure 1: Drug Development Process [53]

1 | Background and Previous Works

This chapter provides an overview of the main components and methodologies we will interact with and it gives an introduction to the research field related to computational chemistry. We start from a broad description of the biology of a protein and how and why the protein-ligand interaction is a fundamental topic in pharmaceutical research to study complex biological and chemical systems, explaining the difference between in-vitro, in vivo and in-silico experiments. Then we move forward with the state-of-the-art virtual screening strategies. Lastly, a detailed summary of the current methodologies applied in molecular docking is given along with a comparison between them.

1.1. Computational Chemistry

In the last decade, important discoveries in different fields of science have been made thanks to big improvements in technology. More and more scientific researches heavily rely upon wide-range and expensive simulations such that the usage of large-scale systems has become a fundamental requirement.
Among the different sectors, *Computational Chemistry* has stood out, especially in recent years, as a branch of chemistry that makes extensive use of computer simulation to model and assist in solving chemical problems. Due to the COVID-19 pandemic, it is increasingly receiving attention and participation. Big companies and pharmaceutical industries started to invest lots of their resources searching for new methodologies to discover new drugs or to predict behavior at the microscopical level.
This mixed approach of applying computational techniques to problems of theoretical chemistry is not only a way to speed-up drug discovery, but it is needed in problems that cannot be solved analytically: a well-known example could be *the quantum many-body problem*, which refers to the analysis of the properties and the conduct of many interacting particles in a generic system. Computational Chemistry contains a wide range of sub-categories that differ one from the other by the chemical system considered and,

consequently, by the number of technological resources in use. A system, in general, can vary from a single molecule, a group of molecules or a solid, or, otherwise, can aim to analyze atoms and their chemical bonds.
One of the main achievements in the last couple of years is surely due to DeepMind's AlphaFold [37]: their result was described as "astounding" [5] and transformational. In fact, they (partially) solved the *protein-folding* problem, an open problem that emerged around 1960 with the appearance of the first atomic-resolution protein structures. Protein-folding can be trivially defined as a prediction challenge where, given a sequence of amino-acids that compose a protein, the goal is to characterize how this sequence spontaneously folds itself to generate eventually a three-dimensional structure. AlphaFold's team applied advanced techniques of deep-learning to successfully tackle one of the hardest challenges in biochemistry, making their model predictions be comparable to experimental results.
A major impact that such a solution can lead to, implies surely an amortization of costs and a productivity growth. In fact, the experimental process to determine the three-dimensional protein structure is currently based on expensive and time-consuming techniques: it employs procedures such as X-ray crystallography, Nuclear Magnetic Resonance spectroscopy (NMR spectroscopy), cryo electron microscopy (cryo-EM) and dual polarisation interferometry.
As well as for protein-folding problem, other biological challenges[46] can be addressed computationally to support experimental research and reduce the time required to process and analyze an even larger volume of data; such developments could lead to faster solutions in drug discovery and a better understanding of biological phenomena.

Computational Chemistry is a wide-range area of research composed of several branches, each specialized in a particular domain. A strongly related field is **Bioinformatics**, which spectrum of research combines different disciplines including biology, mathematics, chemistry, physics and computer science. It is reasonable to collocate bioinformatics as a direct descendant of Computational Chemistry, even if the difference between chemistry and biology is very slight.
Whereas pure chemistry focuses on nature at its lowest possible level, dealing with elementary particles, such as electrons and the phenomena that govern them, *biology*, instead, despite its strong foundation on chemistry, has a different target: the study of life in all its nuances. In the same way as chemistry, biology encloses a multitude of categories and the examination of life happens at multiple levels of organization: from the evolution of populations to the anatomy and physiology of plants, and the molecular biology of a cell. In particular, the latter (molecular biology) can be reckoned as a common topic between the two disciplines.

Molecular biology aims to understand the rules upon which biological activity is triggered by considering both inter-cell and intra-cells environments. In general terms, the study of chemical and physical structure of biological molecules is known as *molecular biology*, as well as all the corresponding molecular activities. It also plays a crucial role in the recognition of structures, functions and internal controls within individual cells, all features that can be used to efficiently target new drugs and get a better understanding of cell physiology. A famous quote by Francis Crick [13] in 1957, and then re-stated in 1970 [15], defined the well-known *central dogma of molecular biology*, which states that once "information" has passed into protein it cannot get out again. It is often interpreted as "DNA makes RNA, and RNA makes protein", although this is not the proper meaning. This reference highlights how the transit of information works at molecular-level, where the transfer of "data" from nucleic acid to protein or to another nucleic acid is possible, but once the information ends up in a protein it cannot be transferred back to other proteins or nucleic acid. In this context, the protein represents the final worker, in charge of effectively computing a job. So, this flow of biological information describes how the communication between nucleic acids (DNA and RNA) is aimed at synthesized proteins (*Protein biosynthesis*), which in turn are in charge of performing a number of critical functions.
As you can notice, protein is a recurrent topic in most of the fields of chemistry and biology, for this reason it is appropriate to give it a proper definition.

In chemistry, *amino-acids* are organic compounds, i.e. compounds that contain carbon-hydrogen bonds, composed of amino ($-NH_3^+$) and carboxylate ($-CO_2^-$) functional groups. They can be categorized according to the position of the core structural functional groups, as *alpha*, *beta*, *gamma* and *delta*. A short chain of amino-acids is called *peptide*, where its molecules are linked by particular bonds known as *peptide bonds*. As a consequence, a *poly-peptide* is a longer and continuous peptide chain. By definition, a poly-peptide that contains more than approximately fifty amino-acids is called **protein**.
Thus, a protein refers to a very large macro-molecule composed of thousands of covalently bonded atoms, assembled to create a linear *polymer* (substance composed of many repeating sub-units) of amino-acids.
Once a protein has been synthesized by the combined activity of DNA and RNA, the corresponding poly-peptide must fold properly in order to produce *active sites* and carry out its function. As briefly discussed in a previous paragraph, Protein folding corresponds to the natural transformation process that brings the poly-peptide from a simple sequence of amino-acids into its correct three-dimensional *native conformation* [50]. The chain of

amino-acids determines the folding and the function to compute. The final structure is not entirely rigid, instead its conformation can change during the time, mostly because of interactions with other molecular entities happen.

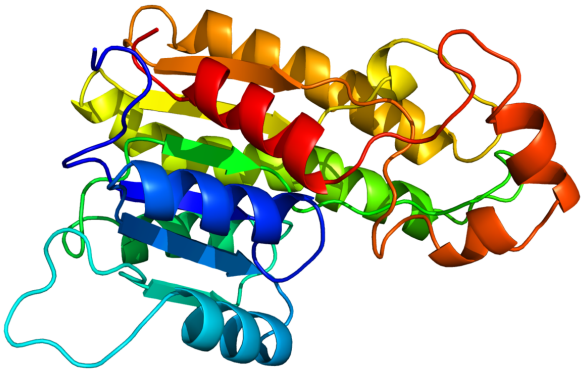


Figure 1.1: 3D Structure of DHRS7B protein

Proteins are essential biological agents, involved in plenty of activities. They can be described according to their large range of functions in the organism and summarized as follows:

- Antibody: bind to a specific foreign particle to help protect the body.
- Enzyme: responsible for almost all chemical reactions that happen within the cell.
- Messenger: transmit signals to coordinate biological processes between cells, tissues and organs.
- Structural component: provide structure and support for cells.
- Transport: bind and carry atoms and small molecules inside a cell and throughout the body

What allows proteins to accomplish their set of functions is their intrinsic ability to bind to other molecules. The three-dimensional structure contains a sort of "pockets" (named *binding site*) on the molecular surface which is the region where the interaction could occur. These binding site pockets are dictated by the so-called *tertiary structure* of the protein, one of the aspects biochemists refer to when dealing with protein structure: usually a protein can be described by different (three or four) levels of organization, named *primary, secondary, tertiary and quaternary* structure; the tertiary is what controls the basic function of the protein.

Protein can bind to different types of molecules, but the binding can be extremely tight and specific. The binding partner is often referred to as a **ligand**, a general term to define a substance that forms a complex with a macro-molecule to serve a biological purpose. The binding of a ligand to a binding site on protein triggers a change in the conformation structure of the latter. This change initiates a sequence of events leading to different cellular functions.
A ligand could represent another protein, generating a protein-protein interaction involved in essential activities like signal-transduction or cell metabolism. In the same way, a small-molecule can bind to a protein pocket with the intention of being transported to other locations in the body.
This kind of interaction produces a *protein-ligand complex* that is formed following a process called *molecular recognition*, which refers, in general, to the ability of two or more molecules to interact one with the others through non-covalent bonding such as hydrogen bonding, hydrophobic forces or van der Waals forces, showing a sort of molecular complementarity. Molecular recognition is an important mechanism that appears in essential processes like self-replication, metabolism and information processing.
So, the protein-ligand complex implies a ligand to bind on a pocket of the protein through molecular recognition. However, the molecular recognition depends on two main factors:

- **Affinity**: it explains the interaction force between the protein and its specific partner. If the attraction with the ligand is strong, then it results in a high-affinity while low-affinity ligand binding involves less attractive forces. In high-affinity ligand binding, a physiological response is triggered by a relatively low concentration of a ligand, which should be adequate to maximally occupy a ligand-binding site. Moreover, it is possible that part of the binding energy is used to modify the protein naive-conformation, altering its behavior
- **Specificity**: the capability of the protein's binding site to bind *specific* ligands, such that the fewer ligands a protein can bind the greater its specificity. On the other hand, increasing the number of ligands that can bind to the protein, the specificity decreases. The strength of electrostatic and hydrophobic interactions influence positively the specificity.

So, in the previous example of a protein able to transport a small-molecule, the interaction happens because of the high binding affinity between them and it occurs in regions where the ligand is in high concentrations. Once the complex reaches the target location, the protein must release the ligand in an environment where the presence of the latter is low. The classical example is the hemoglobin, which transports oxygen from the lungs to other organs and tissues, starting from a high-level ligand concentration region (lungs) to

a low-level concentration.
The factors [19] that lead to high-affinity binding and high-specificity for a target are a good fit between the surface of the two molecules in their ground state and charge complementary. This means that the selection for high-affinity binding automatically leads to highly specific binding. This principle can be used to simplify the screening approach aimed at generating useful drugs.
Different methods to study the protein-ligand interaction exist, mainly focused on hydrodynamic, calorimetric techniques and several others based on spectroscopy and structural methods. However, the remarkable growth of computer power over the last decade creates an original approach to face the research of molecular interactions, exploiting both super-computers and personal computers[1]. In April 2007 a worldwide grid of millions of ordinary computers was exploited for cancer research[2].

1.2. Virtual Screening

The examination of the activity behavior and the structural properties of proteins can be conducted in three different ways:

- **In-vitro**: this approach allows to operate a more detailed and convenient analysis taking into account microorganisms, cells or biological molecules outside their normal biological environment. Basically, it uses components of an organism isolated from their context. The main advantage consists in a simplification of species-specific analysis with respect to researches performed considering the whole organism. It allows also a more detailed investigation of basic biological function since it is focused on a particular section, removing the complexity that an entire system can generate. As a result, in-vitro evaluations can be miniaturized and automated, yielding high-throughput screening methods for testing. On the other hand, extrapolating components from their surroundings has an unavoidable consequence of losing the global perspective of the intact organism such that the outcome may not fully or accurately predict the effect on the general system. For this reason, it is a common practice to submit the extracted drug into a sequence of *in-vivo* trials to determine its effectiveness.

- **In-vivo**: as opposed to the previous case, *in-vivo* experiments are those in which the evaluation of drugs or biological entities is tested on the whole living organism or cell. As mentioned, in-vivo allows understanding better the overall effect of an experiment on a living subject, driving the testing phase directly towards animals,

including humans, and plants. This approach is crucial since often *in-vitro* methods can bring to false drug candidates.

- **In-silico**: in biology, an *in-silico* experiment is the one performed, partially or fully, in a virtual environment or via computer simulation. Due to the vastness of the molecular space and the elevated costs associated with in-laboratory experimentation, a virtual simulation can be developed to perform an initial screening phase in order to limit the number of compounds to be delivered toward a more accurate analysis. Indeed, in-silico methods execute a fast, but approximated space search such that the relevant results are submitted to more precise and slower in-vivo/in-vitro approaches.

Even if *in-vitro* takes advantage of high-throughput screening via robotics labs to physically test thousands of diverse compounds, in terms of speed *in-silico* simulation outperformed the others. In 2007, a group of researchers from the University of Surrey developed an in-silico genome-scale model of a microbe that causes tuberculosis [56]. The Surrey team showed that the model successfully simulates many of the peculiar properties of the TB bacillus, but unlike the biological organisms, the in silico TB bacillus grows in nanoseconds so experiments that would normally take months can be performed in minutes.
Despite the speed-up obtained, the precision of in-silico computation may be compromised by strong assumptions that constrain the freedom of the model, detecting areas of interest rather than exact solutions.
There are millions of chemical 'libraries' generated by *combinatorial chemistry* that contain a huge number of compounds : this technique allows an artificial execution of chemical reactions to obtain one or several products and make it feasible to prepare thousands or millions of compounds in a single process. However, such an amount cannot be synthesized entirely and a precise selection of compounds should be considered. The only way available is to design a computer program capable of evaluating and designating specific compounds in very large libraries. This process is called **Virtual Screening**.

Virtual screening can be defined as a set of (in-silico) computational methods that automatically analyzes large libraries of small-molecules in order to identify potential hit candidates, the ones that are most likely to bind a target protein.
Searching through the entire *chemical space*[45] composed of over 10^{63} possible compounds is not theoretically feasible. Virtual Screening methods have the goal to scale down this enormous chemical space filtering compounds in order to reach a manageable number that can be synthesized, purchased and tested. As the accuracy of the method has increased,

virtual screening has become an integral part of drug discovery process.
The crucial point is to transfer as much information as possible to the virtual screening simulation to increase the rate of successful evaluations of a large number of compounds. Information that can help the search strategy[62] resides in knowledge about:

- the interaction between the ligand and the receptor (protein)
- the structure of the receptor
- others ligands that interact with this receptor

As a rule of thumb, the more information is available, the more efficient the Virtual Screening phase can be.
There are a few initial basic assumptions[62] to limit the search space. Instead of trying to synthesize as many compounds as possible, it is better to focus on specific classes: many of the molecules are not desired, since certain combinations of functional groups are not compatible. A second option is to avoid a full enumeration of the virtual library: as chemists get closer to development candidates, they may begin to fine tune their structure and make small changes; in the same way, a computer program can perform slight changes to interesting compounds that are discovered in a first pass of the library.

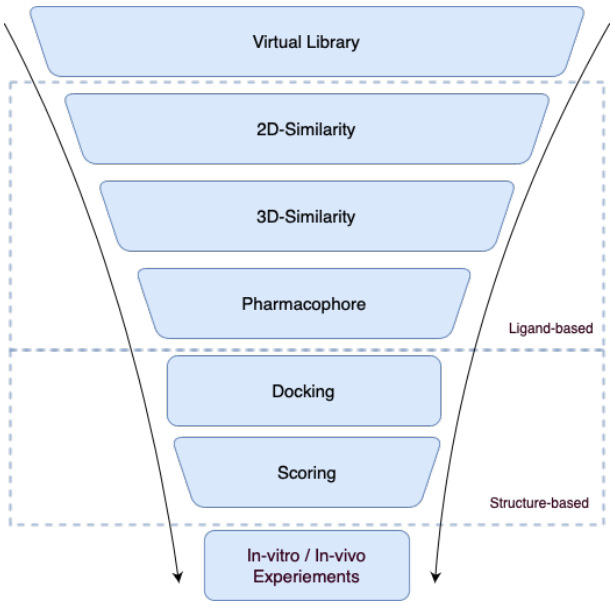


Figure 1.2: Virtual Screening Pipeline [44]

A classical Virtual Screening pipeline is shown in the previous Figure 1.2. The initial virtual library undergoes a sequence of steps aimed at down-sampling the chemical space to pass part of the compounds to the next phase such that molecules that cannot possi-

bly match the active site are filtered out. For example, techniques such as 2D-similarity and ligand-clustering, discussed in detail in Section 1.3, can be applied quite immediately, since they do not require particular computational effort, whereas 3D-simulations involve a longer process since the three-dimensional conformation extraction through X-ray crystallography and/or NMR could be expensive, so limit the number of elements to a selection of compounds could be crucial.

As described in Figure 1.3, there are two broad categories of screening techniques that differ from the knowledge of three-dimensional structure of the receptor:

- **Ligand-based methods**: if the structure of the receptor (protein) is unknown, a set of structurally diverse ligands that binds to the target can be used. In fact, the knowledge of a collection of binding/non-binding molecules serves as ground truth for extracting similar compounds or training Machine Learning models to classify the activity of new ligands.
- **Structure-based methods**: in this case, the receptor structure of the investigated active ligands is well-defined. Structure-based approaches are quite expensive and imply a parallel computing infrastructure to complete their tasks.

In the following sections, a detailed description is provided.

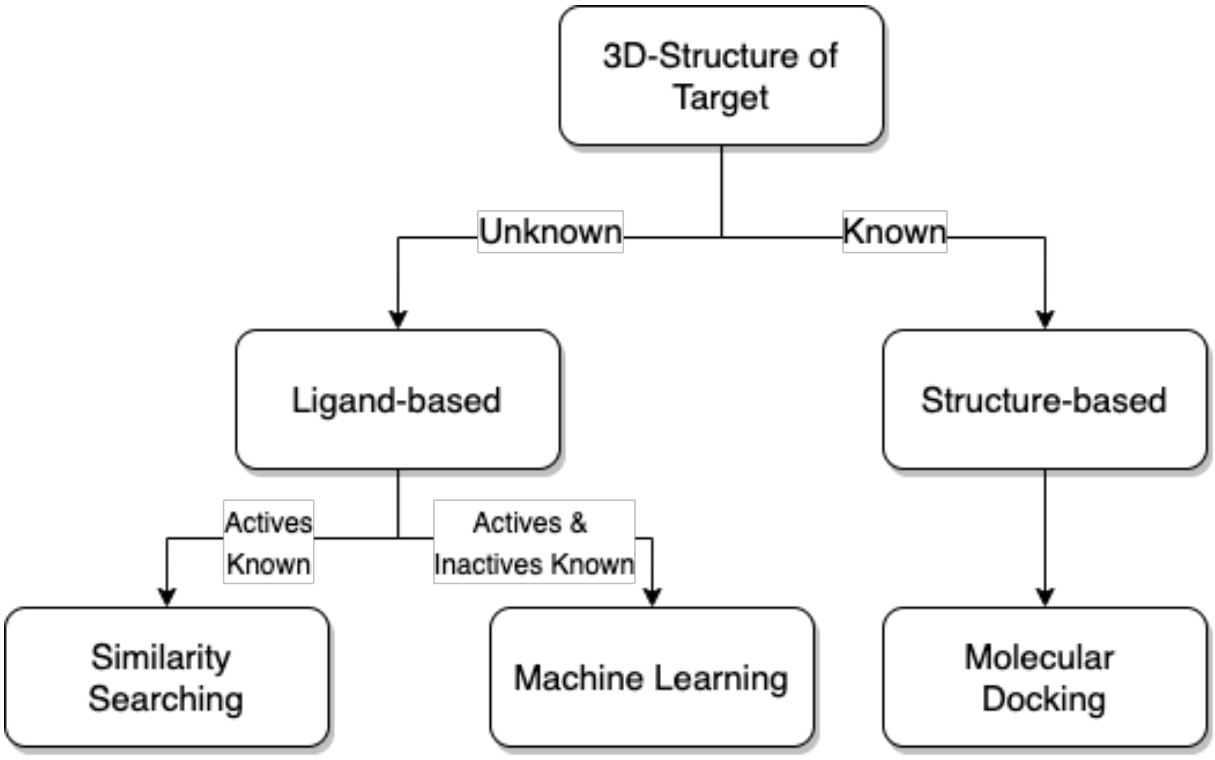


Figure 1.3: Taxonomy of Virtual Screening methods

1.3. Ligand-based Virtual Screening

As stated, *ligand-based* approaches don't rely on the knowledge of the three-dimensional structure of the receptor, but instead on a set of compounds that are known to bind to the target. The basis for these methods is the *similar property principle* introduced by Johnson and Maggiora [36], which states that similar compounds have similar properties. Thus, ligands with high similarity to reference compounds are likely to behave in a similar fashion and therefore have similar effects. Ligand-based methods don't require an extreme computational effort compared to structure-based methods as no macro-molecules three-dimensional representations are involved in the calculation. As a result, a single CPU is adequate to perform a large-scale screening simulation.
Different similarity methods are used in this process [29], mostly based on 2D-representation of molecules that are easy and fast to be extracted and generally applied in the early stages of the VS pipeline as shown in Figure 1.2. The search moves toward ligands that are chemically and structurally similar to the reference compounds: ligands with similar chemical properties can actively "react" to the target and bind themselves; on the other hand, structurally similar ligands will fit the target's binding site and hence will be likely to bind.
2D-molecule structures cannot be directly employed to compute similarity, instead they should be translated into a more computer-friendly representation. For this purpose, a *molecular fingerprint*[11] is widely adopted for similarity searching. Fingerprints are a way of encoding the structure of a molecule in a sequence of bits, each of them including certain information regarding the molecule.
This conversion from 2D-structure to a fixed-length vector is not expensive and could be easily scaled to a vast range of compounds. Consequently, it becomes trivial to make a comparison between two molecules' fingerprints such that similarity functions can efficiently be evaluated. There are different kinds of fingerprinting transformation that can be applied which rely upon distinct factors. According to the nature of the bits, the fingerprinting methodologies are classified in different groups[11]: *sub-structure keys-based, topological, circular or pharmacophore.*

Once the fingerprints are available, a number of similarity methods can be applied to assess which molecules are more similar to the active compounds considered. However, the **Tanimoto coefficient** is the most popular.
Given two fingerprints of compounds A and B:

$$\text{Tanimoto coefficient} = \frac{c}{a + b - c}$$

where a stands for the amount of bits set to 1 in A, b the amount of bits set to 1 in B and c the amount of bits set to 1 in both A and B (intersection between the two vectors). As you can notice, the Tanimoto coefficient points out the fraction of common bits over the total number of bits set to 1, without counting duplicates. This value is by definition "normalized" by 0 and 1, so it can also be perceived as a percentage of similarity between the two compounds.
The compounds can be sorted according to the Tanimoto coefficient in order to process first the high-similar ligands, following the *similar property principle*. To reduce the instances to inspect, a solution could consist of the selection of a cut-off to filter out all the molecules whose similarity score is under that value, keeping the most similar compounds. The entire process is shortly described in Figure 1.4.

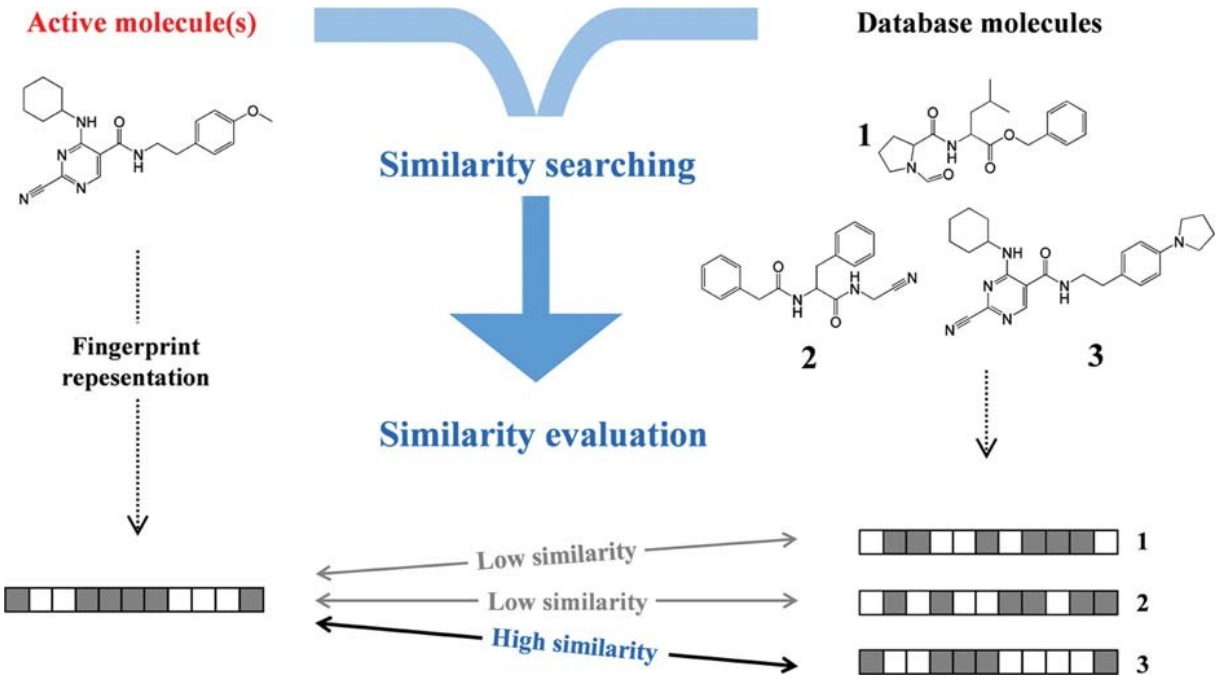


Figure 1.4: Similarity Evaluation by Fingerprint representation [54]

In spite of the simplicity and easiness of similarity searching methods, it has some limitations[29]:

- Not always a high similarity score is correlated to an activity of the compound. It is possible that small structural changes alter the effective activity of the compound in the binding site.
- The selection of the cut-off could be critical. There is no way to effectively evaluate the choice of the value. High cut-off value could reduce a lot the chemical space

in consideration, but could remove possible candidates from the search. Moreover, similarity scores depend on both the fingerprint methods and similarity function adopted, then the optimal cutoff will be dependent on these two factors.

- Most similarity functions, Tanimoto coefficient included, weigh evenly all the bits of the fingerprint vector. This could be, in general, a big issue and lead to wrong outcomes: inactive ligands that do not share with the active compound some critical feature could become selected as a good candidate; as opposed, active ligands that only match critical features could be removed by the processing. The weights of each bit should be properly selected, maybe exploiting machine learning models.

Machine Learning models can be taken into account if both active and inactive compounds are known a-priori. Complex techniques[41][43] can be embraced, mainly based on *supervised* approaches like classification and regression tasks. Molecular fingerprint vectors could be useful and widely used in order to predict compound activity. Machine learning algorithms can successfully face the limitation previously described. If molecular fingerprint vectors are mapped as features of the model, given a training set, these algorithms can learn which bits are more relevant for bio-activity and properly classify new chemical entities (test-set) based on these assigned bit-weights. This should make the comparison much fairer and constitute their major strength with respect to other ligand-based methods.

Finally, if the 3D-structure of the target molecule is known, sophisticated analysis can be performed, as described in the following section.

1.4. Structure-based Virtual Screening

1.4.1. Molecular Docking

X-ray crystallography and protein nuclear magnetic resonance (NMR) spectroscopy increased the number of three-dimensional protein structures available. The main purpose aims to study solutions to predict possible protein-protein or protein-ligand interactions, without that further experiments taking place. Thanks to a massive computational power and advanced methodologies, complex analysis can be carried out to understand the reciprocal behavior of a protein-ligand pair.
In this scenario, **Docking** is the method to predict the best orientation and conformation of a ligand with respect to the receptor it binds to. In particular, a ligand can assume multiple conformations, dictated by the number of *rotatable bonds* that compose it: as

the name suggests, a rotatable bond allows the molecule to freely rotate around itself. This type of bond splits the molecule into two different sub-sets of atoms (*fragments*) such that each one can be rotated independently without altering the chemical properties. The resulting conformations are called *rotamers* (rotational isomer), which are distinguished by the difference in the value of the total energy: the latter is defined by the attractive and repulsive forces caused by the displacement of the atoms, thus different displacements imply an alteration of the energy. The number of rotatable bonds gives a measure of the molecular flexibility such that the higher this number, the higher the number of rotamers and so the higher the number of possible conformations. Therefore, to the six degrees of freedom for placing the rigid body of the ligand in a three-dimensional space, each rotatable bond adds two more degrees of freedom to the problem. As soon as the amount of rotatable bonds increases, the problem becomes intractable in its completeness.
On the protein side, the binding happens in one of the possible binding-sites, known informally as "pockets". The knowledge of the preferred pose can be used to predict the strength of the interaction, which measures how much the two entities are able to form a stable complex molecule.
As well as ligand-based solutions described, *Molecular Docking* is a powerful computational technique integrated into Virtual Screening campaigns as one of the most frequently used *structure-based* methods, particularly useful to reduce the chemical space in the early stages and filter out inactive compounds.
The course of the docking process emulates the molecular recognition operation in which both the protein and the ligand change their conformation to achieve an overall best-fit, optimizing their interactivity. Simulations in this field involve two different phases:

1. Searching Algorithms
2. Scoring Functions

Compared to ligand-based methods, docking offers a deeper introspection of the interaction activity, taking advantage of three-dimensional representations. Although the similar property principle reflects the common behavior of the compounds, one of its weaknesses regards the impossibility of taking into account that small compound modification may result in completely different activity on the selected pocket of the target. To avoid this type of erroneous prediction, a docking process can be considered. On the other hand, since we are dealing with macro-molecules in a three-dimensional space, much more information needs to be processed and interpreted, making structure-based algorithms more computationally expensive than ligand-based methods.

1.4.2. Searching Algorithms

Searching algorithms have the purpose of exploring the positional and conformational space in order to find the best reciprocal pose for the protein-ligand complex (Figure 1.5).

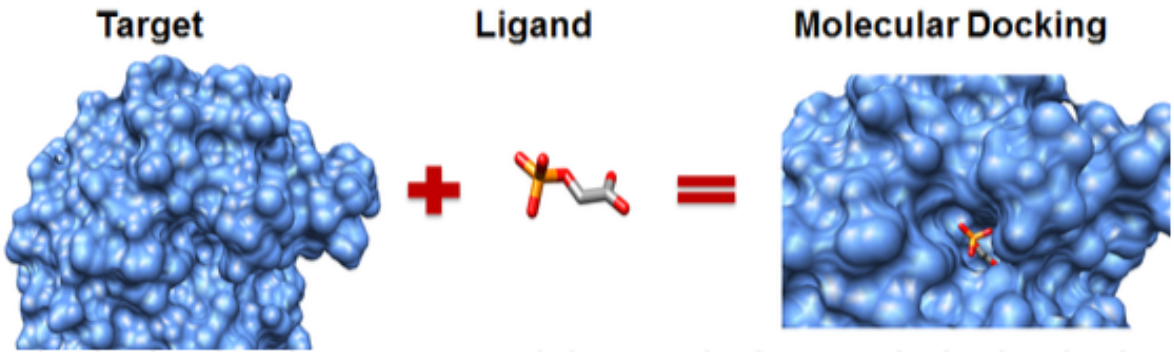


Figure 1.5: Docking representation [57]

The search space is represented by all the possible orientations of the protein with respect to the ligand. If the protein is also allowed to change its conformational shape, then all the possible combinations of conformations of both protein and ligand should be considered. Moreover, the surface of the protein arranges multiple pockets, adding further complexity to this process. It is a computationally difficult problem since several ways of putting two molecules together exist and, actually, with the current computational resources, it is impossible to scrape the entire search space and screen large databases of compounds. As a consequence, a trade-off between the computational effort and the space examined must be reached such that speed and effectiveness in covering the relevant conformational space are guaranteed.
Searching algorithms distinguish each other in the general way they approach the problem. Many factors are involved in a molecular docking process, mainly dictated by the type of exploration one is intended to pursue.

The search takes into account multiple or all binding-sites of the target macro-molecule. It is not uncommon that the binding-site location is known a-priori[59]. If it is not the case and the binding-site is missing, two strategies are available to solve the problem: either an algorithm can be used to predict the most probable binding-site or a *blind-docking* [33] simulation is carried out. However, the latter consumes lots of computational resources, since its search includes the entire surface of the protein for binding sites, in particular it focuses on the factors influencing the accuracy of the final structure like the number of torsional degrees of freedom in the ligand (AutoDock). For what concern the former, many available software can be used to predict the location of the bind (MolDock,

DoGSiteScorer, Fragment Hotspot Maps), each of them based on a particular method such as integrated cavity detection algorithms[58].

The reduction of the number of binding-sites contributes to decrease the space to be evaluated and can help to speed-up the process. Unfortunately, when the ligand flexibility is taken into account, then the number of possible ligand binding to be considered increases dramatically. In the conformational search, multiple structural features of the ligands, such as translational, torsional and rotational degrees of freedom, need to be incrementally modified during the computation to find the best poses and orientations. For this reason, docking algorithms can be classified in two more categories:

- Rigid-body Docking: it avoids every kind of flexibility, neither ligand nor receptor, which obviously limits the precision of results. Instead, it considers essential geometrical features of both molecules and looks for complementarities in the shapes: it is designed to find the molecules with a high shape-affinity with respect to the binding-site. This type of docking is usually utilized to perform a fast initial screening procedure of small molecule databases.
- Flexible-body Docking: after the first screening through rigid-body docking, the flexible docking starts. Due to its intensive computation, its aims at optimization and specific refinements. While the rigid-body limits its freedom to six degrees (translation and rotation), in this case the flexibility can be exploited in its totality. Usually, the flexibility is restricted only to ligands, while the target receptor is assumed to be rigid, thus the pair is called *rigid-flexible*. On the other hand, powerful techniques employ much more effort in trying to describe a complete flexible scenario in which both the ligand and the protein are free from restrictions. *Flexible-flexible* approaches greatly improved the accuracy of the docking, but an extremely complex space has to be explored.

Despite advances in computational methods, docking is still a big challenge to face. Heuristic methods are required to efficiently handle (ligand) flexibility and the corresponding algorithms can be categorized in different classes: *systematic*, *stochastic* and *deterministic* methods. Those methods, implemented by the software in Figure 1.1, differ from each other by the algorithm used and by how much space it is going to explore.

Systematic Methods

Systematic methods explore each ligand's degree of freedom incrementally: ligand structure is slightly modified along with its conformation in order to analyze all possible solutions. The molecule conformations and rotations are dictated by its bond and the angle

defined in each bond. Thus, the higher the number of rotatable bonds present in the molecule, the more complex the search space will be. It follows that systematic searches are prone to undergo a combinatorial explosion of the space. Due to the incremental search, this approach is quite effective in exploring the conformational space, although it requires a large number of runs and trials. Systematic searches eventually converge to the minimum energy solution, which corresponds to the best binding mode. Although the method is functional in exploring the space, it can converge to a local minimum.
Systematic search can be referred to as exhaustive or fragment-based.

Exhaustive They scan the entire space in a predefined order[32]. Exhaustive search algorithms discover the best ligand conformation by systematically rotating all the viable rotatable bonds [65]. Given the huge amount of trials that has to be performed, large conformational space makes the full scan unfeasible. Instead, different heuristics are able to approximate a complete systematic search space of the docked ligand [25], focusing the attention on regions that are likely to contain good ligand poses.

Fragment-based It is a method to discover powerful compounds based on an *Incremental Construction* approach. The algorithm starts by dividing the ligand into small parts, known as **fragments**, usually identified using sensitive biophysical methods. They represent small chemical structures characterized by low complexity, low molecular weight and low binding affinity with the target[40]. Then, the fragments are evaluated in the binding-site one at a time and each one incrementally grows or combines itself to other fragments through covalent bonds. These steps are repeated until a lead with high affinity is obtained. The incremental construction can be performed in two different fashions:

1. The molecular fragments are docked into the active region and linked covalently (called *de-novo ligand design*)
2. The docked ligands are partitioned into a *rigid-core* and flexible parts (*side chains*); the core is docked first into the binding site and side-chains are added incrementally, following geometrical constraints and binding affinity values.

In machine learning theory, it could reflect a sort of boosting ensemble method, which starts from a weak learner and incrementally obtains a powerful model. It allows an efficient exploration of the space with a relatively small sampling, thanks to the combination of small fragments to design strong ligands which would, otherwise, be scattered into the set of possible molecules. This technique is widely used in research for discovering novel strong inhibitors: its result can be compared with *high-throughput screening*(HTS), even if the latter cannot be replaced completely.

Stochastic Methods

The idea is to use statistical sampling to obtain an approximate result for a problem with a difficult analytical solution. Stochastic methods scan only partially the conformational and rotational space by performing random changes in the ligand's degree of freedom. As a consequence, a convergence to the optimal solution is not guaranteed: to minimize this problem, several independent executions of stochastic algorithms are usually performed. Moreover, due to its intrinsic randomness, this strategy is likely to avoid local minima results and increase the probability of finding a global minimum.
Several algorithms are available, but the widely used are:

Monte Carlo algorithm This method[28][8] differs from other simulation approaches since the parameters of the model are handled as random variables, rather than fixed values. The distribution of these stochastic variables must be defined a-priori. The simulation is carried out by repeating the model, each time drawing a different set of stochastic values from the sampling distribution of the parameters. Accordingly, this method creates an initial configuration of ligand in the binding site based on some predefined probability, which is evaluated by some specific criteria (i.e. scoring the free energy). Then, small rotational and/or conformational changes are provided to the ligand and a new state is discovered. This new configuration is scored with the same criteria: if the score optimizes the previous one, then the current state is evaluated further, otherwise it can be rejected or accepted based on the Metropolis criterion. Monte Carlo simulation docks the ligand inside the receptor active site trying numerous random positions and rotations which limits the chances of being trapped in a local minimum. The main issue is that the greater the number of replications, the longer the run-time of the entire simulation and the heavier its computational cost.

Genetic algorithm It is inspired by the principle of population and biological evolution. A certain arrangement of the ligand is described by a set of parameters concerning its translation, rotation and conformation. These parameters represent the *state variables* of the system. Moreover, a *fitness function* is required to evaluate the solution domain: in a molecular docking problem, the energy of interaction of the ligand-protein pair is regarded as the *fitness value*. The process starts by generating a random population which is evaluated by the fitness function. A portion of the population is selected to find a new population in such a way that individual solutions with higher fitness values are more likely to be selected. The new population is generated through the combination of genetic operators, called *crossover* and *mutation*. From the selected portion, a pair of "parents" produces a "child" solution that shares many features of its "parents". This process keeps

going until a new fixed-size population of solutions is produced, and the whole procedure is repeated again. In general, the new population contains better fitness values because the best individuals from the first population are selected, together with a portion of less fit elements. The termination happens when a certain condition is reached: a solution is found, a maximum number of cycles are performed or the fittest solutions cannot be improved furthermore.

Tabù Search It represents a heuristic, used for hard optimization problems, which exploits *local search* methods. The latter is employed when the goal is to find a solution that maximizes/minimizes a criterion among several possible candidates; the algorithm moves from solution to solution by applying small changes until an "optimal" solution is found. Tabu search is an iterative procedure that uses Local Search to move from one potential solution x to a better one x', in the neighborhood of x. The search terminates when a certain threshold is reached or some stopping criterion occurs. Given the combinatorial space we are taking into account, the method starts with an initial feasible solution, then small changes are applied to the current conformation and evaluated by a fitness function to check if the solution is optimized. If no move leads to an improvement, then the worsening moves can be accepted. Furthermore, the algorithm tries to avoid previously-checked solutions marking them a *tabù*. In order to "remember" marked solutions, it relies upon particular *memory-structures* which differ from the temporal range considered:

- Short-term: only recent solutions are avoided.
- Intermediate-term: the prohibitions are enlarged to quite a long period.
- Long-term: lots of rules are memorized to drive the search into new regions.

The set of memory-structures form the *tabù list*, a collection of banned solutions used to detect which is the next move. Given their nature, Local search methods, as well as Tabù search approaches, are prone to stick in local minima regions: in case of hard problems in which short-term memory structures are not enough to explore the neighborhood space, intermediate and long-term memories can be adopted to jump off local minima.

Deterministic Methods

Deterministic models are mathematically described by a system of ordinary differential equations where no randomness is assumed to be present. However, determinism makes the model not take into account uncertainty, which instead is what biochemical processes are subject to[35].
In this kind of method, the orientation, conformation and translation of the molecules are

determined by the previous state, thus the result is strongly dependent on the initial input structure: given the same molecule configuration and a set of parameters, the output will always be the same [30][9]. The high computational cost and the propensity to get stuck to local minimum solutions are the major drawbacks of these algorithms.

Molecular Dynamics The Molecular Dynamics method aims to simulate the movement of a system in a specific time frame considering some thermodynamic state variable. The motion of interactive particles in a system is described by using approximations based on Newtonian physics. The forces acting on the system are determined by molecular interaction potentials, generally considering contributions derived from bonded and non-bonded atoms[61]. Subsequently, the position of the atoms is determined by the integration of Newton's second law of motion[65]:

$$F_i = m_i \frac{d^2 r_i}{dt^2}$$

This process is repeated many times such that a trajectory and temporal evolution of a ligand-based receptor complex can be examined.
With this approach, all the degrees of freedom of both ligand and protein can be considered during the simulation, but it implies a high computational cost which prohibits long running experiments, especially when protein-ligand complex is under study since it involves thousands of atoms. Also, molecular dynamics methods suffer the issue of being trapped into local minima. Despite its limitation, MD can lead to significant contributions, especially when combined with other techniques and methods.

Simulated Annealing It is a probabilistic technique used in optimization problems for approximating the global optimum of a cost function, and is mainly indicated when finding an approximate global optimum is better than a precise local optimum in a fixed amount of time. Thus, simulated annealing should be used for very hard computational problems, where even exact algorithms fail.
Several variables define the state of the system which is brought from an initial configuration to a state where the (free) energy is close to the global minimum. Starting from a state s, the algorithm considers some neighboring state s^*, and with a certain probability it decides to jump to the neighbor state or not. The neighbors are represented by small changes on the current state s based on modifications of conformation, orientation and translation of the ligand. The probability of making the move from s to the next state s^* depends on an *acceptance probability function* $P(e, e_{new}, T)$ which depends on the energy $e = E(s)$ of the current state, the energy $e^* = E(s^*)$ of the next state and on a global

parameter T named *temperature*.

Search algorithms provide a feasible orientation of the compound in the binding site, however this doesn't mean that the ligand actually binds to the protein. Thus, docking should be seen as a generator of good hypotheses on how a compound may bind in the binding site of the target, but not as proof.
Even if it is not a common practice in virtual screening due to its additional computational cost, also the flexibility of the protein can be accounted during the process in an approach called *induced-fit* docking.

Name	**Search**	**Algorithm**	**Features**
Glide	Systematic	Descriptor-matching/Monte Carlo	Flexible Docking
Dock	Systematic	Fragment-based	Flexible Docking
FlexX	Systematic	Fragment-based	Flexible-rigid Docking
HammerHead	Systematic	Fragment-based	Flexible-rigid Docking
LigandFit	Systematic	Fragment-based	Flexible-rigid Docking
AutoDock	Stochastic	Lamarkian Genetic Algorithm	Flexible-rigid Docking
Gold	Stochastic	Genetic Algorithm	Flexible Docking
MolDock	Stochastic	Differential Evolution	Flexible-rigid Docking
ICM	Stochastic	Monte Carlo	Flexible Docking
ZDock	Deterministic	Molecular Dynamic	Rigid Docking
RDock	Deterministic-Stochastic	Genetic-Algorithm/Monte Carlo	Rigid-flexible Docking

Table 1.1: Software for Molecular Docking simulations [21][22]

1.4.3. Scoring Function

Molecules interact with each other through different forces that follow the laws of thermodynamics, such as hydrogen bonding or hydrophobic interactions and van der Waals forces, and determine a binding energy, given by the binding constant K_d and the Gibbs free energy ΔG_L [23]. The latter is a thermodynamic potential that measures the reversible work to be performed by a thermodynamic system at a constant temperature and pressure. As a consequence, a protein-ligand binding is spontaneous only if the difference in ΔG_L between the unbound and bound state (ΔG^0)[42] is negative, thus lower energy scores represent better protein-ligand binding with respect to higher energy values, and can be expressed by the following formula:

$$\Delta G^0 = \Delta H - T\Delta S = -k_b T \ln(C^0 K_d)$$

where:

- T is the temperature
- ΔH is the enthalpy of the system
- ΔS is the entropy of the system
- k_b is the Boltzmann constant
- C^0 is a reference concentration of 1 mol/L

The Gibbs free energy is a state function means that it depends only on the initial and final state, which represent respectively the energy of the protein and the ligand when separated and the energy of the complex.

The docking quality is measured by a scoring function which is used to evaluate the energetically best ligand conformation when bound to the target. A scoring function is a mathematical predictive model that estimates the binding energy and hence the stability of the complex. The model tries to emulate as much as possible the real forces that occur, considering different thermodynamics and chemical parameters. The more variables are taken into account, the more precise will be the score at the cost of a bigger effort during the computation. In general, a trade-off between accuracy and speed should be reached such that the function doesn't represent a bottleneck for the docking process, but nor a weak predictor of ligand-protein affinity. Top-ranked ligands can continue the screening process by being subjected to High-Throughput Screening in order to be synthesized.

Scoring functions take a pose as input and return a score of the interaction, ranking the ligands evaluated by their score. They are categorized in three main categories:

- Force field-based
- Empirical
- Knowledge-based

Force field-based scoring functions express the energy of the complex interaction as a sum of different bonding and non-bonding terms. The score is computed by accumulating van der Waals and electrostatic interaction energy between the protein and the ligand:

$$E_{bind} = E_{vdW} + E_{elec}$$

This category of methods doesn't take into account entropy and solvent effect, which represents big limitations in the complex evaluation and can lead to poor performances [39].

Empirical scoring functions are based on counting the number of various types of interactions such as hydrophobic and hydrophillic contacts, the number of hydrogen and rotatable bonds. This kind of functions use several inter-molecular interaction terms which are balanced with experimental data. In such a way, binding energy can be approximated by a sum of individual uncorrelated terms. In fact, each energy contribution is weighted in a linear combination of parameters, typically defined as:

$$\Delta G_{bind} = w_0 * \Delta G_{motion} + w_1 * \Delta G_{interaction} + w_2 * \Delta G_{desolvation} + w_3 * \Delta G_{configuration}$$

The weights are learned by a simple linear regression model, therefore an adequate training set with known binding affinities is needed in order to optimize the energetic factors. Since empirical scoring functions are based on a statistical model to be trained, the major drawback is their strong dependence on the accuracy of data used to learn the weights. Once the weights are assigned, the binding score calculation is much faster than the force field-based, since the energy terms of interest are simpler to compute.
Different functions exist, some of them take into account also non-bonded interactions (G-Score[31]) or hybrid approaches: for example, a *semi-empirical force-field* scoring function is used in DOCK[20], composed of Lennard-Jones potential and Amber force field.

Knowledge-based scoring functions are based on the principle that interactions between types of atoms that occur more frequently than a random distribution can actively contribute to the energy force of the binding. This structural information is extracted

from already-known protein-ligand complexes via the application of the Boltzmann Law to convert the frequent atom-pairs into pairwise potentials. [52]. The major advantage of knowledge-based scoring function is the great balance between accuracy and speed, since there is no attempt on reproducing binding affinities (empirical methods) or force field calculations. However, these kinds of functions are limited on the information currently available for experimentally determined molecular complexes.

The previously described methods are generally called **classical scoring functions** for which Table 1.2 provides a classification of the different software in the market. In recent years, other types of scoring functions were born based on artificial intelligence methods: they are named **machine-learning-based** scoring functions.

Force Field-based	Empirical	Knowledge-based
Dock	GlideScore	SMoG
AutoDock	AutoDock	PoseScore
ICM	X-Score	MotifScore
LigandFit	F_Score	PMF_Score
	LigScore	
	LUDI	

Table 1.2: Software for Scoring functions [21][22]

Once machine learning enters the game, lots of new approaches based on the different kinds of models are created, and also in molecular docking new techniques are adopted. A crucial requirement that determines the success of machine-learning-based scoring function is a good training set to train the model and meaningful descriptors (features) to express the data. A noteworthy model that shows great results in scoring docked poses is the **Random-Forest** algorithm (RF), which is based on a bagging ensemble of decision trees.

It's interesting to evaluate the performance of scoring functions, independently of conformational search algorithms, to highlight their strengths and weaknesses. The comparative assessment of scoring functions (CASF)[55] placed the standard measurements to evaluate scoring functions, which is based on four indicators:

- Scoring power: it measures the ability of a scoring function to produce scores in linear relationships with the experimental binding data, measured by Pearson correlation coefficient.
- Ranking power: it refers to the ability of correctly ranking the known ligands of a target protein by the binding affinities when the precise binding pose of the ligands are given. It doesn't require a linear correlation between binding scores and experimental binding data.
- Docking power: it refers to the ability of the scoring function to identify the native ligand binding pose among the computer-generated poses. An ideal scoring function would rank the native conformation as the best one.
- Screening power: it corresponds to the ability of a scoring function to distinguish binder from non-binder ligands for a given receptor.

CASF[55] shows that **X-Score** (and its derived versions) represents the best classical scoring function for what concern *scoring power* and *ranking power*, however it is not the optimal choice for *docking power* and *screening power*.
Due to its ability to produce scores correlated to experimental data and to rank ligands correctly based on that score, we will consider X-Score for our simulation, as noted in Chapter 4.

2 | High Performance Computing

2.1. Introduction

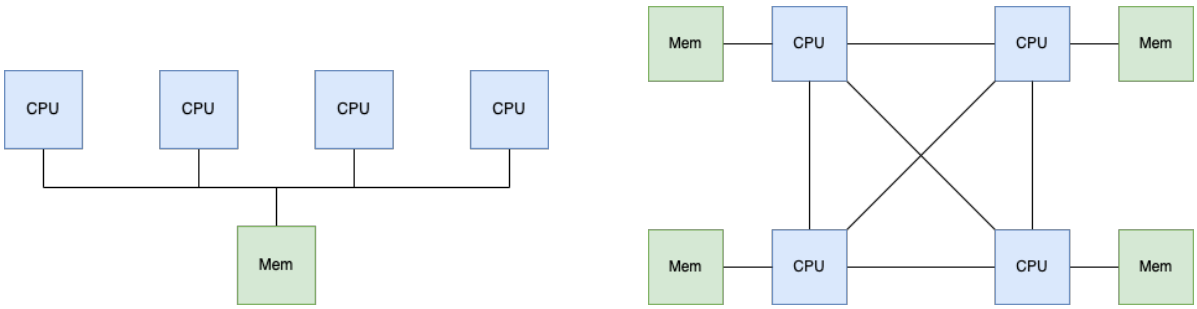
Early studies on molecular docking were performed by keeping a static receptor structure where rigid ligands were docked. At that time, only a few protein structures have been properly defined and the global process was quite inefficient: the rigidity of both the protein and the ligands yielded inaccurate results, in addition to the fact that the maximum number of molecules that could be docked in a reasonable time was around a hundred. Since the 1990s, incremental improvements in the power and efficiency of large-scale systems have made possible the processing of over a billion compounds in a few days, so that *high-throughput molecular docking* has become the new de-facto standard.
Even if multiple ligands can be processed in a reasonable amount of time in a single CPU core, a large-scale screening application must be performed to gain some insight about the activity of a protein.
The available datasets usually comprise millions of compounds to be tested against one or multiple receptors; since molecular docking serves as an initial screening step, the time spent on the simulation compared to the number of processed ligands should be at least equal to in-vitro *high-throughput-screening (HTS)*.
Generally, the performance of the virtual screening phase depends on different factors such as the algorithm chosen, the size and the complexity of the dataset or the width of three-dimensional structure of the protein. However, parallel execution of the workload always decreases by several orders of magnitudes the running time, splitting the data processing among multiple CPU cores available within a super-computer or a cluster.
Nowadays the use of massive parallel super-computing programs to quickly sample the configurational space has been accentuated by the Covid-19[3] pandemic, looking for promising candidates to help the discovery of new drugs. To reach this goal, *high-performance-computing* strategies have been deployed, leveraging the use of super-computing systems composed of several computer-nodes, each consisting of dozens of cores. Thanks to this kind of organization, it is feasible to perform a large number of operations, parallelizable among the nodes.

High-Performance Computing[66] programs are built upon a complex and powerful computer infrastructure, typically composed of a set of hundreds of interconnected machines. Applications must efficiently exploit the corresponding architecture to avoid any waste of resources and increase their productivity. These infrastructures are continuously kept updated with the newest and fastest components: most of the high-performance-computing simulations[48] leverage the latest GPU architectures to enable massive data parallelism in order to heavily accelerate the calculation. This up-to-date hardware increases the power of the super-computer, but leads to instabilities in existing software that developers have to overcome. The refactoring results in labor-intensive and inefficient effort and requires some level of expertise that it's not immediate to learn. For these reasons, high-level programming standards are widely adopted to reduce complications when switching to modern platforms.
Conversely, a range of architectural solutions can be adopted for large-scale systems, each one with specific features that allow achieving worthy performance.
Both the advantages, the simplicity of migration toward better architectures and platform-dependent optimizations, should be considered when the design of a high-level programming model takes place. Moreover, since the HPC market is relatively small, it is convenient to standardize the main programming models such that they are usable in different super-computing platforms.

During the years, different architectural paradigms have emerged and they can be summarized as follows:

- *Shared-memory parallel systems* (SMP): the processors exchange information by reading from and writing to physical memory, shared between all the actors. The two types of shared memory are known as *Uniform Memory Access* (UMA), where access time to a memory location is not dependent on which processor makes the request, and *Non-Uniform Memory Access* (NUMA), where the memory access time depends on the memory location relative to the processor, thus a processor can access its own local memory faster than non-local ones.

- *Distributed-memory parallel systems* (DMP): each processor has its own private memory and can operate only on its local data. Processors are organized in a network (or via point-to-point links) and the only way they can communicate with each other is through an exchange of messages. The success of these architectures highly depends on the quality of network interconnections. Another challenge to face



(a) Unified Memory Access (UMA) (b) Non-Unified Memory Access (NUMA)

Figure 2.1: Shared Memory Parallel Systems

in order to avoid idle processors during the program execution is how to distribute the data over the memories: ideally, each processor should handle an amount of data that allows all the processors to complete their work at about the same time.

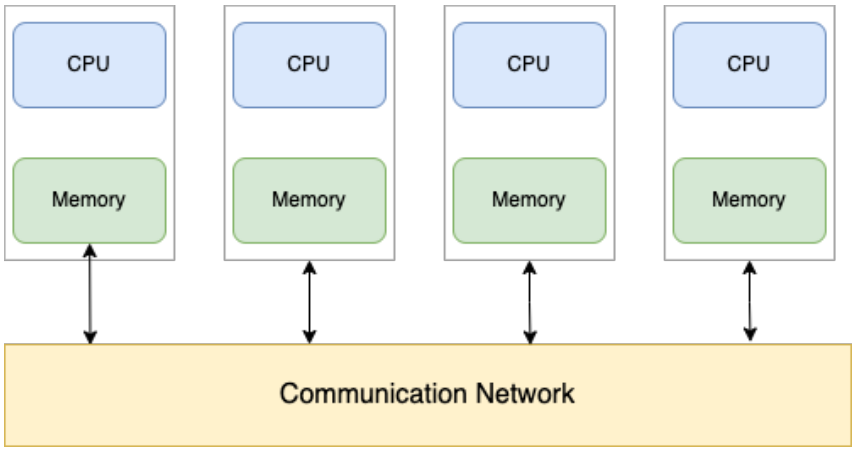


Figure 2.2: Distributed Memory Architecture

- *Vector super-computing systems*: they provide the highest level of performance for certain applications but they are expensive to purchase and operate. The architecture is more complex than the previous ones, composed of a classical scalar unit, along with a vector unit attached to it. Once the data are loaded in the main memory, the scalar control unit detects if the requested instructions are scalar or vector operations: if a scalar operation is decoded, the scalar control unit executes the instruction, otherwise it will be sent to the vector control unit. The latter implements an instruction set that operates efficiently on a large one-dimensional array of data.
- *Clusters of Workstations* (COW): it corresponds to a set of workstations connected by a LAN (local-area-network), employed to compute a single job together. This architecture is cheaper than the others since it doesn't require any custom network

or infrastructure, a feature that on the other hand could slow down the performance.

The decrease in networking costs helps the diffusion of COW architectures, which becomes popular thanks to their ease to build and their ability to scale to thousands of processors, such as those deployed in laboratories and industries.
Instead, SMP systems exploit the parallelism through a *multi-threading* programming model, where multiple threads sharing the same memory are spawned to execute one or more operations. Different implementations of multi-threading executions exist such as POSIX Threads and OpenMP. The main benefit of shared-memory programming resides in its simplicity, due to a global accessible memory and a "light-weight" parallel execution (with threads) that allow increasing the throughput. However, the shared memory cuts both ways and leads this paradigm not to scale well in massive parallel computations: besides the exponential growth in the production cost of an SMP with lots of processors, increasing the number of threads accessing the memory produces congestion in the data traffic.
The only way to work around this issue and provide a scalable system to perform massive parallelizable tasks is to adopt a distributed-memory architecture. In fact, in DMPs there is potentially no limit to the scalability of the system: several hundred thousand CPUs can be incorporated. The maximum range of expansion is usually dictated by the network interconnections performance, which should be kept as fast as possible, and by the management of the I/O operations: the latter is an open problem that affects large HPC infrastructure where the bottleneck of intensive I/O applications limits the performance of the entire system.
Today, the scalability of DMP and the simplicity of SMP technologies are sometime combined to provide two distinct levels of architectural parallelism. Thus, *distributed shared memory* allows addressing physically separated memories as a single shared address space and provides an interface that hides messages passing to the programmer, making the programs more portable, at the cost of slower accesses to the (distributed-shared) memory and additional protection against simultaneous accesses to shared data. To keep away from errors in data due to the synchronization mechanism, programmers have to be aware of the consistency models to maintain the memory coherence.

With the advent of parallel and distributed computation, developers had to create alternative methodologies to write their own software in such environments, abandoning the traditional sequential programming where a single process executes operations one after the other. As opposed, parallel programming lets the developer run multiple processes concurrently, exploiting as many resources as possible to speed up the calculation. How-

ever, writing parallel applications efficiently is a challenging task since requires a good understanding of the platform in order to distribute evenly the computation across different CPUs and provide them the required data as fast as possible to avoid unnecessary delays: the strategy to choose which memory location should store a particular data becomes critical and could have a major impact on the performance.

2.2. Message Passing Interface

Large parallel applications usually rely upon a memory distributed among different processors, like in DMP and COW architectures. Each processor can directly access its own private memory and operate on that. In addition, different processors can communicate with each other via an explicit communication protocol that allows transferring data stored in memory from a processor to another that needs it, through the system's network. This exchange of information is typically referred to as "message passing", so the corresponding programming model is known as **Message Passing Interface**[24]. MPI is a *message-passing library interface specification*. It is a specification, and not an implementation, in the sense that it describes a standardized and portable message-passing design to be applied on parallel computing infrastructures. From that, multiple open-source implementations were created for different programming languages.
MPI was developed by a community of parallel computing vendors, computer scientists and application developers. The goal of MPI is to describe a widely used standard for writing message-passing programs, focusing on portability and ease of use. Its main purposes include:

- An efficient communication protocol
- The design of an application programming interface (API) that can be implemented on many vendor's platforms
- Ability to use in a heterogeneous environment
- Language-independent semantics

MPI consists of a set of library routines that abstract the low-level management of network communication among different nodes, showing a convenient high-level interface. An approach that well suits the message passing paradigm is the so-called SPMD, that is *single-instruction, multiple-data*, where each processor, identified by an MPI identification number, runs the same code with different data. As well as SPMD, also *multiple-instructions, multiple-data* (MIMD) approaches are relatively easy to implement.

The library includes several features and routines aimed to facilitate the development of scalable and efficient parallel systems. A few of them will be briefly described in the next subsections.

2.2.1. Point-to-Point Communication

The basic operation in MPI involves the exchange of messages between a pair of processors: *send* and *receive* are the methods in charge to fulfil this mechanism. A simple procedure can be implemented as follows:

```
#include "mpi.h"

int main(int argc, char *argv[]){
    char message[20];
    int rank;  // MPI processor identifier
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0) {
        strcpy(message,"Hello, World");
        // Processor 0 send a message
        MPI_Send(message, strlen(message)+1, MPI_CHAR, 1, 99, MPI_COMM_WORLD);
    }
    else if (rank == 1) {
        // Processor 1 receive the message
        MPI_Recv(message, 20, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
        printf("received :%s:\n", message);
    }
    MPI_Finalize();
    return 0;
}
```

The *send* method called by Processor 0 contains different information regarding the message itself, the destination processor and additional information used by *receive* operation to select a particular message. *Send* and *receive* methods used in the example are *blocking* operations, i.e. the processors are blocked until the communication is finished.
Non-blocking operations can be considered as well, which result in the invocation of two different methods: `MPI_ISend()` and `MPI_IReceive()`. These functions return immediately even if the communication has not yet completed.

Point-to-point communication guarantees that the **order** of a sequence of messages to the same receiver is always respected, that is the receiver cannot receive the i-th message before the $(i-1)$-th one.

2.2.2. Datatypes

Besides basic data types natively supported, MPI allows extending data classes through the definition of custom (or derived) types. This is particularly useful when the user wants to enrich the messages with particular objects or structs that encapsulate a collection of information. This flexibility makes feasible the support of heterogeneous environments where types might be represented differently on different nodes. The general mechanism should be able to transfer directly, without copy, objects of various shapes and sizes, by providing to MPI a definition of the data structure involved in the message passing.
A general datatype is composed by:

- A sequence of basic data types
- A sequence of integer (byte) displacements

The displacement array is required for data structure alignment. Passing a data structure as unique block is much faster and more convenient than transferring an item at a time, especially in terms of network traffic.

2.2.3. Collective Communication

The *Collective Communication* encloses all those functionalities that involve a communication within a group or groups of processors, no more a point-to-point interaction. Some of the routines start from or end to a single node, which is called *root*. Moreover, a communicator, in charge of defining the group, is needed when collective communication takes place.
The type-matching condition is stricter than in point-to-point communication, i.e. the amount of data sent must be equal to the amount of data specified in the receiver.
A collective operation usually belongs to one of the following categories:

- All-to-One: an operation is executed by all the processors in the group, but only one receives the final outcome. Includes:
  - *Gather* method: it collects data from all the members of a group to one member
  - *Reduce* method: from all the members of a group, data is first collected and then a particular operation (like sum, max or min) is applied to the collection.

The result of the reduction is stored in only one processor.

- One-to-All: only one processor computes the result and transfers it to the other nodes. Includes:
  - *Broadcast* method: a single message is spread from one processor to all the others
  - *Scatter* method: as opposed to gather, scatter operation is similar to broadcast operation where data is sent from one member to all the others
- All-to-All: all processors contribute to the result and receive it. The results could not be unique, instead each node computes and shares its output with the others such that eventually each node contains a collection of results. Includes:
  - *All Gather, All Reduce, All Scatter* methods: such as their corresponding method, but with the difference that the result is sent to all the members of a group.
  - *Barrier* method: it serves to synchronize all the processors at a certain point during the execution.

2.2.4. I/O

Scientific applications often need to deal with a very large amount of data that are usually processed and analyzed efficiently in HPC systems. If not properly handled, I/O operations could result in a bottleneck for the performance due to the increasing number of disk accesses that makes the access latency no more negligible. HPC systems face this issue on the hardware side with a parallel file-system that acts as a support for high-level libraries able to manage parallel accesses to the files. In MPI, the parallel I/O ability is informally called MPI-IO and it corresponds to the set of routines able to abstract I/O management in a distributed environment.
The set of specified routines comprehend basic I/O operations for parallel data access that differ from each other by three main properties:

- Positioning: each processor keeps two pointers referring to a certain file. A *private* file pointer can be used by a single processor to access the file seeking a particular position, whereas a *shared* file pointer is shared among all the members in a group and can be moved by any processor.
- Synchronization: also for I/O operations, a blocking I/O routine does not return until the request is completed. On the contrary, a non-blocking I/O routine starts

an I/O operation but does not wait for the entire execution.

- Coordination: data accesses can involve a single processor or a group to coordinate the reads and writes operations.

More complex features are available to handle cases in which the distribution of the data is not-contiguous in memory, allowing customizable ways of *viewing* a file.

2.3. HPC-as-a-Service

The role of HPC has evolved from a technology reserved for the academic research community to be a crucial pillar for several areas of science. In the era of Big Data and Artificial Intelligence, most of the programs rely upon data-intensive utilization needed to provide insights or train huge machine learning models.
Massive computations are crucial in several other fields[14] where high capacity, precision and speed are fundamental requirements. For instance, *Weather Science and Climatology* have shown a rapid growth in analyzing complex data and chaotic processes. The claim for better climate models aims to address important scientific challenges, such as climate sensitivity, and to handle uncertainty quantification in prediction models. In fact, in weather forecasting or oceanography, the quality of the solutions is strongly related to the spatial resolution of the simulation: the wider area taken into account, the better the result. Nevertheless, enabling kilometer-scale resolution, ranging from tens to hundreds of kilometers, implies a substantial increase in computing power in order to drive more complex and longer experiments.
Same computational challenges occur in *Energy* applications to face both researches on new renewable sources to reduce the environmental impact, and techniques to improve the efficiency of current energy sources. Energy HPC techniques collect a broad range of methods to improve wind turbines, higher photovoltaic efficiency or material properties optimizations, all aspects with a potential immense social and commercial impact. The oil and gas industry, for instance, is pursuing progress in monitoring seismic activity through the addition of high frequencies data acquisition, more sensors and sources. This information is then interpreted by physical model approximations combined with artificial intelligence deep learning methodologies to handle a huge amount of data in heterogeneous formats and to increase the accuracy of classification and segmentation problems, along with temporal analysis exploration.
Last but not least, *Life Science and Healthcare* are reaching rapidly lots of interest in science through the collaboration between different scientific figures, such as chemists, doctors and biologists, with highly specialized computer engineers. In particular, as largely

described in Chapter 1, bio-molecular studies via virtual screening approaches have become one of the most competitive topics, able to gather a vast community of university researchers and top-level industries with the purpose of discovering new solutions to speed up the drug discovery process and improve its precision. The result of such studies could reduce exponentially the cost of pharmaceutical production while increasing the availability of new medications in the market. However, virtual screening pipelines, and related methods, require an increasing amount of molecular data to be effective: the higher the number of molecules investigated, the higher will be the probability of finding new drugs. Therefore, storing and analyzing petabytes of data in a reasonable time can be achieved through innovative bioinformatics approaches supported by an intensive and distributed computation. It can be easily deducted that all these above-mentioned scientific areas, and many others, have in common an extreme and growing necessity of systems able to scale in both computational power and storage. In fact, the importance of data-centric technologies noted in almost all recent scientific researches leads to difficulties in storing, analyzing and distributing a huge mass of information: as the quality and the number of heterogeneous data and measurements increase, the space required to save them increases as well. Therefore, dealing with such a load locally becomes no more feasible for universities and industries without incurring in serious hardware limitations.

On the other hand, the processing of these datasets cannot be performed without spreading the computations in a distributed environment across thousands of cores. Additionally, many of the analyses require optimized hardware components for training or visualization purposes, which leverage integrated accelerators such as GPUs (Graphic Processing Unit) or TPUs (Tensor Processing Unit).

High-Performance-Computing centers help to tackle these challenges by providing large-scale infrastructures for high-performance machines. A new paradigm of HPC, known as **HPC-as-a-Service**, is intended to provide high-level processing capacity to the customers through the cloud. Thus, HPC centers facilitate the access to specialized resources to perform complex calculations and to work with petabytes of data, avoiding the user the cost of new equipment and the need for on-premise clusters.

Due to its low investment costs, HPCaaS turned out to be the preferred choice for most of the data-intensive applications[27][12] that require an HPC support, thanks to its ease of deployment and customizable scalability.

In this way, the work can be efficiently executed in a multi-node distributed fashion with Message-Passing-Interface standards and parallelizable using multiple accelerators sharing memory. Data can entirely be stored in large secondary solid state storage to handle petabytes of data and extreme I/O procedures.

The increasing demand for computational resources in so many areas brings to several

advances in HPC systems such that applications, which previously were based on petascale computational power, can now exploit exascale architectures and improve the performance by some order of magnitude. *Petascale* and *Exascale* denote the calculation capacity of a super-computer; in particular, the switch from one to the other corresponds to an increment in the performance from 10^{15} (1 PETA) to 10^{18} (1 exa) floating-point operations per second (FLOPS). The latter is extremely hard to reach and it was attained for the first time in April 2020, with the distributed computing network Folding@home [1].

2.3.1. IT4Innovations

IT4Innovation National Supercomputing Center is an HPC center at VSB, Technical University of Ostrava, which operates the most powerful supercomputing systems in the Czech Republic and is one of the top in Europe. Its resources are provided to Czech and foreign research teams from both academia and industry and represent the state-of-the-art in super-computing technologies.
IT4Innovations research activity is mainly addressed to:

- Big Data processing and analysis:
- Machine Learning and Virtual Reality
- Parallel scalable algorithms
- Energy problems
- Advanced visualization
- Modelling of nano-technologies
- Material design

Currently, IT4Innovations Center is composed of three super-computers. *Karolina* is the latest and the most powerful super-computer cluster in IT4I, installed in summer 2021 and with a peak performance of 15.7 PFLOP/s. *Barbora* is the second most recent cluster, installed in the autumn of 2019 and with a peak performance of 848 TFLOP/s. The last cluster is a highly specialized very powerful system for artificial intelligence and machine learning computation and it will not be considered further. Instead, until the end of 2021, another cluster named *Salomon* was available with a theoretical peak of performance of 2 PFLOP/s. Their main features are summarized in Table 2.1.

The storage for each cluster is organized into two main shared filesystems, named HOME

Feature	Karolina	Barbora	Salomon
General			
No. Nodes	829	201	1009
No. Cores	106,752	7,232	24,192
RAM	313 TB	44544 GB	129 TB
Peak Performance	15.7 PFLOP/s	848 TFLOP/s	2 PFLOP/s
Accelerators			
Accelerated Nodes	72	8	432
Accelerators	8 x NVIDIA A100 (40 GB HBM2)	NVIDIA Tesla V100-SXM2	2 x Intel Xeon Phi 7120P, 61 cores, 16 GB RAM
Network			
Compute Network	InfiniBand HDR	InfiniBand HDR	InfiniBand FDR56 / 7D Enhanced hypercube
Storage			
HOME	31TB (25GB/u)	28TB (25GB/u)	500TB
SCRATCH	1000TB (20TB/u)	310TB (10TB/u)	1.69 PB
RAMDISK	Not-specified	180GB	110GB

Table 2.1: IT4Innovation Clusters

filesystem and SCRATCH filesystem. The HOME filesystem contains users' home directories and it is intended for the preparation, evaluation, processing and storage of data generated by active projects. It corresponds to a high-availability cluster of a pair of active-passive NFS servers such that a replication of the filesystem is always available and can be restored in case of failure. The total capacity is 31 TB, shared among all users, restricted at 25 GB per user.
The SCRATCH filesystem is realized as a parallel Lustre filesystem and it guarantees high-performance access to input and output files. It is accessible via the Infiniband network and supports a maximum capacity of 1000 TB, shared among all users, restricted to 20 TB per user. The SCRATCH filesystem is aimed to store temporary scratch data generated during the calculation. All I/O intensive jobs must use this storage as the working directory.
In addition, every computational node is equipped with file system realized in memory, so-called RAM disk. Since it loads the data directly in memory, this storage should be used to save small temporary scratch files for high-performance access. A further data store available only on Karolina and Barbora is named PROJECT and it represents a central storage for all the projects on IT4Innovation. The PROJECT is accessible from all IT4Innovations clusters and allows sharing global data among clusters.

Computational resources on each cluster are handled by a workload manager called *PBS Pro*. PBS is a widely acquired software designed to improve productivity and optimize utilization in clusters-based architectures. It scales to support millions of cores with fast job dispatch and minimal latency. A suitable job request should specify the following fields:

1. A queue for the job
2. Number of nodes required
3. Number of cores per node
4. Maximum wall time for the computation
5. Your project ID
6. A job script (Template)

When a job is submitted to one of the clusters, the request is sent to the PBS Job Scheduler which is in charge of allocating the required resources. In this phase, the job is initially added to a queue that collects all the jobs submitted to the cluster, but not yet executed. This allows PBS to optimize the scheduling and allocate users' jobs in a fair-

share fashion, ensuring that individual users consume approximately an equal amount of resources. Once in the queue, the job is suspended waiting for the nodes to be available. After the resources are correctly allocated, the job script is executed and it will last at most the maximum wall time specified in the job request. The job script is written in bash and it describes a sequence of instructions for loading modules and libraries, executing the calculation and storing the results.

3 | Recommender System

With the transition between the static Web 1.0 to the user-centric Web 2.0, it's become increasingly important to be able to offer a personalized experience to the users; for this reason, the last two decades have seen a rise in the Recommender Systems (RS) field, both from an industry and an academic point of view.

Recommender Systems (often abbreviated in RS or RecySys) are one of the most popular trends in the recent years and, even if these techniques have been around for decades, it's only in the past few years that they've become one of the most flourishes and discussed field of the Data Mining and Machine Learning world. There are several reasons behind the success of these methods, just to name a few:

- The transition from Web 1.0 to Web 2.0: the focus of the online world shifted from static web pages to dynamic, personalized and user-centric platforms like Amazon, Netflix or Spotify.
- The amount of data available online increased at an incredibly fast rate such that tons of information can be stored in specific RS datasets.
- The innovation in the hardware sector, which has fueled the whole area of Machine Learning, making possible things unimaginable until a few years before.

In this scenario, it's become of key importance for companies to be able to offer a personalized experience to the users to let them navigate and discover the best possible contents within the hugeness of data available on the web: e-commerce websites, like Amazon, offer users personalized suggestions for products that they may like, music streaming services like Spotify recommend to each user songs that are close to their taste and the same thing is done by Netflix and YouTube with visual content. All these major companies put a lot of effort in improving the best experience for users through advanced RS techniques, since providing a personalized aid turns out to be beneficial both for the content providers, that aim at maintaining users on their platform for as long as possible, and for the users, since good personalized recommendations let them discover new contents that they may like.

Recommender Systems can be defined as a collection of software techniques, derived from

the Data Mining and Machine Learning fields, that aim at providing "item" suggestions to "users". Even if these methods are always discussed in the context of these Web 2.0 platforms that we just mentioned, we can have various definitions of “item" and “user" depending on the domain. Let’s see a couple of examples:

- In a music streaming platform like Spotify, the users are the actual users of the platform and the items correspond to the songs available on the platform. The objective of the RS methods will be to understand the taste of the users based on the past interactions they had to be able to recommend new songs.
- In a trip booking website like Tripadvisor, the users are the platform users and the items are of various types: restaurants, hotels, attractions and so on. The goal of such websites is to let the users discover possible locations of interest based on past trips or general reviews on the items.
- In a social network like Facebook (Meta), there is no real distinction between users and items: in fact, the users and the items are exactly the same and the goal of a RS, in this case, is to suggest to the users other users they may know based on the (friendship) connections that the user has at that point in time.

As it should be clear, there is no unique definition for the terms users and items, to remain very general we could define the users as those entities that are capable of taking some action in an environment whilst the items are exactly the entities the users interact with (as we just saw from the previous examples, users and items are generally disjoint sets, but this is not always true as in the case of a social network in which users and items coincide). Users and items are the actors of a Recommender Systems, but the real key character of this domain is the “interaction", in fact we just defined the items and the users as those entities that “interact" with each other. Interaction is a very broad term that can have several different meanings depending on the domain of application and this perfectly fits the fact that, since pretty much any complex system can be seen as a bunch of entities interacting with each other, the theory and the models proper of the Recommender Systems field can be applied to a wide variety of domains without loss of generality. The goal of RSs is to let the users discover new possible interactions with items that they may like with a reasonable probability.
In this thesis, we use Recommender Systems algorithms to model the interactions between proteins and molecules, in order to predict the interactions with an high affinity score. For the rest of this chapter we will give a broad overview on the basic concepts of the Recommender Systems theory.

3.1. Data Structures

In order to deal with a problem using the Recommender Systems theory, we firstly need to be able to identify some elements that are common to pretty much every RS:

- **Users**: a set of users U.
- **Items**: a set of items I.
- **Ratings**: a set of ratings S, that are the scores assigned by users to the interactions with the items, for example a 1 to 5 stars review on Amazon.
- **User features**: set of attributes proper of the users F_u.
- **Item features**: set of properties of the items F_i.

To highlight the behavior of those elements, this information is converted to different matrices. In the following paragraphs, we see these data structures more in detail to understand how to prepare the data to apply the most popular and discussed Recommender Systems techniques.

3.1.1. User Rating Matrix

The User Rating Matrix (URM) is the key data structure used for storing and manipulating the interaction data of a Recommender System, it is a $|U| \times |I|$ shaped matrix that captures the historical interactions between user and items.
Each cell (u, i) of the URM contains a numeric value r_{ui} that represents the score of the interaction between user u and item i. This numeric value is different depending on the domain, but, in general, there are the two kinds of data sets that one can encounter when dealing with a RS: **explicit** and **implicit** data sets.

Explicit Feedback

Explicit Feedback settings are the ones in which the user directly expresses a preference value over the interaction with an item, usually using a pre-defined scale. This is the case of e-commerce websites like Amazon or Tripadvisor, which let users rate items on a 1 to 5 star scale and also let them leave a review in the form of a short piece of text.
In this situation, in which the data gathered is a direct measurement of the preference of the users, we are not only able to observe if an interaction occurred, but also to understand whether it was a positive or a negative one. In explicit feedback scenarios, we can then discern between the following three levels: positive, negative and missing interactions.

Implicit Feedback

Even if the usual RS examples always involve some sort of explicit feedback system, it should be clear that the vast majority of the interaction that happen online is not explicit at all: even on platforms like Amazon, which give users the possibility to directly rate items, only a very small percentage of the users is willing to rate items so that the actions that are by far the most frequent are views, clicks and actions such as adding an item to the cart, which are totally implicit. Implicit feedback indirectly reflects opinions by observing user behaviors such as the purchase history, browsing history, search patterns, clicks and even mouse movements [38, 47].
In other words, implicit feedback scenarios can be defined as those settings in which it is possible to observe if a user interacted with an item but, unlike explicit scenarios, there is no way to understand if the user liked the item so that the ratings of the URM will be just ones or zeros.
As already said, implicit feedback is definitely more common with respect to the explicit one and it should be noted that implicit settings are also substantially trickier since there are only two levels that we can distinguish (that's why implicit feedback is also named binary feedback): positive and missing data (which is actually a mixture of missing and negative feedback).

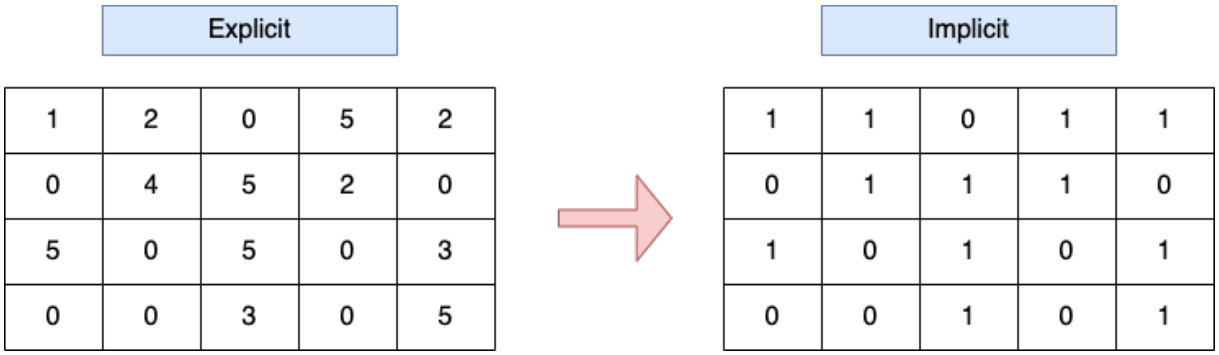


Figure 3.1: Explicit to Implicit URM conversion

Figure 3.1 shows two URMs, one is explicit and the other one is implicit; more precisely, the implicit one is obtained from the explicit one by taking as positive feedback only those interactions with a rating greater or equal than 3 (this process of deriving the implicit form of the URM from the explicit one is called *implicitization*; note that the reverse is not possible).

3.1.2. Item Content Matrix

In most domains, in addition to the interaction data, items and users usually present some observable and measurable properties, called *features*. The **Item Content Matrix** (ICM) is the data structure that serves the purpose of storing item properties, specifically it is a $|I| \times |F_i|$ shaped matrix that in each row contains the features F_i of an item i, i.e. those characteristics, highly dependent on the application domain, that define the item's properties, which can take on different values and forms i.e. floating numbers, integers, binary values, enumerators, unstructured pieces of text etc.
For example, in a movie platform like Netflix the movies are the items and the item features could be properties like the movie genre, the duration, the year or the title.

3.1.3. User Content Matrix

The **User Content Matrix** is the counterpart of the ICM for users, as it is a $|U| \times |F_u|$ shaped matrix storing the user features that describe additional properties regarding the users. As for the item features, these attributes can take on different values and thus need some sort of preprocessing steps and it's worth noting that they are more rare with respect to the item features as it's always more difficult to be able to gather data about users rather than items.
In the previous example of a movie platform, some user features could be the user's birthday, location, gender or preferred language.

3.2. RS Models

After presenting the classical data structures that are used to model a Recommender Systems problem, we are now going to briefly introduce the main classes of algorithms that are used to predict the users' future preferences with respect to items. Three main categories of models can be identified:

- **Collaborative Filtering Techniques**, which exploit the interaction data in the URM.
- **Content Based Techniques**, which use the item and user features.
- **Ensembles**, hybrid approaches that combine collaborative filtering and content based techniques to improve the overall performance of the model.

3.2.1. Collaborative Filtering Techniques

Collaborative filtering (CF) methods try to take advantage of the “collaborative" power of users and items, starting from a simple yet very powerful assumption: since the observed interactions are often correlated, for every user we can then use the ratings given by other like-minded users in order to infer the missing ones [34]. In simpler words, this means assuming that if two users agree on some item i then they will probably agree on some other item j. These techniques can be divided into *user-based* and *item-based* approaches depending on whether they apply the collaborative assumptions to users or items.

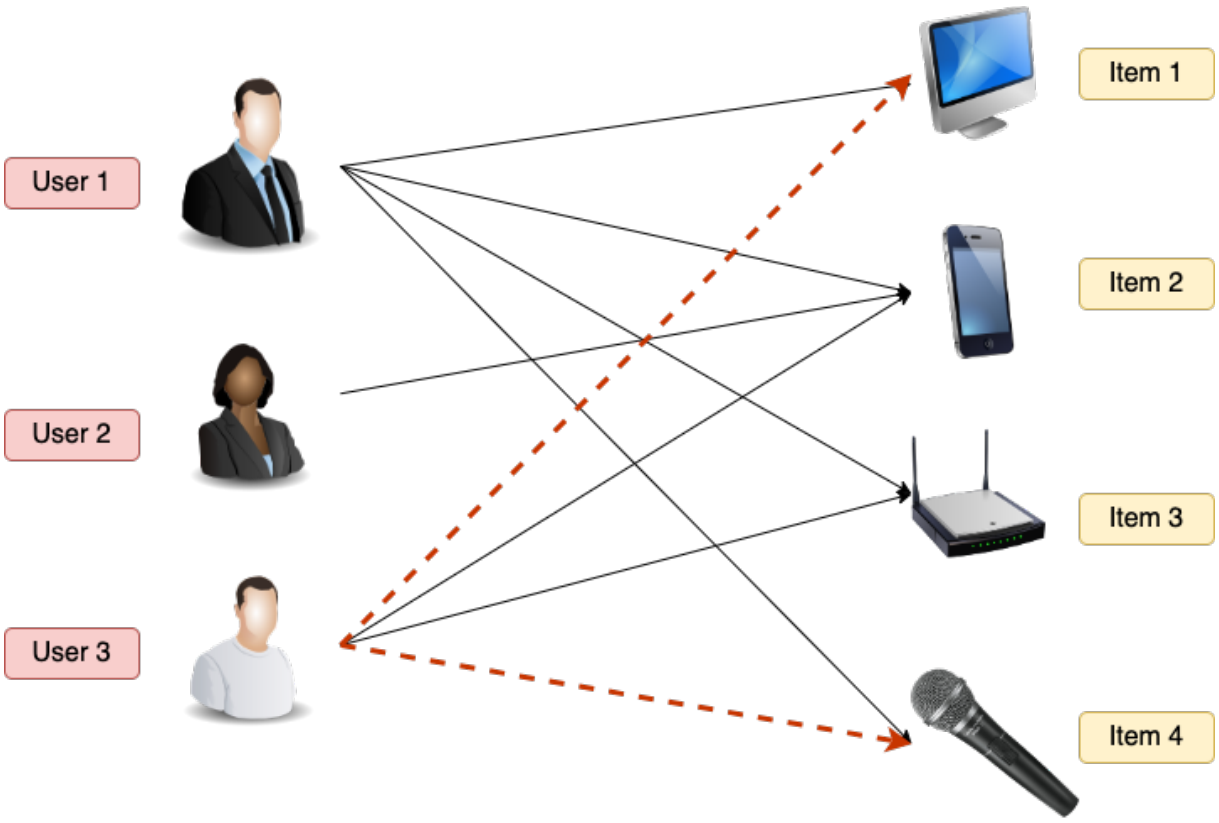


Figure 3.2: User Interactions

For example, let’s consider an e-commerce website like Amazon in which users are able to navigate and purchase items. In Figure 3.2 we can see an example scenario that helps at better visualizing the concept of collaborative filtering, considering three users:

- user 1 purchased all the items;
- user 2 purchased only item 2;
- user 3 purchased items 2 and 3.

With the collaborative assumptions of the ratings, a collaborative filtering approach (more precisely a user based CF) would then suggest items 1 and 4 to user 3 because user 3 has a similar purchase history as user 1, which purchased those two items.
From this example we can see that collaborative filtering methods only make use of the rating matrix (i.e. the historical interactions between users and items) in order to compute the future predictions; this means that, in contrast to content-based approaches, they are totally domain-independent and can yet discover hidden aspects of the relationships among users and items.
There are two main families of algorithms in which we can divide the CF techniques: **memory based CF** and **model based CF**.

Memory Based CF

Memory Based approaches often use some heuristic in order to compute the predictions. The classical techniques consist in defining a **similarity** measure or a custom metric between users or items and compute these metrics using all the historical observed interactions to recommend to each user the most likely items according to the computed similarities. Since the metric is dependent on the interactions, every time that new data is added to the training set the similarities need to be computed from scratch. There is no training phase for such kinds of models, thus the ratings of the URM can be used directly in the predictions.
Some popular examples of Memory Based CF algorithms are ItemKNN, UserKNN or, simply, the Top Popular. These methods are among the simplest to implement and yet often produce very good results even compared with much more complicated algorithms [16].

Model Based CF

This class of methods is intrinsically different than memory-based algorithms, especially in the way the available data is used to extract predictions. In fact, instead of relying on the whole dataset to compute recommendations (*memory-based*), they exploit this information to build a recommender model. Therefore, a *training* phase is needed to give the model the ability to learn from previous interactions. The model can be interpreted as a parametric function that takes as input the URM and/or ICM and uses them to adjust the weights of a machine learning algorithm in order to estimate ratings. Unlike *memory-based* techniques whose computational complexity grows with both the number of users and items, *model-based* approaches typically summarize the data in a compressed representation that allows to scale the dimension of the input dataset and to provide fast

predictions.
Different models have been developed for Model-based CF, most of them inspired by machine learning techniques such as regression, latent-factor, naive Bayes and artificial neural networks.

3.2.2. Content-Based Techniques

Content-based techniques include another set of recommendation algorithms in which additional information is used to compute predictions. Besides the user-item ratings (URM), a description of individual items and users is provided to the context. The description usually refers to a set of properties, broadly known as *features* or *attributes*, which is employed to build the *Item-Content-Matrix* (ICM), in case of items' features, and the *User-Content-Matrix* (UCM), in case of users' features. However, in several problems users' features are not available or hard to be extracted, so it is not rare to take into account just the ICM.
The idea behind content-based approaches rely on recommending items similar to those a user interacted with, supported by the assumption that user's preferences remain unchanged during the time: a user, that expressed a preference for an item, will probably like similar items. As a consequence, a pairwise measure of how much an item (user) looks like another one must be computed, which generally implies the construction of a **similarity** matrix.
Figure 3.3 shows an example of how a Content-based algorithm recommends a movie to a user. Assume a user watched *Gran Torino*, a drama movie directed by Clint Eastwood; since this movie has the same director (Clint Eastwood) as *Million Dollar Baby* and both are categorized into the genre of drama, then we can presume the user will probably like *Million Dollar Baby* too and the model recommends it.
As the reader may have noticed, content-based recommenders are user-specific, i.e. they focus only on the user profile they are evaluating, thus no information about other users' interactions is taken into account to provide predictions.

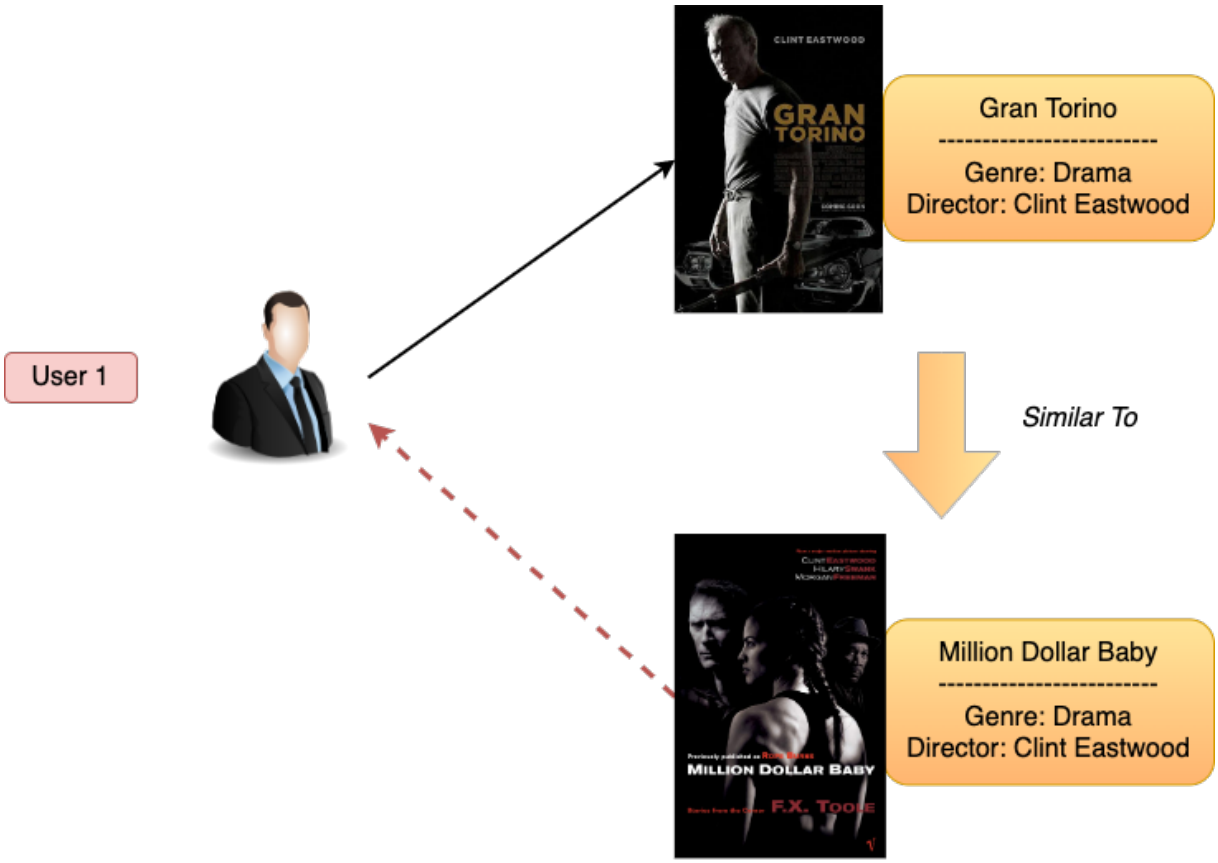


Figure 3.3: Content-based Example

4 | Contribution

Most of the state-of-the-art methodologies discussed in Chapter 1aim at discovering new molecular complexes, studying the attractive forces that could relate a macro-molecule with a much smaller one. Several software has been proposed in literature to exhaustively process a large database of compounds against a single protein. However, a common drawback that emerges regards the elevated cost required to perform the screening analysis and employ the appropriate resources. Moreover, unlike some techniques that leverage the properties of the ligands, the current approaches do not exploit any sort of similar behavior among the proteins, but instead each of them is treated independently. Our solution tries to avoid a systematic execution of the whole chemical library promoting, instead, a small subset of interesting compounds.
In the remainder of this chapter, we explain our approach in Section 4.1, whereas a more detailed discussion about the implementation is provided in Section 4.2, which describes how data for our research is generated thanks to the collaboration with IT4Innovations' Supercomputer, and in Section 4.3, which explains the recommendation methodologies used in this analysis.

4.1. Proposed Approach

The main goal of this work is to prioritize the evaluation of certain molecules that are more likely to fit the protein binding-site with a high score, exploiting the set of protein-ligand pairs already analyzed. In such a way, it should be possible to minimize the number of stages required to discover all the promising molecules as well as the cost of the screening, deferring to a later evaluation those ligands that are predicted to have a low interaction score. This purpose has been achieved through the application of a recommendation engine in a chemical scenario, able to analyze the impact of classical models in the prediction of the most promising compounds. In particular, we try to simulate a multi-stage screening quest in which each stage corresponds to the elaboration of a prefixed number of ligands: this can be compared to a real-life scenario, where High-Throughput-Screening experiments are performed on a bulk of molecules at a time, considering the unmanage-

able size of the chemical space.
The Recommendation Platform proposed consists of a *data-centric* application that collects the results of previously computed docking pipelines to lighten the effort for the next protein evaluations. The system focuses on filtering out a wide dataset of small molecules in order to select only the best candidates that bind a particular protein (Figure 4.8). Once the evaluation of such candidates is complete, the information regarding their binding affinity is added to the current knowledge to increase the performance of the models in predicting the next candidates.

Thus, given a budget of computational effort, we limit the further experimentation on just a small portion of the chemical space, preferring a precise set of promising molecules rather than a random selection of candidates. As a consequence, a deep analysis can be performed on just a subset of the compounds, reducing drastically the costs and the computational time of execution.

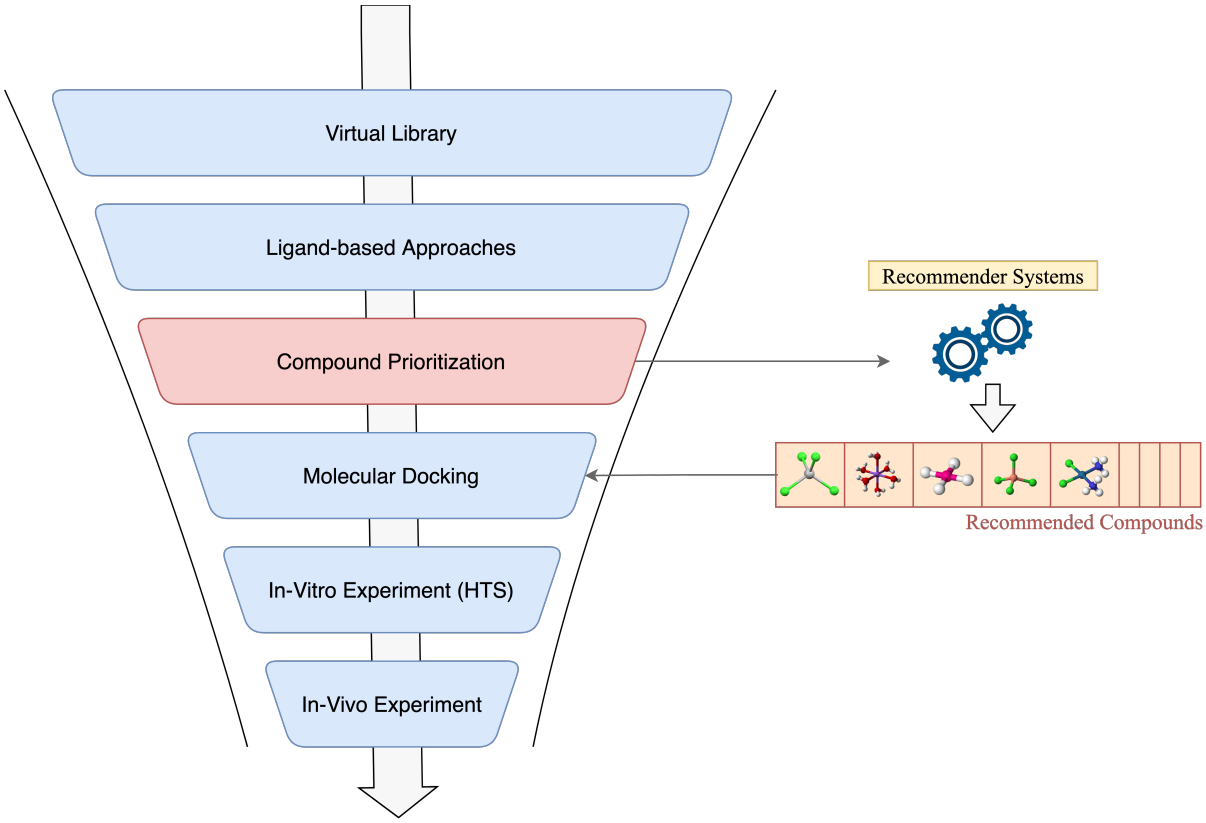


Figure 4.1: RS Compound Prioritization

4.2. Data Generation

Since no dataset is currently available to continue further our analysis, it was necessary to gather the data on our side, applying a molecular docking pipeline to a set of proteins and ligands. As stated, this kind of screening requires an intensive effort in computation, which is hardly obtainable without a high-performance architecture. Our research is primarily based on the optimization of a Molecular Docking pipeline modeled on a super-computing system, able to scale up a screening execution of millions of ligands.
In the next sections, the reader is introduced to the datasets considered and their format. Then, the discussion is moved toward the main components involved in the docking and scoring phase, and, finally, we show how the integration with IT4Innovations' super-computer has been effectively deployed.

4.2.1. Dataset

Because of the very large number of chemical structural properties, a global and complete description of the molecular characteristics is extremely hard to be embodied in a singular representation. Several formats[17] have been proposed on the market, both proprietary or open to the community, each encoding different information in a different manner. There is no one single format that is ideal, but instead many of them are used in different contexts, and they can often be converted from one to the other for easier access or sharing. During the data preparation is important to carefully consider format choices compatible with the software used for the docking purpose.
In our research, the input data is dominated by two main actors participating in the screening phase, namely a *ligand* and a *protein*. Since the information needed to encode a compound is limited, ligands are usually collected in large sets and then evaluated against a single protein, which, instead, requires a more complex characterization.

Ligand

For what concerns the ligand, one of the most common formats, readable by almost all cheminformatics software applications, is the **MDL Molfile**, i.e. a text-based chemical file format for holding information about the molecule. In particular, the *Tripos Mol2* standard is a complete, portable representation of a SYBYL molecule, a specification for unambiguously describing the structure of chemical molecules using short ASCII strings. The file format splits the description into different parts:

1. A Title line specifying the name of the molecule
2. A Timestamp
3. A Comment line
4. A Counts line: definition of the number of atoms, bonds, 3D objects, and Sgroups.
5. An Atom block: each line indicates the coordinates of an atom in the space, followed by a description of the atom itself (atomic symbol, mass difference, charge, stereochemistry, associated hydrogens)
6. Bond block: each line refers to a single bond, which states the atoms connected by the bond, its type and its topology.
7. Properties block: additional properties of the molecule. It is reserved for future expandability
8. END line

The representation encodes atoms, bonds, connectivity and coordinates of a molecule and contains all the information needed to reconstruct the SYBYL version.

In most data banks, different molecules are grouped together by means of some common property, either the molecular weight, number of atoms, rotamers or a combination of them. We took into account multiple databases of ligands that differ from each other by the number of atoms and the number of rotatable bonds. In particular, we considered molecules having a number of atoms in the set [20, 25, 30, 35, 40, 45, 50] along with a number of rotatable bonds equal to 1, 4 or 8. The total amount of molecules examined was about 8.5 million, not equally distributed among the datasets.

Protein

Protein datasets, instead, typically support a more complex data representation in order to properly traduce the three-dimensional structure into a textual file. As well as in most of the researches, the input format is the one proposed by the **Protein Data Bank**[7], which is a standard for mapping atomic coordinates. A PDB file format contains several lines of information, each one called *record*; generally, a file consists of different types of records, aggregated in a specific order to describe a structure:

1. ATOM: three-dimensional coordinates (x,y,z) for atoms in standard residues (amino acids and nucleic acids)

2. HETATM: three-dimensional coordinates (x,y,z) for atoms in non-standard residues
3. TER: it defines the end of chain of residues
4. HELIX: it highlights the location of helices
5. SHEET: it defines sheet substructures
6. SSBOND: it identifies disulfide bonds

The PDB archive is a repository of atomic coordinates and other information describing proteins and other macro-molecules. This information is gathered by biologists using X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy to determine the location of each atom relative to each other in the molecule.

In our simulation, in order to evaluate the ligand bindings, we consider 39 different proteins derived from the RCSB Protein Data Bank. A precise list is provided in Table 4.1.

PDB Proteins			
1a30	1w4o	3s8o	NSP12palm
1jyq	2yge	NSP6	3su5
2vw5	NSP16	Nprot	NSP3
1yc1	3su2	NSP14	3su3
3lka	1u1b	4djr	SPIKEACE
NSP13allo	3cyx	3nq3	3ozt
NSP12ortho	NSP13ortho	3ov1	3f17
3gy4	1ctr	1uto	3oe5
2d1o	3ehy	1sln	1o3f
2yki	NSP9	3CL	

Table 4.1: Set of proteins extracted by the Protein Data Bank (PDB)

4.2.2. Docking and Scoring

Molecular docking strategies aim to analyze and predict to which extent a small molecule interacts with its receptor. The strength of the interaction is defined by the displacement in the space of the ligand-protein pair. To accomplish this task, a docking pipeline is usually composed of a two-step procedure, involving a *search algorithm* and a *scoring*

function.

The searching algorithm examines different conformations and orientations of the ligand with respect to the protein and selects the most promising poses. The protein is interpreted as a rigid body, while the position of the ligand can be adapted. In most of the available software, a *Tripos Mol2* molecule description is served to a searching algorithm along with a file specifying the location of the binding-sites inside the three-dimensional structure of the receptor. To face the huge number of possible solutions, various approaches and algorithms have been deployed over the years, each of them characterized by the policy used to sample the search space and the method employed to evaluate a pose.

After that, a scoring function evaluates the best poses that come from the previous stage and assigns a score to each of them. The difference among the scoring functions resides in the typology and the descriptors used to judge the affinity between the molecule and the receptor.

The results of this pipeline, depicted in Figure 4.2, heavily depend on the algorithms adopted for both the docking and scoring phases.

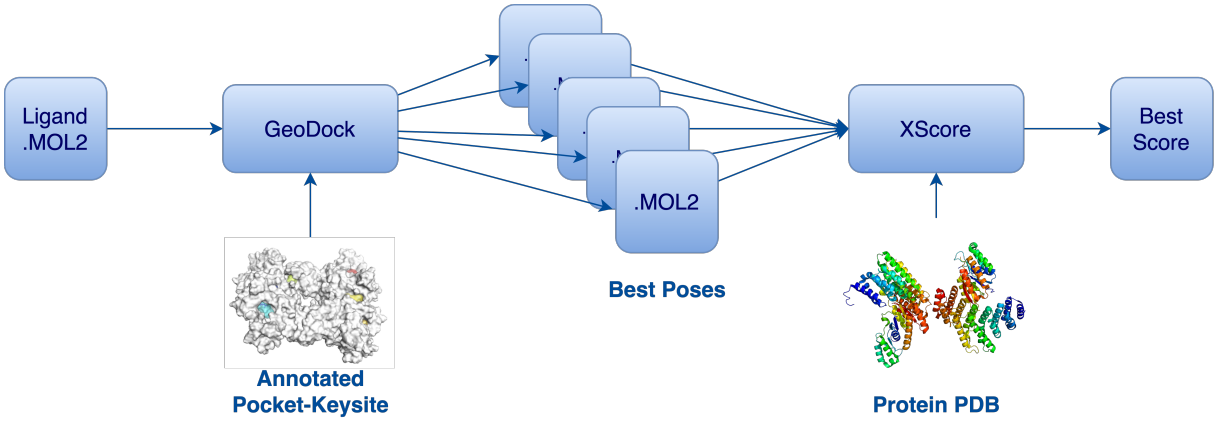


Figure 4.2: Docking and Scoring

In the following subsections, a description of the algorithms chosen for our purpose will be provided; in particular, we selected **GeoDock** as searching algorithm and **X-Score** as scoring function.

GeoDock

The conformational search is focused on a particular module of the *LiGenDock*[6] docking application, which originally centers the screening on a two-level approach: initially, considering only geometrical features, it filters out those ligands that cannot fit the target

pocket. Subsequently, the remaining molecules are simulated through physical and chemical interactions seeking the best estimation of their reciprocal three-dimensional pose. In this work, the attention is addressed to the *geometrical* part of LiGenDock, exploiting a mini-app called **GeoDock**[26]. The latter provides a tunable and optimized version of the LiGenDock module using only geometrical features such as the position of the ligand atoms with respect to the binding-site and the shape of the ligand fragments. In particular, it explores the space using ligand roto-translations and fragment rotations operators. Whereas the target receptor is kept as a rigid structure (as well as its pockets), the ligand flexibility is taken into account such that the evaluation of the different poses inspects all its degrees of freedom.

Listing 4.1: GeoDock pseudo-code

```
pocket = load_pocket()
ligand = load_ligand()
for (pose_id = 0; pose_id < N; pose_id++) {
    generate_starting_pose(pose_id, ligand);
    align_ligand(ligand, pocket)
    for (rep = 0; rep < num_repetitions; rep++){
        optimize_pose(ligand, pocket)
    }
}
```

Listing 4.1 shows the pseudo-code of GeoDock algorithm. It uses a greedy heuristic based on gradient descent with multiple restarts. The overall procedure can be summarized in three main steps[60]:

1. Generation of the starting pose: to generate the starting pose, GeoDock randomly sample the conformational space following a uniform distribution, considering only the most important fragments

2. Alignment: both the ligand and the protein are considered as rigid bodies. In this step, the goal is to find the best orientation of the ligand that fit the protein pocket, evaluated with an in-house empirical scoring function called at every rotation.

3. Optimization: once a rigid fit was found, it tries to optimize the displacement of its atoms inside the pocket, sequentially rotating each fragment of the ligand. This step is repeated at most `num_repetitions` times to refine the shape. The fundamental idea of GeoDock optimization is that is preferable to spend more time when the computation is more promising. In particular, for each bond, the left and the right

fragments are rotated independently gradually increasing the angle up to 360°. At each step, the validity of the ligand shape is taken into account: if valid, an *overlap score* is computed to check for possible improvements. The overlap score is the reciprocal of the minimum square distance between the ligand and the pocket:

$$o = \frac{l}{\sum_{i=0}^{l} \min_{j=0}^{p} d^2(L[i], P[j])}$$

where l is the number of atoms in the ligand L, p is the number of 3D points in the pocket P and d^2 represents the squared distance between the i-th atom of the ligand and the j-th point of the pocket. The overlap function is the most-expensive operation, hence shapes that are likely not to lead to any improvement should be avoided.

Different starting poses are sampled such that the entire procedure is evaluated multiple times. To explore as many shapes as possible, this algorithm favors aggressive approximations instead of precise matching. Accordingly, the result is likely to be a good approximated solution with the advantage of being available in a reasonable time.

X-Score

X-Score[63] is an empirical scoring function, developed by Dr. Renxiao Wang in Dr. Shaomeng Wang's group at the Department of Internal Medicine, University of Michigan Medical School, which combines terms accounting for hydrogen bonding, deformation effect, hydrophobic effect and van der Waals interactions. Ideally, it can be stated as:

$$\Delta G_{\text{binding}} = \Delta G_{vdW} + \Delta G_{H-\text{ bond}} + \Delta G_{\text{hydrophobic}} + \Delta G_{\text{rotor}} + \Delta G_0$$

In particular, the software gives the user the possibility to calibrate the final score through the sum of three sub-functions that differ from each other for the hydrophobic effect term $\Delta G_{\text{hydrophobic}}$:

$$\begin{aligned} \text{HP\_Score} \quad = \quad & C_{0,1} - C_{\text{VDW},1} * (\text{VDW}) + C_{\text{HB},1} * (\text{H-Bond}) + \\ & + C_{\text{HP}} * (\text{Hydrophobic Pair}) - C_{\text{RT},1} * (\text{Rotor}) \end{aligned} \tag{4.1}$$

$$\begin{aligned} \text{HM\_Score} \quad = \quad & C_{0,2} - C_{\text{VDW},2} * (\text{VDW}) + C_{\text{HB},2} * (\text{H-Bond}) + \\ & + C_{\text{HM}} * (\text{Hydrophobic Match}) - C_{\text{RT},2} * (\text{Rotor}) \end{aligned} \tag{4.2}$$

$$\begin{aligned} \text{HS\_Score} \quad = \quad & C_{0,3} - C_{\text{VDW},3} * (\text{VDW}) + C_{\text{HB},3} * (\text{H-Bond}) + \\ & + C_{\text{HS}} * (\text{Hydrophobic surface}) - C_{\text{RT},3} * (\text{Rotor}) \end{aligned} \tag{4.3}$$

where the weights (corresponding to C_*) are suitably tuned.
The user is allowed to choose any combination of these function, by default they are averaged to obtain the final score:

$$\text{X-Score} = (\text{HP\_Score} + \text{HM\_Score} + \text{HS\_Score})/3 \tag{4.4}$$

X-Score framework doesn't provide any intrinsic mechanism to perform the conformational search, hence it is usually combined with other software like DOCK, or GOLD in structural-based drug design.

4.2.3. HEAppE

As previously stated, the amount of information we have to process involves roughly 8.5 million ligands and 39 proteins. The docking pipeline must be run for each ligand-protein pair as in a cartesian product, for a total of about 330 million combinations. Analyzing all of them is CPU-expensive and time-consuming, a computation that a single machine cannot handle.
Instead, we have to leverage on a High-Performance-Computing infrastructure able to efficiently manage the workload. In particular, our simulation was run on *Salomon*, one of the IT4Innovations National Supercomputing Center's clusters, as described in Chapter 2.3.1.

Although *PBS*, the IT4Innovation's Job Manager (Section 2.3.1), efficiently distributes the workload across the supercomputer nodes, an inexpert user could find intricate the management of his own submitted jobs: he should access the remote cluster via the command-line interface, define the parameters, submit the job via PBS and use other PBS commands to query the status of his jobs.
To relieve the users from any additional duty, IT4Innovations has developed an application framework called **HEAppE** (Figure 4.3) that provides simple and intuitive access to a super-computing infrastructure.
HEAppE, which stands for **High-End Application Execution** Middleware, promotes the easiness of use as an *HPC-as-a-Service* platform via an object-oriented client-server interface.

This framework simplifies the customer experience through a set of utility functions to

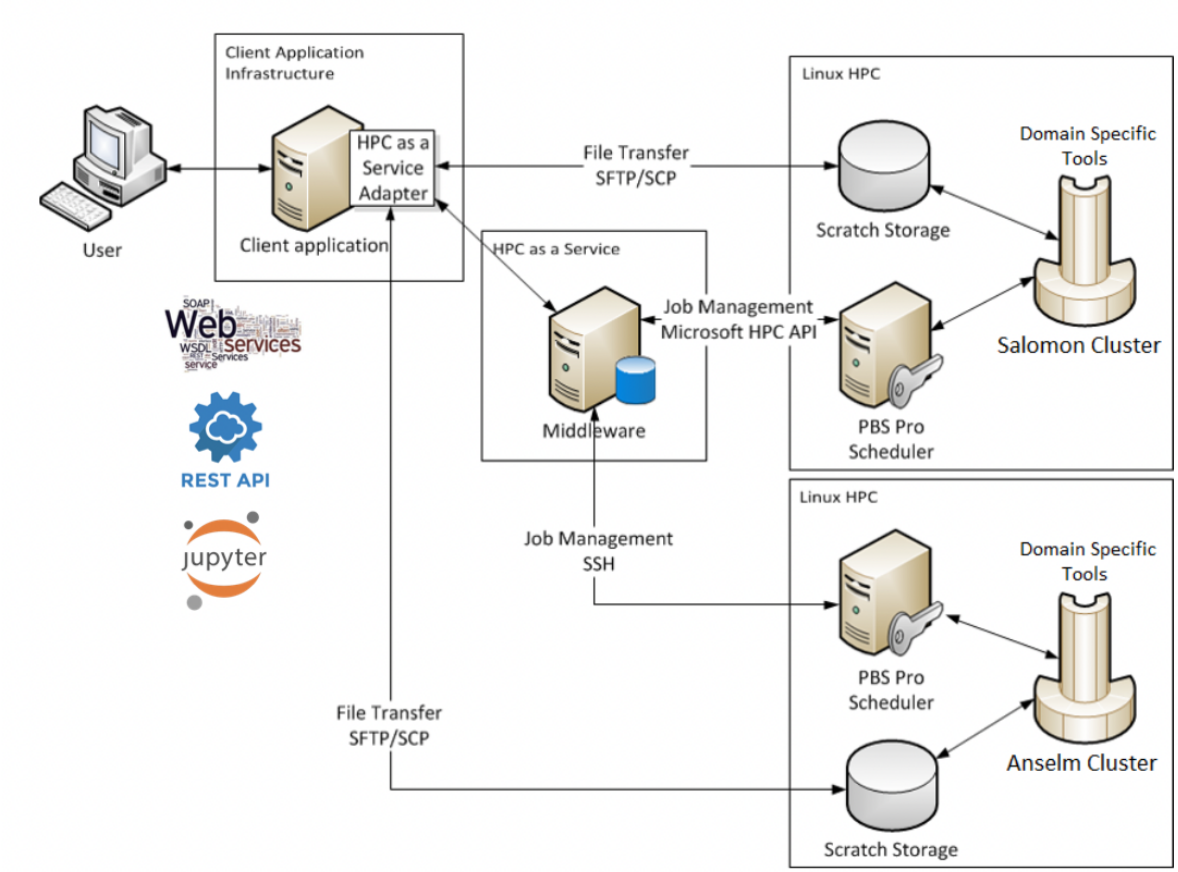


Figure 4.3: HEAppE Framework [10]

handle different operations:

- Job Submission, Management, Monitoring and Reporting
- Execution of Pre-defined Models
- File Transfer
- Resource Access and Limitations
- Notification mechanisms
- Authentication and Authorization

Since it doesn't require any particular type of hardware, it can operate in existing high-performance and future exascale computing systems and, currently, it is adopted by multiple super-computing centers.
In order to submit jobs, a *Command Template* should be created with the definition of the script or executable to be run along with a set of parameters defining the job.
An authenticated user can interrogate the infrastructure via the client application inter-

face using standard web services, REST API or Jupyter Notebooks. All the user requests are forwarded to a middleware server, placed in the *middle* between the user and the cluster. The middleware server has the visibility of all the available clusters within the supercomputer center, thus it is able to limit the user accesses to only the authorized clusters. Once a *job* request is captured by the middleware server, it converts the request in a proper PBS operation and delegates the PBS Scheduler to fulfill the request. The data required by a job can be uploaded directly into the cluster storage via SFTP/SCP file transfer protocols, as well as the job results can be fetched. Additional functionalities depend on the cluster the user is referring to.
Several projects, where a remote access to an HPC system is crucial, incorporates successfully the HEAppE framework for distributed execution of drug discovery pipelines, DNA sequencing or image processing.

4.2.4. IT4I Simulation

The super-computing power was served (as-a-Service) through the HEAppE framework, which guaranteed a lot of flexibility in the management of the high number of jobs submitted. The submission process was made even simpler thanks to a Jupyter Notebook interface that allowed us to specify the input parameters and launch a job without any explicit SSH connection to the remote cluster. Since the ligands were spread over 19 different databases and each of them had to be evaluated against all the proteins, queuing individually all the jobs may strain the system, leading to inefficient job scheduling and overall degradation of performance for all users in the cluster. Instead, HEAppE provided us a systematic way to collect several jobs within a single command via *Job Array* operation: a job-array is a compact representation of many jobs sharing the same job-script (Command Template) and having the same resource allocation parameters (number of nodes, number of cores, priority, ..); each sub-job is recognized by a $PBS_ARRAY_INDEX and runs its own instance of the job-script.

Given the nature of the problem, each ligand-protein pair acts independently since its calculation is not correlated to any of the others. Therefore, the screening process perfectly fit a distributed environment in which an embarrassingly parallel approach allows us to split up the workload across different cores. The parallelism is achieved at a two-level basis (Figure 4.4):

- **Inter-node**: the work is balanced among different processors inside the same cluster. The nodes subdivide the tasks and communicate with each other through an

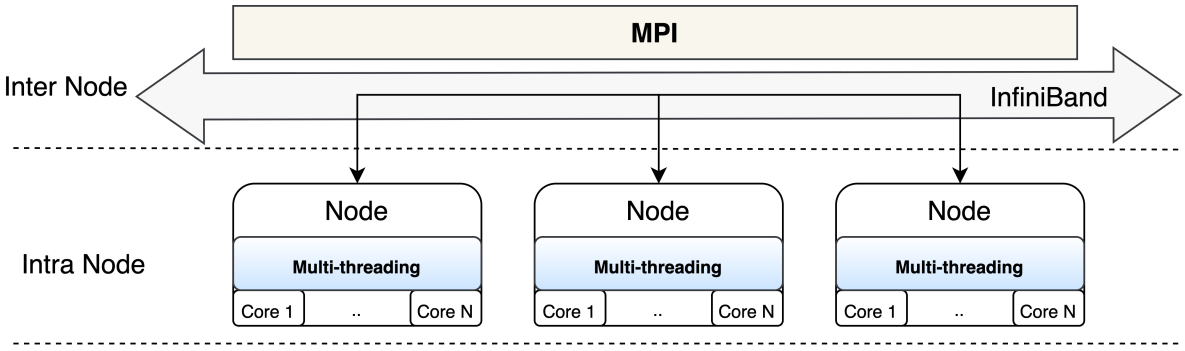


Figure 4.4: Level of Parallelism

MPI implementation called MPI4Py, which provides Python binding for the Message Passing Interface standard.

- **Intra-node**: multiple CPU-cores inside each node allow us to apply multi-threading strategies in order to increment the data concurrency. Python offers a built-in high-level module, named *threading*, that interfaces with the lower level *_thread* module. Even if the Global Interpreter Lock (GIL) limits the execution to a single thread at a time, the *threading* module becomes appropriate to run multiple I/O tasks simultaneously.

MPI

The overall process operates following a *Master-Slave* paradigm where the communication between the processors is essential to equally manage the work and collect the results.
Since the dataset is located in the SCRATCH filesystem, shared among all the machines involved in the execution, firstly the nodes elaborate an approximated position to be assigned to their *private file pointer* in order to split the data across all the participants. In each node, the initial position to start reading data is trivially computed by dividing the ligand's database file size by the number of nodes and multiplying the result by its MPI_rank:

$$position = \frac{filesize}{NUM\_PROCESSES} * MPI\_rank$$

where $MPI\_rank \in [0..NUM\_PROCESSES]$ is the MPI node identifier. After the initial position assignment, the processors check if their file pointers are addressing the starting point of a molecule description, defined in a *Tripos Mol2* file by the tag `<TRIPOS>MOLECULE`; otherwise the pointer is moved forward towards the next molecule.
This procedure allows the nodes to consider each a particular portion of the dataset, i.e. a collection of ligands, and the execution can be carried on further. Once the pointers are correctly placed, each processor activates a sub-process, called *Executor*, that evaluates

the interaction strength of every ligand with respect to the selected protein. At the end of the screening stage, the docking results are stored in an intermediate file which keeps the scoring value of the evaluated interactions. A collective communication within the group, preceded by a barrier used for waiting all the processors to complete, synchronized the accesses into a final document that gathers all the individual, partial evaluations. This last operation allows processors to efficiently parallelize the writes and conclude the distributed computing (Figure 4.5)

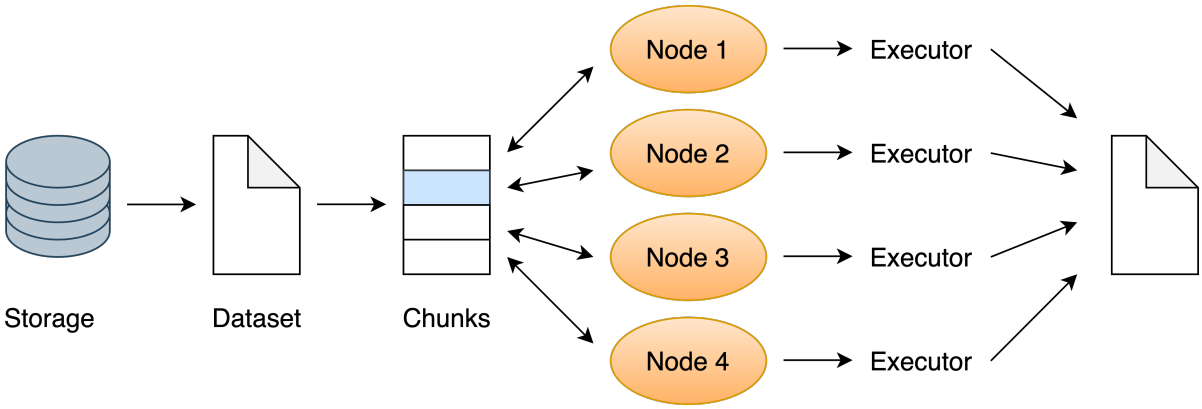


Figure 4.5: Inter-node workflow

Executor

The real core of the computation resides in the **Executor** process where the *intra*-node parallelism is efficiently exploited. The main challenge is to coordinate the operations in such a way that every core adequately serves the docking and scoring pipeline. To achieve this purpose, the work is balanced among different actors that interoperate in an execution pipeline. The communication at the core level happens through reading and writing on a shared memory data structure, commonly named **Queue**. The latter lets the information be exchanged safely between multiple threads and provides all the synchronization routines, internally managed with a locking semantics. In our work, the queues are implemented as FIFO memories. The actors can be briefly summarized as:

- Reader: it reads a ligand from the database
- Worker: it performs the docking and scoring evaluation
- Writer: it writes the worker's result in a file

In particular, they are spawned in a multi-threading environment that uses a dedicated thread for each stage of the pipeline. The overall procedure, depicted in Figure 4.6, can be interpreted as a multi-step Consumer-Producer paradigm in which the different queues act as accumulators of "messages" between two different layers.

At first, a **Reader** thread parses the input file starting from the position established by the private file pointer; as we described in Section 4.2.1 , in the chemical library the molecules are represented by the *Tripos Mol2* format, which involves a multi-line description of a single ligand. Located a molecule, the *reader* enqueues it in a *Tasks Queue* that collects the parsed molecules as single items *(1)*.
In the second step, a **pool of Workers** brings on the calculation consuming the items allocated by the *Reader*.
Since the queues are shared among the threads, multiple workers can elaborate different ligands at the same time, fetching the one currently in the head. The worker processing consists of a sequence of operations:

1. Fetching the molecule from the queue *(2)*
2. Performing the docking step to choose the best poses among the 256 generated *(3)*
3. Applying the scoring function to the best poses *(3)*
4. Write the score into an intermediate file *(4)*
5. Add the completed molecule into the *Completed Queue (5)*

Since this sequence is computationally expensive, the tasks are taken charge of by a pool of threads to optimize the performance. Moreover, processing millions of ligands implies the generation of as many intermediate files storing the scoring result.
An unexpected behavior was the deep performance degradation due to the high number of I/O operations. Originally, these documents were stored in the SCRATCH storage, which is interconnected to the cluster machines via an Infiniband network and designed to fulfill intensive I/O jobs. However, the large amount of reading and writing operations concentrated in a certain period led to the creation of millions of small files and resulted in several peaks in the I/O usage that completely block the access to the Lustre filesystem across all the submitted jobs, not only the one in consideration. To overcome the problem and widely reduce the load on the SCRATCH filesystem, the entire management of the interim results was moved into a local and private memory section of each node, called *RAMDISK*. The main advantage of this approach relies upon the removal of any communication with the Lustre filesystem as the intensive operation are kept internally on the node. On the other side, not to weigh down the in-memory storage, the intermediate files should be removed as soon as no longer useful. This work is successfully accomplished by the **Writer**.
The latter constitutes the last component of the pipeline as well as the last consumer that eventually produces the final result. In fact, the role of this thread is to check for any

processed molecule in the *Completed Queue*. If any, the molecule's temporary result file is read and its score is accumulated in an internal buffer, which reflects a CSV-like fashion composed of the ligand's SMILE representation and the relative score. Once the score is transcribed, the corresponding temporary result file is deleted. When the accumulation buffer reaches a prefixed BATCH_SIZE, the results are written into a final document so that the buffer is able to flush and restart the accumulation.
The *Executor* elaboration terminates when all the molecule's scores are collected into a unique document, which will be gathered by an MPI routine to form the general document of nodes' results.

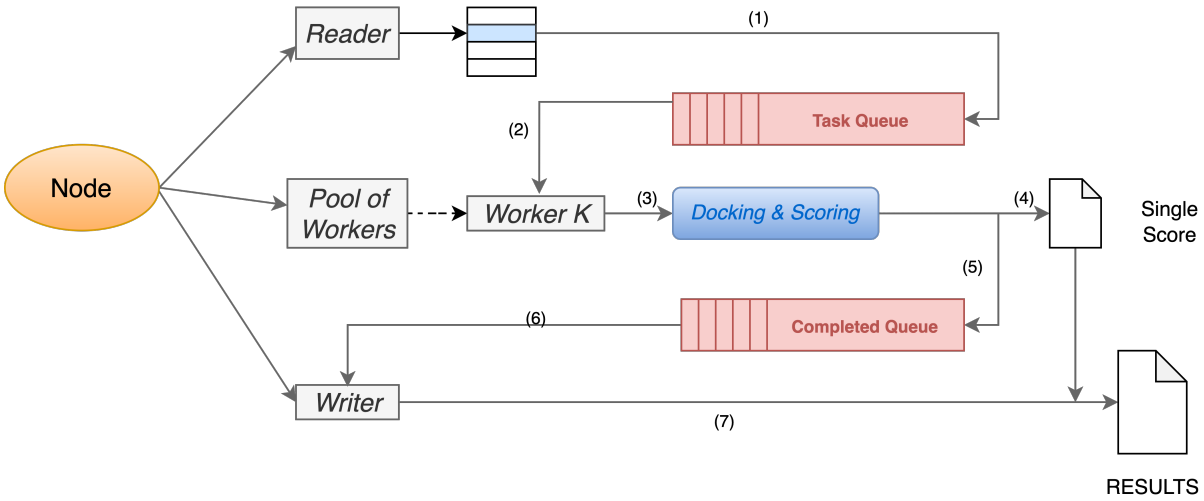


Figure 4.6: Executor Workflow

4.3. Recommendation System for Virtual Screening

Combinatorial chemistry has gained lots of interest due to its ability of generating chemical libraries composed of a wide amount of compounds. Those libraries can subsequently be integrated in virtual screening processes in order to outline some common traits of the molecules, cluster them in similar groups or evaluate the affinity by means of some docking algorithm. For the latter, there is no preference in the choice of the ligand sequence to submit such that a random selection is generally applied. However, the increasing number of molecules to be processed exacerbates the problem of detecting the most peculiar molecular complexes, which implies a subsequent increase in the time and the costs required to provide a large-scale analysis.
On the contrary, since the amount of molecules one can consider is limited by memory and timing constraints defined by the simulation, it should be interesting to accurately select

the portion of compounds that are likely to bind the target site in order to speed-up the execution time and restrict the search to just the relevant ligands. This approach reflects the goal of this thesis, where we exploit the affinity information of other proteins already processed to train different recommendation models. Given a new protein, the models are in charge of re-ordering a set of molecules in such a way that the most promising molecules appear at the top of the sequence.
In the next sections, we detail the entire process, starting from a description of the dataset (Section 4.3.1), the considered models (Section 4.3.2), and concluding with the evaluation method (Section 4.3.3).

4.3.1. Dataset

The HPC simulation widely described in Section 4.2.4aimed at establishing the binding affinity that passes between a molecule and its receptor. After the screening of a large number of compounds, the interaction scores are summarized into a dataset that represents the starting point for this analysis. In fact, a classical approach to a recommender problem involves two principal entities, broadly named *users* and *items*, whose meaning are usually shaped based on the problem we are addressing (see Chapter 3.1). In our case, the proteins take the place of *users*, while *items* are replaced by the ligands; the conjunction of both determines the *User-Rating Matrix* (URM) where the *ratings* reflect the role of the interaction scores.
Conventionally, a recommendation problem largely leverages the sparsity of the URM to handle its intrinsic high-dimensionality, based on the idea that a user hardly interacts with all the items: this property leads to better in-memory data representation and allows personalized recommendation per user.
Our case is totally opposite since the composition of all the scores into a URM results in a **complete** and **dense** matrix, therefore the sparsity property cannot be efficiently exploited. This is due to the fact that the docking pipeline returns always a non-zero value for the interconnection affinity. Unlike other researches in which the activity/non-activity is determined by a threshold that discretizes (0 or 1) the existence probability of a complex, our scores cannot undergo the same process: the score represents an indicator of the interaction strength, thus also the lowest one should be taken into account since they may refer to a less powerful attraction between the ligand and the protein, but still activity. Moreover, the distribution of the scores differs from protein to protein and comprehends values in a range larger than the probability set ([0,1]).
As a consequence, the choice of an appropriate threshold able to effectively differentiate active molecules from inactive ones is hard to be set.

Since the *implicization* process cannot be applied, the URM has to be considered as an **explicit** dense matrix where a score is assigned to each cell. In particular, the shape of the matrix corresponds to (NUM_PROTEINS, NUM_LIGANDS), i.e. for each evaluated protein, we have a collection of 8.5 million scores associated to the molecules taken into account.
Figure 4.7 shows the distribution of the scores per protein: the scores are compressed in a range bounded by the lowest score of 2.81 and the highest of 10.93. The range of values is even narrower if we consider the average score for each ligand, since the minimum and the maximum (average) scores are respectively 3.637 and 8.53, which implies that a really large amount of molecules lay in a relatively small portion of scores. Among the

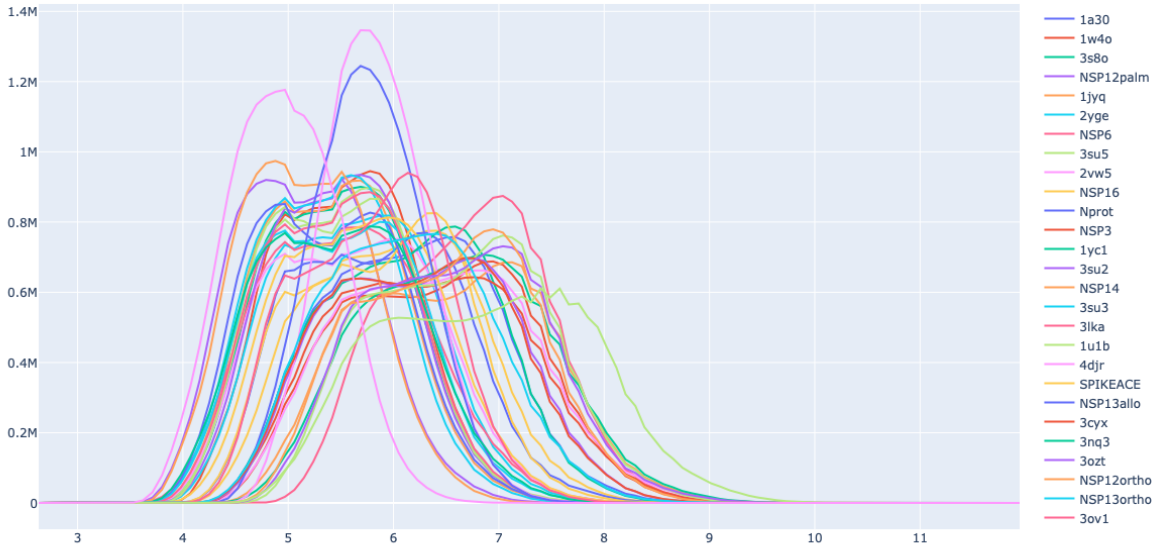


Figure 4.7: Distribution of the scores

8.5 million ligands, we designate as the most promising molecules, for each protein, the N with the highest scores: such molecules represent the **test-set** of our problem, hence the ones that we intend to discover.

Besides the matrix of interactions, we can add information to the system by exploiting a predefined set of properties that regards the molecules. These features, described in Table 4.2, are collected to build the Item-Content-Matrix (ICM), which can be used in conjunction with a Content-based recommender to extract predictions or similarity measures. On the other hand, no additional data is available to further improve protein description,

thus a corresponding User-Content-Matrix (UCM) cannot be built.

Pre-Processing

The need for the application for multiple pre-processing steps occurs in most of the data-centric decision systems in order to standardize data coming from multiple sources. Depending on the initial data, a variable-length pre-processing pipeline could include methods concerning data pruning, data cleaning, feature selection, and scaling. Besides pruning and cleaning techniques, feature selection methods and scaling operations are usually tested to increase the model performance[4] or to elicitate some insight.

In our scenario, however, data is clean by default since the data-generation stage (Section 4.2) organizes them following a precise schema structure and no multiple data sources are involved. Therefore, a simple removal of *Nan* values, which derive from erroneous *MOL2* file formatting, is needed for cleaning the dataset, while different scaling operations can be applied to try to improve the overall result.
In particular, other than evaluating the algorithms with the current data, two more scaling approaches are considered:

- *Standard Scaler*: it standardizes the values by removing the mean and scaling to unit variance. The standard value of a sample x is computed as:

  $$z = \frac{x - u}{s}$$

  where u is the mean of the training samples and s corresponds to their standard deviation.

- *Minimum-Maximum Scaler*: it transforms the values by scaling them into a precise range. For our purpose, the values are scaled between 0 and 1. The transformation is given by:

  $$z = \frac{x - min}{max - min}$$

  where x represents a sample, min and max the corresponding minimum and maximum values.

Name	Description
Molecular Weight	Weight of the ligand
Num Atoms	Number of atoms
Num RotatableBonds	No. of single non-ring bond, attached to a non-terminal, non-hydrogen atom
Num Rings	No. of cycles of atoms and bonds
Num AromaticRings	No. of hydrocarbons that contain benzene, or some other related ring structure
Num Bonds	No. of bonds between atoms
Num RingBonds	No. of simple cycle of atoms and bonds in a molecule
Num BridgeBonds	No. of bonds that acts as bridge between two molecules
Num RingFusionBonds	No. of rings having two atoms and one bond in common
Num RingAssemblies	No. of identical cyclic components linked by a bond
Num Chains	No. of chains, i.e. series of atoms of the same element
Num ChainAssemblies	No. of chain assemblies
Num Macro Chains	No. of large atomic chains
Num TerminalRotomers	No. of conformational isomers
NPlusO Count	Number of N^+O
NumSP3	No. of sp^3 orbital, i.e. when a s orbital is combined with 3 p orbital
Fsp3	Fraction of sp^3 carbon atoms
perc aromaticrings	Percentage of hydrocarbons containing benzene or related ring structure
perc heteroatoms	Percentage of atoms not carbon or hydrogen

Table 4.2: Features of molecules

4.3.2. Models

Given the dataset previously described, for each protein i and ligand j a binding-score r_{ij} is defined. The recommendations extracted by a trained model represent a set of ligands, provided to a protein, that is predicted to have the highest score. In the current state-of-the-art, ligands are evaluated without any preferential sorting, hence the molecules are associated to a uniform probability distribution that results in a random selection of them. A **Random** model does not take into account any context of the proteins as well as any historical affinity score and for its extreme simplicity, it does not require any training step.
However, leveraging the user's sessions is the core of any recommendation system since the study of the users' past behavior, as well as the items they liked, could heavily impact the quality of the predictions. Following this idea, we propose three different recommenders: Top Popular, Content-based Filtering and Collaborative-Filtering.

Top Popular

The Top-Popular algorithm is the most intuitive model which bases its recommendations on the *popularity* of each item. In classical *implicit* Recommender Systems, the popularity of an item measures the number of times the latter interacts with the users. However, since our dataset is *explicit* and *dense*, we define the popular items as those having the **highest average rating** among the users: in particular, the Top-Popular model in our scenario looks for the molecules whose average interaction scores are the largest. Mathematically, the average of the ratings of a particular item is defined in Eq. 4.5

$$\hat{r}_j = \frac{\sum_u r_{uj}}{N_j} \tag{4.5}$$

where $\hat{r}_j$ represents the predicted score for item j and N_j represents the number of proteins that rated the item j: given the completeness of our URM, N_j will be always equal to the number of proteins used for training.

Content-based Filtering

Besides the URM, some models can be fed with additional information regarding users or items. This kind of information is typically represented as a set of features associated to each element, useful to enhance their description.
In our scenario, the availability of a collection of features for the ligands enables the construction of the ICM, which can be added to the context to increase the recommendation

precision. The ICM is particularly useful to measure how much two molecules are similar, looking at their attributes. Even if not present within our data, the same approach could be undertaken to calculate the similarity between proteins building the corresponding UCM, that is the matrix representing the features of each protein.
For our purpose, we rely upon two different similarity metrics to create a similarity matrix based on the ICM:

Cosine Similarity The cosine similarity returns a measure of the angle that passes between the feature vector of item j and the one of item k. More precisely, it measures the cosine of the angle delimited by the two vectors: the more the vectors are similar, the narrower the angle and thus, the larger is the cosine:

$$s(x, y) = \frac{x * y}{||x|| * ||y||}$$

In classical problems, an additional hyper-parameter, named *shrink* S, is summed to the denominator to penalize those items with few interactions, but it is useless in a completely dense dataset.

Euclidean Similarity The Euclidean Similarity is derived directly from the *Euclidean distance*, which refers to the distance between the two vectors in the Euclidean space. In particular, it is calculated as the square root of the sum of the squared differences of each feature, which is equal to the norm-2 of the difference vector:

$$d(x, y) = ||x - y||_2$$

The corresponding similarity is computed by:

$$s(x, y) = \frac{1}{1 + d(x, y)}$$

such that the similarity is maximized when the euclidean distance is equal to 0.

The scale diversity of the features can compromise the similarity scores and lead to erroneous results: if an attribute has a bigger magnitude than another one, the outcome of operations like dot-product or norm-2 is mainly dictated by the bigger terms. To tackle this problem, the ICM attributes are scaled using the *Min-Max Scaler* in order to adjust the values between the common range [0,1].
However, the naive implementation of the similarity matrix brings to additional issues.

First, the pairwise (item-item) scores outline a similarity matrix extremely dense that becomes intractable if we consider its shape to be about (8.5M, 8.5M): such an amount of data requires petabytes of memory capacity to be stored as well as a highly intense computation. Moreover, it is noted in literature that keeping all the similarities leads to poor performance due to the noise added by the lowest scores. To face the problem, the solution is to arrange the Content-based model keeping just the K highest scores for each item, which drastically reduces the amount of information to be stored. This approach, named **K-Nearest Neighbours** (KNN), combined with *Content-based Filtering*[49] algorithm allows to estimate the rating for the protein p and the molecule i as:

$$\hat{r}_{pi} = \frac{\sum_{j \in KNN(i)} r_{pj} * s_{ji}}{\sum_{j \in KNN(i)} s_{ji}}$$

where $KNN(i)$ defines the K molecules most similar to i and s_{ji} represents the similarity score between the molecules i and j.

Collaborative Filtering

The last algorithm we proposed is the Collaborative Filtering [67], which is based on the idea that similar proteins "like" similar ligands; with this assumption, the ratings for a particular protein can be inferred by looking at the interactions of other proteins. We use a *memory-based* collaborative filtering approach where the URM is exploited to compute the similarity matrix. The latter can be expressed in both the protein-protein or ligand-ligands formulations to highlight the corresponding similarities between the proteins (user-based) or the ligands (item-based). Although the *Item-Item CF* leads to the same density and noisy solvable issues discussed before, this kind of method is highly discouraged in situations where the number of items is much higher than the number of users [51]. For this reason, our discussion is limited to the *User-User Collaborative Filtering* algorithm supported by two ranking coefficients to calculate the similarity matrix: *Spearman Correlation Coefficient* and *Kendall Rank Correlation Coefficient*. In general, a *rank* measures the relative order of a set of items, while the corresponding *rank correlation* quantifies the similarity between two rankings.

Spearman Correlation Coefficient The *Spearman Correlation* is a non-parametric index of the rank correlation and measures how well a monotonic function can describe the relationships between two ranking variables. Such as other correlation coefficients, it assumes values between the range [-1, 1] so that the higher the correlation, the more

similar are the two variables. The similarity value is computed by:

$$s(X,Y) = \frac{cov(R(X), R(X))}{\sigma_{R(X)}\sigma_{R(Y)}}$$

where $R(\cdot)$ specifies the ranks of a variable, $cov(R(x), R(y))$ is the covariance of the rank variables and σ denotes the standard deviation.

Kendall Rank Correlation Coefficient Noted also as *Kendall's τ coefficient*, the *Kendall Rank Correlation Coefficient* is a measure of the rank correlation focused on the reciprocal order of any pair of observations. Given the random variables X and Y, any pair of the joint variable (x_i, y_i) and (x_j, y_j) is *concordant* if the reciprocal ordering of the x and y observations correspond: in other words, if both $x_i < x_j$ and $y_i < y_j$ (or both are $>$), the ranks are concordant in the sense that the items i and j appears in the same relative order in both the random variables; otherwise, they are said to be *discordant*. Then, the coefficient is computed as:

$$\begin{aligned} s(X,Y) &= \frac{\text{(no. of concordant pairs)} - \text{(no. of discordant pairs)}}{\binom{n}{2}} \\ &= \frac{2}{n(n-1)} \sum_{i<j} sgn(x_i - x_j) \cdot sgn(y_i - y_j) \end{aligned} \tag{4.6}$$

where the binomial $\binom{n}{2}$ represents the number of combinations to choose two items from a set of n elements, while $sgn()$ extracts the sign of the real number passed.

In the end, the recommendations for the User-based CF can be extracted by computing the ratings for a protein p and a ligand i as follows:

$$\hat{r}_{pi} = \frac{\sum_u r_{ui} * s_{up}}{\sum_u s_{up}}$$

where s_{up} is equal to the similarity score of the p with respect to another protein u.

4.3.3. Evaluation

A proper design of the evaluation of a RS is crucial in order to gain an understanding of the effectiveness of the various algorithms. The evaluation system is not unique, but instead it depends on the context of the problem we are evaluating. Moreover, different validation methods adopted during the training phase could impact in different ways the

robustness of the models, hence the system responsible for assessing the quality of the recommendations should be chosen consciously.
The main objective of this thesis is to extract the most promising molecules for a **new protein**, exploiting a dataset that describes the interaction strengths of a known set of proteins with respect to a much larger collection of ligands. A promising molecule is described by a large affinity with the protein of interest, which corresponds to a high binding interaction score. Consequently, the problem can be re-formulated as a *re-ranking* task in charge of detecting and *recommending* molecules with the greatest affinity.

To validate our work we are inspired by a real drug discovery scenario in which a multi-step screening procedure is performed during the research of a new drug: to face the wide size of the chemical space, the evaluations are carried out by considering groups of molecules at a time and following an *iterative* process to explore as many molecules as possible. This involves the repetition of the validation procedure multiple times, in which, for each *round*, a set of recommendations is extracted, then evaluated and eventually added to the current dataset to increase the ability of the model on providing good predictions on the succeeding recommendation rounds.
The general idea can be summed up in the following points:

1. Consider a new protein
2. Train a recommendation model based on the current protein-ligand interaction scores
3. Recommend to the new protein a set of ligands for which the interaction is likely to be strong
4. Evaluate the recommendations (in-vivo, in-vitro) to determine if a protein-ligand complex can actually be synthesized
5. Add the recommended items along with their *real* affinity score to the dataset.
6. Repeat the steps from Point 2) a predefined number of times.

This procedure allows to incrementally evaluate a subset of ligands and improve the recommendations for the next round, exploiting the previously extracted information to develop better predictions. The focus is on **minimizing** the number of rounds involved to detect all the best molecules such that, ideally, ligands not prone to bind are postponed during the in-laboratory experimentation.
Obviously, we cannot compute any kind of in-vivo or in-vitro experimentation because of their complexity and cost, but we try to emulate this behavior by using directly the

docking pipeline to assign a value to the interaction strength.

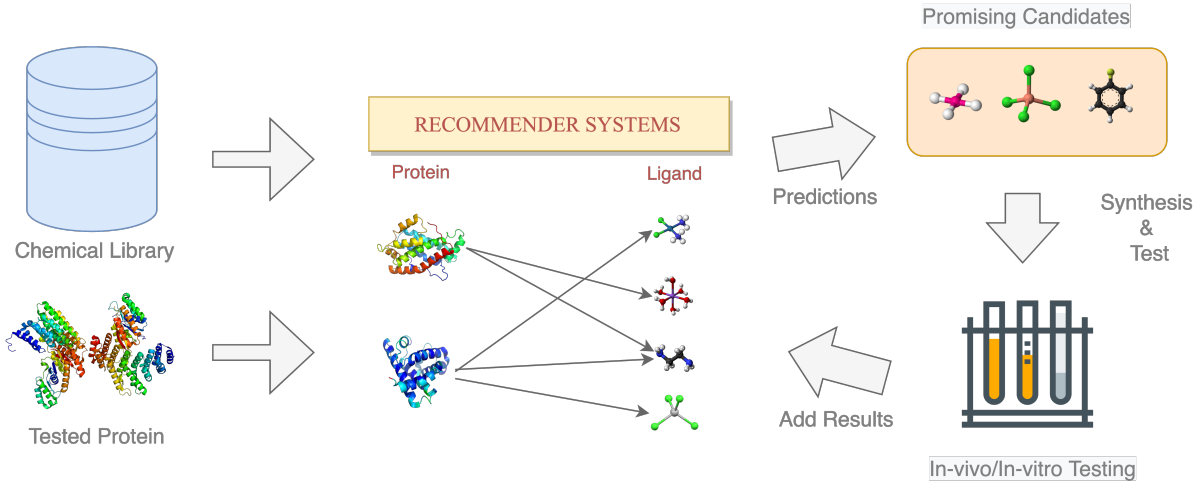


Figure 4.8: Evaluation Pipeline

5 | Results

In this Chapter we are going to outline the final results that come from the proposed Recommender Systems models. In the next sections, we firstly define the experimental setup (Chapter 5.1) we adopt, followed by a definition of the evaluation metrics (Chapter 5.2) we use to test the performance of the models. Finally, in Chapter 5.3 a comparison of the result of the models is aimed at assessing the quality of the recommendations.

5.1. Experimental Setup

The experiment consists of the use of Recommender Systems models to define the sequence of ligands to analyze, prioritizing those that are more likely to bind the target protein. We try to emulate a real-case scenario where *rounds* of screening evaluations are progressively executed (Chapter 4.3.3). Following the same idea, we propose a *cross-validation* approach in which each validation-fold undergoes a number of consecutive rounds of recommendations, pursuing the goal of discovering as fast as possible the complete set of most relevant items.

5.1.1. Leave-One-Protein-Out Cross-Validation

To assess the performance of the models, we leverage a *leave-one-out* cross-validation integrated with the iterative evaluation described in Chapter 4.3.3. In general, a common cross-validation procedure splits the training set into k smaller and disjoint sets called *folds*. Then, for each fold F:

1. The model is trained using all the folds except F
2. The model metrics, such as precision, accuracy or recall, are validated on the remaining part F, which is indeed considered as a test-set

The metric results of the folds are aggregated and averaged to establish the performance measure reported by the cross-validation. This approach is useful to limit the risk of overfitting and increases the robustness and the stability of the predictor.

In this chemical context, the cross-validation can be interpreted as a ***leave-one-protein-out*** approach such that each validation fold is represented by a particular *protein*. The validation fold starts without any interaction score previously computed, i.e. the corresponding protein's profile is empty. On the other hand, the relevant items, which represent the *test-set* of that fold, correspond to the N molecules whose binding interaction score is the greatest.
During this validation process, the tested fold is subject to multiple *rounds* of recommendations. At each *round*, models predict a collection of compounds that are likely to fit the protein's pocket; these recommended molecules are docked, their interaction scores extracted and added to the protein's profile to provide better predictions in the following round.
The fact that we start testing a new protein without knowing any of its past molecular affinities leads to a well-known issue in recommendation systems, called **cold-start** problem: it points out the difficulty of a system in extracting recommendations to a user for whom there is a lack of information since no interaction has occurred yet. This problem is mainly noticeable in Collaborative Filtering and Content-based models where the final rating is computed by a dot product between the URM and the similarity matrix, such that if no scores are available for the considered protein in the URM, then the predicted ratings are null. In the validation process described above, this problem occurs in the first round of evaluation when a new protein is taken into account, but its profile of interaction scores is unknown. To mitigate it, we employ the use of the *Top-Popular* algorithm on the **first round** of recommendation for both the *Collaborative Filtering* and *Content-based* recommenders. In fact, the Top-Popular is not affected by the *cold-start* problem due to the fact that its predictions just focus on the current knowledge to extract the item popularity, and no similarity matrix is computed. After the first round of recommendations, the other models can take place without appealing to the aid of the Top-Popular algorithm and the process can continue further.

5.1.2. Testing Setup

During the test phase, we consider, for each protein, the *most promising* candidates as the group of $N = 10.000$ molecules having the highest scores. Concerning the iterative evaluation, we limit the number of *rounds* to a maximum of 10, each one characterized by 10.000 recommendations. Thus, the goal of our recommenders is to *minimize* the number of *rounds* employed to discover the most promising molecules. We tested the models described in Chapter 4.3.2, comparing our baseline Random Model with respect to two

different types of recommenders:

- *Collaborative* Recommender Systems: these systems study the protein-molecules interactions available in the training set to be able to predict the preference of another protein
- *Content-based* Recommender Systems: these systems exploit the *features* of the molecules to discover molecules similar to the ones available in the protein's profile.

The proposed models are executed over all the validation folds and their metric results are eventually averaged. Following this setup, in the next sections, we will discuss in detail the metrics employed and the results obtained.

5.2. Evaluation Metrics

In order to be meaningful, the adopted metrics should be representative indicators of the model performance. In our context, the goal is to capture the number of rounds required to discover all the most promising molecules of a protein whose interactions are initially unknown. To measure the quality of the model predictions in our multi-stage scenario we rely upon two different metrics, the first one applied to each round whereas the second one returns an estimate of the total performance.

5.2.1. Recall

At each round, we are interested in maximizing the number of promising molecules that have been identified. To measure the quality of such recommendations, we choose the *Recall* metric. The Recall is defined as the fraction of relevant items that are correctly predicted, as reported in Eq. 5.1:

$$Recall = \frac{|\text{relevant items} \cap \text{predicted items}|}{|\text{relevant items}|} \tag{5.1}$$

However, in Recommender Systems problems the goal is to provide the list of top-N relevant items, therefore the original definition of Recall is modified to model the concept of *cutoff at k*, where k specifies the number of recommended items. Eq. 5.2 defines the **Recall@k**, which shows the number of relevant items correctly detected among the k recommendations.

$$Recall@k = \frac{\text{Number of relevant items@k}}{k} \tag{5.2}$$

5.2.2. Area Under the Recall Curve

Since for each protein we recommend multiple times different sets of ligands, it is crucial to keep track of the number of items correctly predicted during the rounds. For this purpose, the Recall@k computed at each round is summed to the values of Recall@k on previous rounds. In this way, we have a measure of how many relevant items have been detected until a particular moment in time. This implies the introduction of a *Cumulative Recall* (Eq. 5.3) which defines the total portion of relevant items discovered until the round t:

$$\text{Cumulative Recall}(t) = \sum_{0}^{t} \frac{|\text{relevant items} \cap \text{predicted items}(t)|}{|\text{relevant items}|} \tag{5.3}$$

A curve can be delineated by plotting the values of CumulativeRecall at each round t: in general, the steeper the curve, the faster the convergence of the model. Computing the area under that curve, we can measure the speed of convergence of a model after R rounds.
In a discrete environment composed of T equally spaced intervals Δx, the *Area Under the Curve* (AUC) is calculated using the trapezoidal-rule as described in Eq. 5.4:

$$AUC = \Delta x \left(\sum_{t=1}^{T-1} f\left(x_t\right) + \frac{f\left(x_T\right) + f\left(x_0\right)}{2} \right) \tag{5.4}$$

where $\Delta x = \Delta x_k = \frac{x_T - x_0}{T}$ corresponds to the size of each interval, while $f(x_t)$ is the value on the curve, defined by function f, at position x_t.
In our study, we are mainly interested in the **Area Under the Cumulative Recall Curve**, i.e. the area of the curve defined by the *CumulativeRecall* function after R rounds.

5.2.3. Pearson Correlation Coefficient

The **Pearson Correlation Coefficient** measures the *linear* correlation of two vectors of observations. Since it relies upon the computation of the covariance between the vectors, which is a linear indicator of the joint variability of two variables, this coefficient can only detect a *linearity* measure in the correlation of the variables. Then, the Pearson Correlation Coefficient is defined as:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{5.5}$$

where X and Y represent the vectors of observations, $cov(X, Y)$ is the covariance of these vectors and σ corresponds to the standard deviation of a vector. The Pearson Correlation Coefficient $\rho_{X,Y}$ can be interpreted as a normalized version of the covariance $cov(X, Y)$, having the values lying in the range $[-1, +1]$

5.3. Results

To evaluate the recommendation models described in Chapter 4.3.2, we adopt the *leave-one-protein-out* cross-validations (Chapter 5.1.1) and we average the results of the metrics over all the folds. Given a single fold (a protein), to get an understanding of how much we are able to detect a subset of promising molecules at each *round* we use the Recall metric, while a measure of the amount of such molecules discovered after R rounds is represented by the Area Under the Cumulative Recall Curve (AUC). The promising candidates for a protein correspond to the N molecules with the highest affinity score; therefore, the goal of the models is to spot all those candidates as fast as possible, *minimizing the number of rounds* employed to detect them all.
In the following sections, we are going to outline the quality of Collaborative Recommenders, which study the preferences of several proteins to make predictions for a new one, and Content-based Recommenders, which leverage the *features* of the molecules to provide their recommendations; these models will be compared with respect to the state-of-the-art baseline explained in the next section.

5.3.1. Baseline: Random Model

State-of-the-art techniques don't allow any preferential choice of the ligand to submit into the docking pipeline, thus their sequence of molecules to process can be interpreted as a **Random** search.
Figure 5.4 shows the CumulativeRecall obtained at the subsequent rounds of our cross-validation process and the Area-Under-the-Curve to define the total performance of the models over the rounds.
Accordingly to this Figure, a Random search is not the ideal method since it becomes hard to locate the promising molecules without analyzing the entire dataset. This is due to the extremely low probability associated to discover the set of $N = 10.000$ relevant items among the 8.5 million elements, roughly equal to 0.11%; being inversely proportional to the size of the dataset, this probability is even lower for much larger datasets.

5.3.2. Collaborative Approach

Collaborative approaches base their predictions on the "collaborative" behavior of all the users taken into account, in the sense that the preferences that other users give to the items are exploited to generate suggestions for another user. In our case, we leverage the interaction scores of a set of proteins to recommend ligands to another protein.
A comparison between the performance of collaborative approaches and our baseline is described in the next Section.

Top Popular

The quality of the Top Popular is strictly tied to the "popularity" of the items in the dataset. As stated in Chapter 4.3.1, our dataset is *dense* and *explicit*, thus the common meaning of "popularity", which refers to the number of times users interact with an item, can no longer be applied. Instead, we consider as *popular* those items that have the **highest average affinity score**, i.e. we average the item interaction scores over all proteins and we extract the items with the highest value.
To get some insight about the performance of the Top Popular, we can study how the molecules are ranked inside each protein. In particular, for each protein in the dataset, we can rank their molecules based on the affinity score they have, from higher to the lower, and find the molecules ranked in the top positions in one or more proteins. This concept can be expressed by considering the first $K = 10.000$ ranked molecules for each protein and calculating the frequency associated to every molecule that appears at least one time among those rankings.

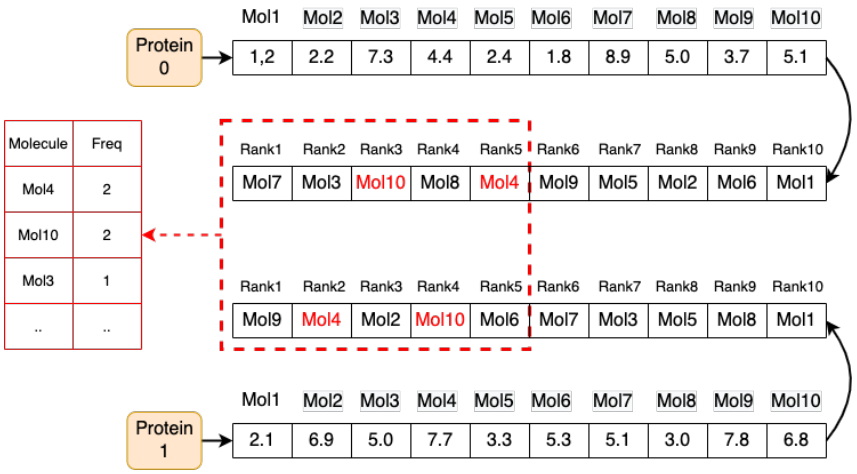


Figure 5.1: Example of counting the frequency of a molecule in the top-5 positions

Figure 5.1 shows an example of this counting: once ranked by their score, only *Mol4* and

Mol10 have frequency equal to 2, since they appear in both protein's top-**5** positions. Formally, it can be defined as:

- *Frequency in Top-K positions*: it counts the number of times a molecule is present among the first K ranks of all the proteins' profiles:

$$\text{Frequency@TopK-ranks}(item) = \sum_{p} \mathbb{1}_{(item \in Rank@K(p))}$$

  where the function $Rank@K(p)$ selects the first K-ranked molecules in protein p and $\mathbb{1}$ returns 1 if the item's rank belongs to the first K positions of the protein p, 0 otherwise.

Consequently, we can group the molecules by this frequency in order to establish how many of them appear in one or in just a portion of proteins' top ranks; the result is shown in Figure 5.2.

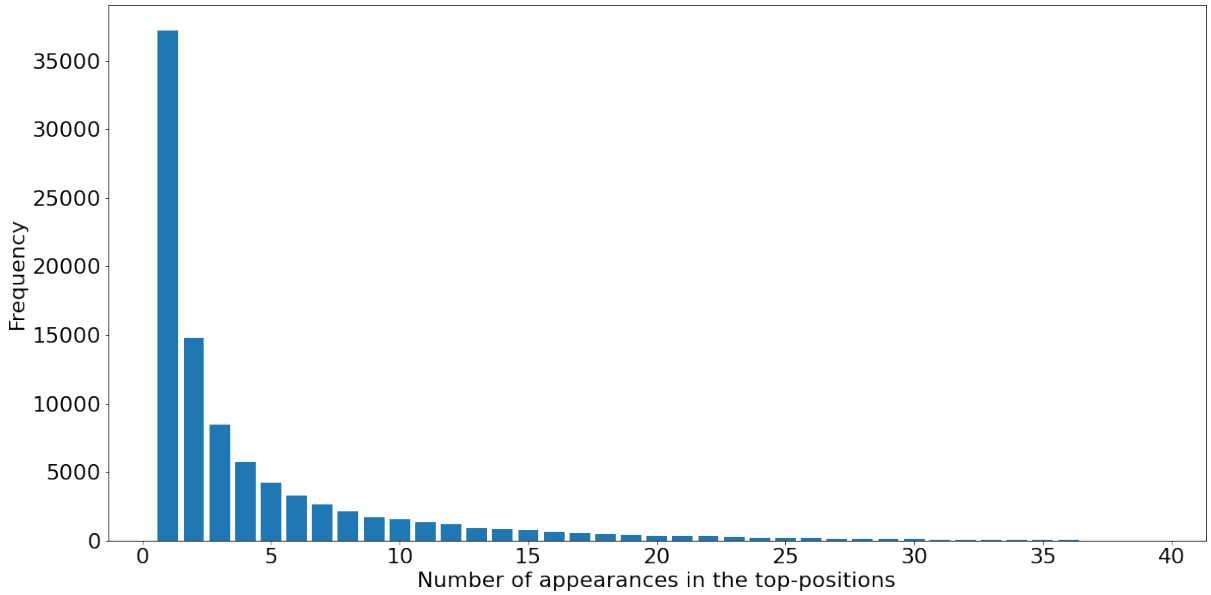


Figure 5.2: Frequency of Molecules Appearance in Top Positions

To give an idea of the meaning behind Figure 5.2, it shows, for instance, that there are more than 35.000 molecules whose interaction score appears among the top K positions in *only one* protein's profile, whereas about 5.000 molecules whose score appears among the top K positions in *four* different protein's profiles.
Then, it is possible to infer that few molecules appear in all or most of the proteins' top ranks, which means that there are proteins that share the same preferences for some ligand. This shared behavior motivates us in exploiting recommendation algorithms to capture collective preferences and it gives us an idea about the performance of the Top

Popular model. In fact, the more a molecule appears in the upper ranks of proteins, the simpler it will be for the Top Popular to correctly predict it since its score will be high **on average** over all the proteins. Analyzing in detail the predictions for a single protein on the *first round* of evaluation can clarify this concept further. Then, taking into account protein *1a30* as the tested fold in our cross-validation, Figure 5.3 can be described as follows:

- The blue bars refer to the previously discussed Figure 5.2, restricting the view to only the *promising molecules* of protein 1a30. This set of molecules represents the *test-set* for this fold, since they are the items we want to detect. Among those molecules, there are about 600 of them that appear just one time in the top ranks of the proteins, and a single one that is ranked at top positions in all the 39 proteins.
- The orange bars refer to the molecules recommended by the Top Popular algorithm in the first round of evaluation. In particular, we are considering only the molecules correctly predicted, i.e. the ones recommended by the model and belonging to the test set of the protein 1a30. Also in this case, the height of the bars specifies the amount of molecules (correctly recommended) that are ranked in the upper positions M-times among all the proteins.

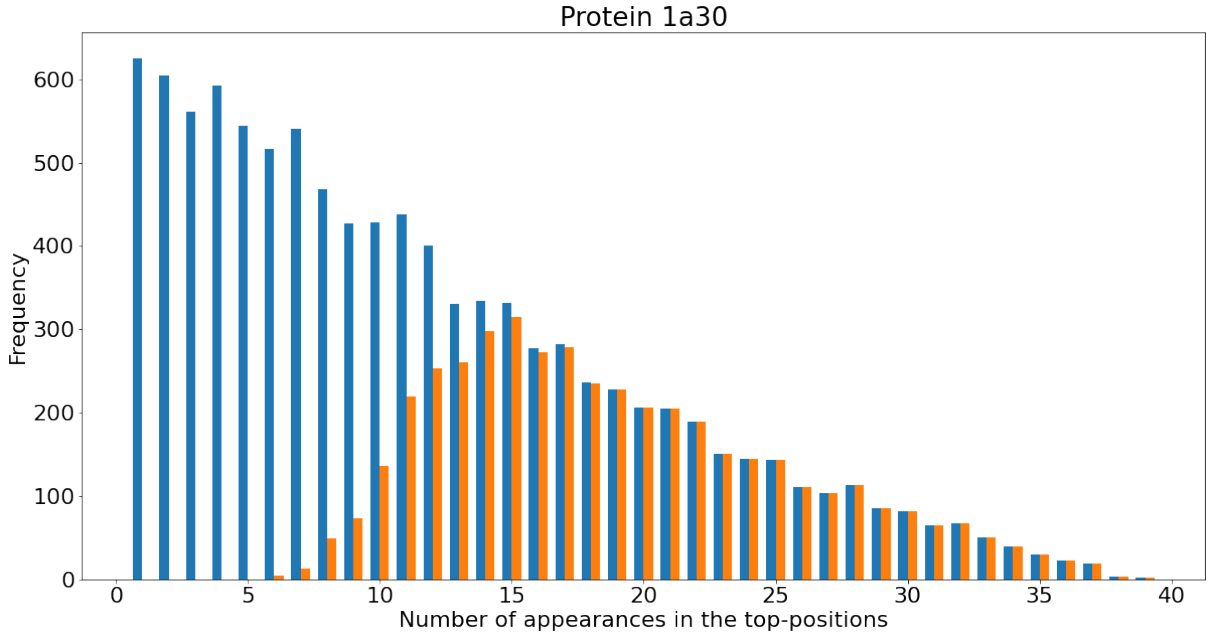


Figure 5.3: Round 0: Recommendations for protein 1a30

From Figure 5.3 we can deduce how simple is for the Top Popular to detect those compounds that are frequently ranked in the top positions: if a molecule is ranked high in *at least half* of the proteins, then its score will be high **on average** over all the proteins and

the model will recommend it.
The Top Popular reveals to have good performance in predicting relevant items in the first round of evaluations, which is particularly useful to face the *cold-start* problem on the other models. However, it struggles in recommending molecules that are less "popular", as it is shown in the left-side of Figure 5.3.
Finally, Figure 5.4 shows the comparison between the CumulativeRecall of the Random model and the Top Popular after 10 rounds, highlighting the AUC value for both the curves. While the Random model seems to detect no promising molecules, on the other hand the Top Popular model is capable of detecting at each round part of the relevant molecules so that after 10 steps, most of the promising candidates are correctly spotted. Moreover, the Top Popular seems to be particularly powerful at the starting point, when no previous affinity scores are available for the tested protein. The discrepancy in the results that the two models exhibit shows how a simple recommender can effectively prepend a docking pipeline to prioritize the sequence of the compounds.

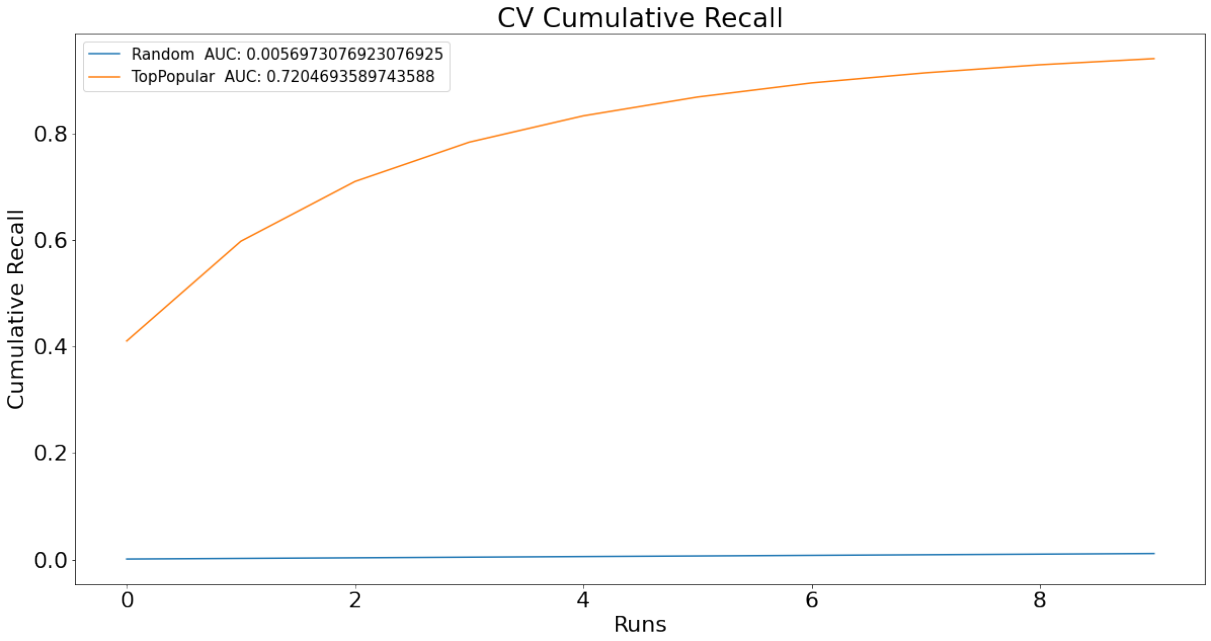


Figure 5.4: Comparison between Random model and Top Popular model

Collaborative Filtering

The performance of Top Popular recommendations outlines how *collaborative* analysis can be successfully exploited to face the problem. For this reason, a *Collaborative Filtering* model (Chapter 4.3.2) is chosen to improve the detection of protein's preferences by studying the similarities between them. In particular, we adopt a *user-user memory-*

based Collaborative Filtering model, supported by the *Spearman Correlation Coefficient*: in other terms, the Collaborative Filtering model extracts a similarity matrix (*memory-based*) to measure how similar two proteins are (*user-user*); the pairwise similarity is computed by the application of the Spearman Correlation Coefficient on the users' profiles.

The results of the model are shown in Figure 5.5 in comparison with the performance

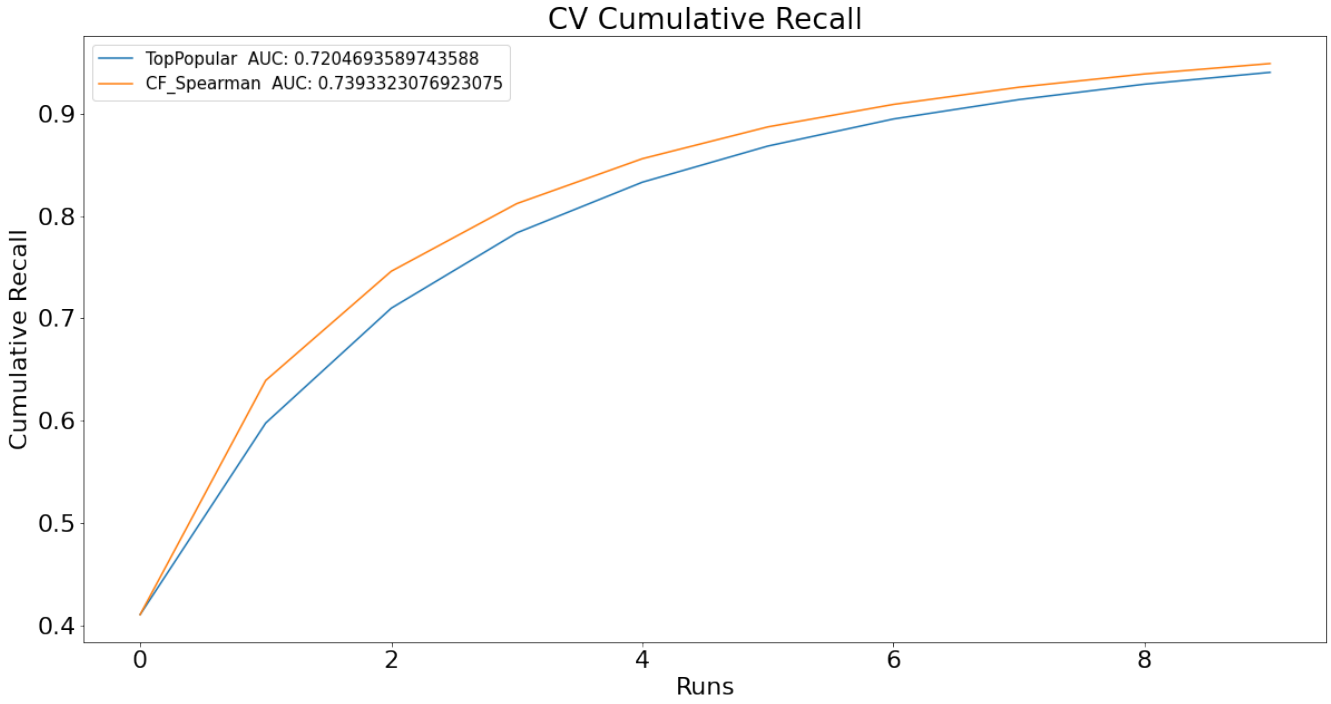


Figure 5.5: Top-Popular and Collaborative Filtering results

of Top Popular. As we can see, the Collaborative Filtering is able to increase the overall AUC score by leveraging the collective behavior of the proteins: at each round, but especially in the initial ones, the model detects and recommends a larger set of relevant items. This leads to faster discovery of the most promising candidates, increasing the speed of convergence of the algorithm.

5.3.3. Content-based Approach

Content-based approaches leverage additional information to find items similar to the user's preferences. This information describes some property (*feature*) about the items such that a pairwise measure of the similarity between two items can be extracted. In our problem, the set of *features* described in Table 4.2 is available and it is used to serve the Content-based Filtering recommender.

Content-based Filtering

In order to discover the "unpopular" molecules (see Chapter 5.3.2), we exploit their features to increase the chances of detecting new relevant items. To this purpose, we apply the Content-based Filtering model, whose performance is compared to the Top Popular and Collaborative-Filtering solutions in Figure 5.6. In this context, the Content-based model aims at discovering items that are structurally similar to those available in the protein's profile. Since the dataset is explicit, the affinity scores in the protein's profile, which explain the protein's preferences, can be exploited to weigh more some items with respect to others, such that we can predict compounds similar to those weighted more. As well-known in the state-of-the-art, the Content-based recommender comes out to be better than a Random model, but in this case, it is not as powerful as the aforementioned Top Popular algorithm nor the Collaborative Filtering.

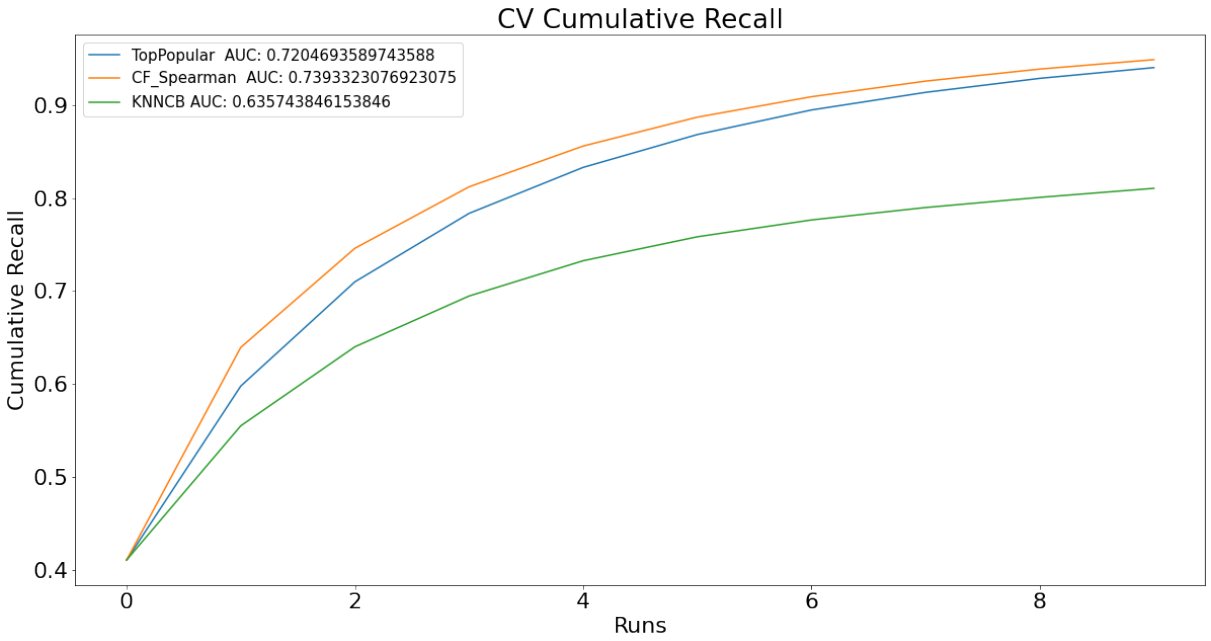


Figure 5.6: Top-Popular and Content-based Filtering

The inferior performance of the Content-based recommendations can be related to the quality of the features we have. In fact, looking at the correlation between them, shown in Figure 5.7, we notice that a group of features is highly correlated: in particular, Molecular_Weight, Num_Atoms, Num_Rings, Num_AromaticRings, Num_Bonds and Num_RingBonds have a correlation close or higher to 0.6. Highly correlated features are usually avoided in classical Machine Learning solutions since one variable can be linearly predicted from another, thus discarding them can improve the stability of the model as well as its accuracy.

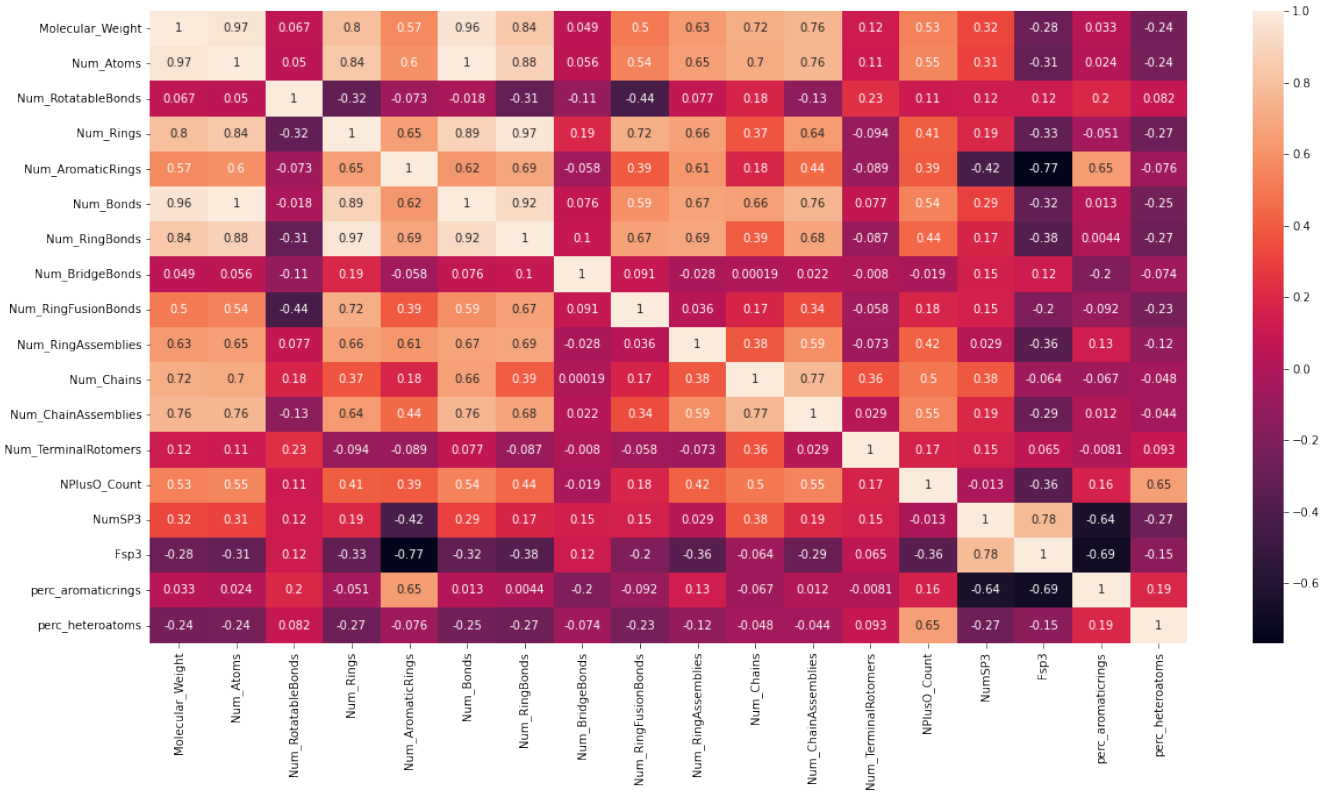


Figure 5.7: Pearson Correlation

Moreover, the remaining attributes don't impact heavily the performance of the model, even if their relative correlations are low. To clarify this concept, we use the *Principal Component Analysis* (PCA) to understand which combination of those features mainly describes the variance of the projected data.

Principal Component	Explained Variance Ratio
1	0.9365764074303904
2	0.05380602664340277
3	0.004281900732658818
4	0.0029358760538973367
5	0.0010077260492363262

Table 5.1: Principal Components of PCA

Looking at the percentage of explained variance on the first five *principal components*, depicted in Table 5.1, we notice that the *first Principal Component* contributes with a ratio of 93% on the total variance, which widely corresponds to the biggest portion of

the latter. In detail, each component is defined as a linear combination of the available features:

$$Y = w_{i1}X_1 + w_{i2}X_2 + .. + w_{iq}X_q \qquad q \in (0, NUM\_FEATURES)$$

where w_{ij} represents the weight that the i-th principal component associates to the j-th feature X_j. Since the first principal component can be interpreted as the direction that maximizes the variance, we search among its attributes the ones weighted more, in order to capture which features mainly influence the *principal component*. To this purpose, Table 5.2 shows the coefficients related to each feature, sorted by their values.

Feature	Coefficient
Molecular_Weight	0.9903892200838507
Num_Bonds	0.08457758034875348
Num_RingBonds	0.072872905198618
Num_Atoms	0.07141769187337957
NumSP3	0.01670931572439731
Num_Chains	0.01603075082656827
Num_ChainAssemblies	0.01543136135177321
Num_Rings	0.013140501200239546
NPlusO_Count	0.012149666130198552
perc_aromaticrings	0.008058653894403994
Num_AromaticRings	0.007825235760892725
Num_RingAssemblies	0.006744983486441347
Num_RingFusionBonds	0.00612226620127966
Num_RotatableBonds	0.001342123046646477
Num_BridgeBonds	0.000972217866848553
Num_TerminalRotomers	0.0005227286562260877
Fsp3	-0.00046709925148353154
perc_heteroatoms	-0.016058868315619364

Table 5.2: Coefficients of the First Principal Component

As clear from that Table, Molecular_Weight seems to be by far the most meaningful attribute in our dataset: its coefficient has the highest value, which means that the *first principal component* is highly influenced by this feature.
Given its importance, we examine how the affinity scores behave with respect to the Molecular_Weight: also in this case, we can measure how much this feature is correlated with the affinity scores generated by *X-Score* scoring function applying the Pearson Correlation.

The values of the corresponding coefficients are presented in Table 5.3: the results show a general trend of the observed scores to be highly correlated to molecular weights. To outline this correlation, a better visualization is exposed in Figure 5.8, where the interaction scores of the molecules for *protein 1a30* are considered along with the corresponding molecular weights. The blue points depict the entire set of observations; among them, we highlight the molecules having the highest score (red points), i.e. the best binding candidates for the protein in account. As shown, the scores tend to grow as the molecular weight increases with an almost linear dependency. As a consequence, the molecules with the highest binding affinity are characterized by an elevated weight, which generally implies a large *number of atoms*. Therefore we can conclude that **X-Score** scoring function is deeply biased toward heavier molecules

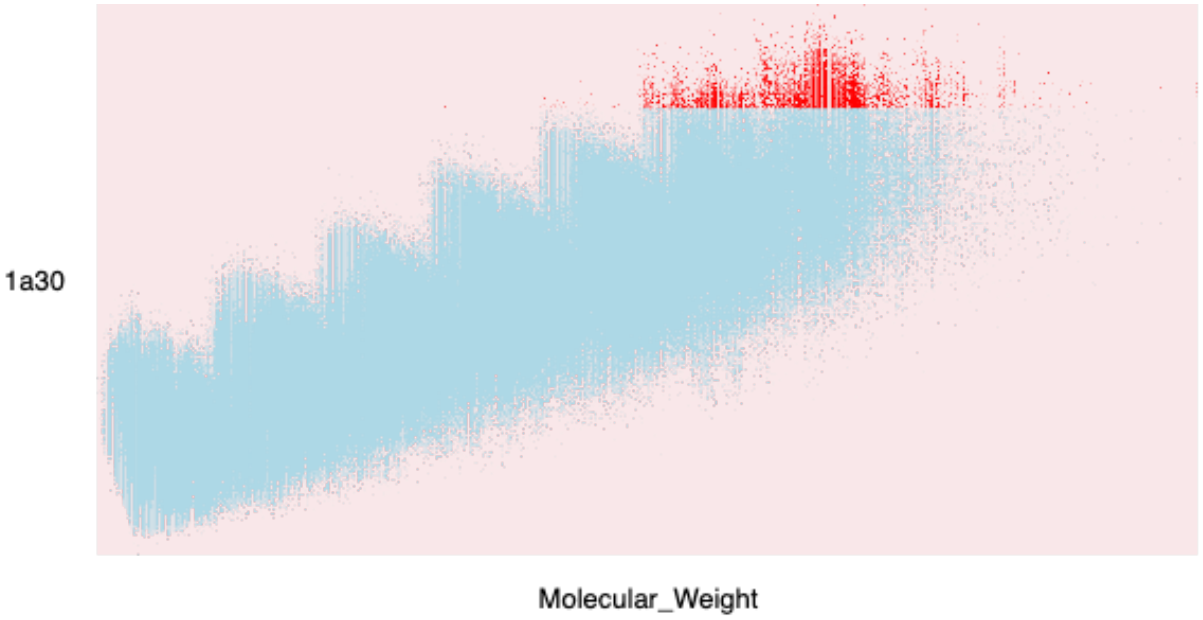


Figure 5.8: Distribution of the scores of protein 1a30 w.r.t. Molecular Weight

Protein	Coefficient	Protein	Coefficient
1a30	0.8794410524690838	1ctr	0.852753859777316
NSP16	0.870858772286348	1o3f	0.8717309995481427
NSP9	0.8204735407660785	3su3	0.8510631469393335
3su5	0.861776353298642	2vw5	0.8749032172857608
2yki	0.8784812023259723	NSP6	0.8315851078588629
SPIKEACE	0.8994082414175064	NSP13allo	0.8922807460988689
3ehy	0.8740655044042194	3nq3	0.8198079564830478
3su2	0.849029948597251	3oe5	0.8709211050256361
1w4o	0.8639737702016373	3s8o	0.8540784622477039
NSP12palm	0.8636731689809293	Nprot	0.8823970895602535
3CL	0.8695487952507241	3cyx	0.8835157125435559
NSP13ortho	0.8874754686904863	1jyq	0.8525661788681704
NSP12ortho	0.865743968323274	1sln	0.8513069240360993
2yge	0.8624791094611084	3lka	0.8496387948694393
3ov1	0.8562784524526939	3f17	0.8760265461024969
3ozt	0.871353269272262	3gy4	0.770575885566345
NSP14	0.894308471376838	1uto	0.8010676207349611
1yc1	0.8767351266177021	2d1o	0.8799236342333021
4djr	0.8776619548797823	1u1b	0.8796103117796996
NSP3	0.8714612692429098		

Table 5.3: Correlation Coefficients of Affinity Scores with respect to Molecular Weight

6 | Conclusions and future developments

In this thesis, we first reviewed the state-of-the-art methods in Virtual Screening problems, describing the difference between Ligand-based and Structured-based approaches. A particular emphasis was placed on **Molecular Docking** techniques which aim at finding the best reciprocal orientation and conformation of a chemical candidate when it binds a macro-molecule, given the three-dimensional structure of the latter. These techniques are exploited to analyze large libraries of compounds (*ligands*) and to select just a portion of them having a high affinity with the protein in account. However, virtual screening approaches, especially those leveraging Molecular Docking methods, are computationally-intensive and time-consuming since the number of pairwise protein-ligand evaluations grows with the number of compounds in the chemical library, which generally is in the order of millions. To support the computation and leverage the embarrassing parallelism of these screening methods, *high-performance-computing* architectures are adopted: in such systems, a distributed environment hosts the execution of an application, which should be properly designed to optimize the usage of nodes in the cluster. We described a recent paradigm in HPC systems that enables the use of HPC-as-a-Service for academic and industrial purposes and avoids the costs related to the creation and management of an in-house HPC cluster, allowing users to run their applications remotely and use the cluster as long as they need. In the state-of-the-art techniques, there is no preference in the sequence of ligands to evaluate. This approach can decrease the performance of screening, especially if the computation on a cluster is subjected to time and resource limitations on its usage.

The main contribution of this thesis regards the application of Recommender Systems to prioritize the screening of compounds starting from those that are more likely to bind the target protein: this approach could greatly improve the quality of the screening phase, considering that the wideness of the chemical space does not allow its complete exploration and usually the simulations are limited in the number of molecules that can be processed at time; so having a method that selects the relevant molecules can be effi-

ciently used to wisely conduct simulations, delaying (or avoiding) the computation for unlikely candidates.
Since no current data were available to train the models, we extracted a huge dataset of protein-ligand interaction scores running a Molecular Docking simulation on IT4Innovations' HPC cluster. Once data generation was completed, we built classical recommendation models and evaluated their performance following a custom leave-one-out cross-validation, which emulates a real-case scenario. We showed that content-based recommenders, which are models based on the user/item's features, go far beyond a random exploration of the items; however, the use of collaborative approaches, such as Top Popular and Collaborative Filtering, has proven to be the best solution in discovery compounds with the highest binding affinity. Therefore, Recommender Systems can be successfully exploited to select the relevant portion of the chemical space and, thus, reduce the computational costs and execution time associated to docking simulations

6.1. Future Works

The application of Recommender Systems in optimizing drug discovery pipelines has shown promising results. However, further improvements and researches can be addressed:

- A detailed analysis of the data generated by the docking pipeline revealed that scores assigned by the scoring function *X-Score* are strongly biased toward heavy molecules. Since finding a scoring function able to perfectly measures the binding forces of a protein-ligand interaction is still an open problem in computational chemistry, our approach should be evaluated with different scoring functions to understand the impact they have on the models
- In our work, we focused on classical *memory-based* algorithms to build the models because of their ability to include new data directly in the prediction. Motivated by the results we obtained in our research, an attempt to improve the performance of the system could derive from the use of *model-based* techniques or advanced machine-learning algorithms.
- Our simulations were subjected to limits on the HPC cluster usage, which restricts the amount of proteins and ligands that we could consider. Further experiments can be conducted in both directions, increasing the number of proteins to make the models much more stable, or enlarging the chemical space to provide a wider exploration of ligands.

Bibliography

[1] URL https://foldingathome.org/?lng=en-GB.

[2] URL https://www.altair.com/hpc-cloud-applications/.

[3] A. Acharya, R. Agarwal, M. B. Baker, J. Baudry, D. Bhowmik, S. Boehm, K. G. Byler, S. Chen, L. Coates, C. J. Cooper, et al. Supercomputer-based ensemble docking drug discovery pipeline with application to covid-19. *Journal of chemical information and modeling*, 60(12):5832–5852, 2020.

[4] M. M. Ahsan, M. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique. Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, 9(3):52, 2021.

[5] M. AlQuraishi. Casp14 scores just came out and they're astounding, 2020. URL https://twitter.com/MoAlQuraishi/status/1333383634649313280.

[6] C. Beato, A. R. Beccari, C. Cavazzoni, S. Lorenzi, and G. Costantino. Use of experimental design to optimize docking performance: The case of ligendock, the docking module of ligen, a new de novo design program. *Journal of Chemical Information and Modeling*, 53(6):1503–1517, 2013. doi: 10.1021/ci400079k. URL https://doi.org/10.1021/ci400079k. PMID: 23590204.

[7] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Res*, 28(1): 235–242, Jan 2000.

[8] P. Bonate. A brief introduction to monte carlo simulation. *Clinical pharmacokinetics*, 40:15–22, 02 2001. doi: 10.2165/00003088-200140010-00002.

[9] N. Brooijmans. Molecular recognition and docking algorithms. *Annual review of biophysics and biomolecular structure*, 32:335–73, 02 2003. doi: 10.1146/annurev.biophys.32.110601.142532.

[10] I. N. S. Center. Heappe framwork. URL https://sc18.supercomputing.org/proceedings/tech_poster/poster_files/post192s2-file2.pdf.

[11] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63, 2015.

[12] Cineca. Annual report 2020-2021. URL `https://www.hpc.cineca.it/sites/default/files/REPORT_HPC_20202021_0.pdf`.

[13] M. Cobb. 60 years ago, francis crick changed the logic of biology. *PLoS biology*, 15 (9):e2003243, 2017.

[14] P. S. S. Committee. The scientific case for computing in europe 2018-2026. 2018.

[15] F. Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[16] M. F. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[17] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer. Description of several chemical structure file formats used by computer programs developed at molecular design limited. *Journal of Chemical Information and Computer Sciences*, 32(3):244–255, 1992. doi: 10.1021/ci00007a012. URL `https://doi.org/10.1021/ci00007a012`.

[18] J. A. DiMasi, H. G. Grabowski, and R. W. Hansen. Innovation in the pharmaceutical industry: New estimates of r&d costs. *Journal of Health Economics*, 47:20–33, 2016. ISSN 0167-6296. doi: https://doi.org/10.1016/j.jhealeco.2016.01.012. URL `https://www.sciencedirect.com/science/article/pii/S0167629616000291`.

[19] B. E. Eaton, L. Gold, and D. A. Zichi. Let's get specific: the relationship between specificity and affinity. *Chemistry & biology*, 2(10):633–638, 1995.

[20] T. J. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz. DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J Comput Aided Mol Des*, 15(5):411–428, May 2001.

[21] J. Fan, A. Fu, and L. Zhang. Progress in molecular docking. *Quantitative Biology*, 7 (2):83–89, 2019.

[22] L. G. Ferreira, R. N. Dos Santos, G. Oliva, and A. D. Andricopulo. Molecular docking and structure-based drug design strategies. *Molecules*, 20(7):13384–13421, Jul 2015.

[23] N. Foloppe and R. Hubbard. Towards predictive ligand design with free-energy based computational methods? *Curr Med Chem*, 13(29):3583–3608, 2006.

[24] M. P. Forum. Mpi: A message-passing interface standard. Technical report, USA, 1994.

[25] R. A. e. a. Friesner. Glide: a new approach for rapid, accurate docking and scoring. 1. method and assessment of docking accuracy. *J Med Chem.*, 47(7):1739–49, 2004.

[26] D. Gadioli, G. Palermo, S. Cherubin, E. Vitali, G. Agosta, C. Manelfi, A. R. Beccari, C. Cavazzoni, N. Sanna, and C. Silvano. Tunable approximations to control time-to-solution in an HPC molecular docking mini-app. *CoRR*, abs/1901.06363, 2019. URL `http://arxiv.org/abs/1901.06363`.

[27] D. Gadioli, E. Vitali, F. Ficarelli, C. Latini, C. Manelfi, C. Talarico, C. Silvano, C. Cavazzoni, G. Palermo, and A. R. Beccari. Exscalate: An extreme-scale in-silico virtual screening platform to evaluate 1 trillion compounds in 60 hours on 81 pflops supercomputers, 2021.

[28] J. F. Gilabert, D. Lecina, J. Estrada, and V. Guallar. Monte carlo techniques for drug design: The success case of pele. *Biomolecular Simulations in Structure-Based Drug Discovery*, pages 87–103, 2018.

[29] A. Gimeno, M. J. Ojeda-Montes, S. Tomás-Hernández, A. Cereto-Massagué, R. Beltrán-Debón, M. Mulero, G. Pujadas, and S. Garcia-Vallvé. The light and dark sides of virtual screening: what is there to know? *International journal of molecular sciences*, 20(6):1375, 2019.

[30] I. A. Guedes, C. S. de Magalhães, and L. E. Dardenne. Receptor-ligand molecular docking. *Biophys Rev*, 6(1):75–87, Mar 2014.

[31] C. Gueto-Tettay, A. Martinez-Consuegra, L. Pelaez-Bedoya, Drosos-Ramirez, and J. C. G-score: A function to solve the puzzle of modeling the protonation states of Beta-secretase binding pocket. *J Mol Graph Model*, 85:1–12, 10 2018.

[32] W. H. Halperin I, Ma B and N. R. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins*, 47:409–443, 2002.

[33] C. Hetényi and D. van der Spoel. Efficient docking of peptides to proteins without prior knowledge of the binding site. *Protein Sci*, 11(7):1729–1737, Jul 2002.

[34] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008. doi: 10.1109/ICDM.2008.22.

[35] I. Irurzun-Arana, C. Rackauckas, T. O. McDonald, and I. F. Trocóniz. Beyond

Deterministic Models in Drug Discovery and Development. *Trends Pharmacol Sci*, 41(11):882–895, 11 2020.

[36] M. A. Johnson and G. M. Maggiora. *Concepts and applications of molecular similarity*. Wiley, 1990.

[37] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, O. Ronneberger, R. Bates, A. Žídek, A. Bridgland, et al. Alphafold 2, 2020.

[38] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. 2009. doi: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5197422.

[39] J. Li, A. Fu, and L. Zhang. An overview of scoring functions used for protein–ligand interactions in molecular docking. *Interdisciplinary Sciences: Computational Life Sciences*, 11(2):320–328, 2019.

[40] Q. Li. Application of fragment-based drug discovery to versatile targets. *Frontiers in Molecular Biosciences*, 7, 2020. ISSN 2296-889X. doi: 10.3389/fmolb.2020.00180. URL `https://www.frontiersin.org/article/10.3389/fmolb.2020.00180`.

[41] A. N. Lima, E. A. Philot, G. H. G. Trossini, L. P. B. Scott, V. G. Maltarollo, and K. M. Honorio. Use of machine learning approaches for novel drug discovery. *Expert opinion on drug discovery*, 11(3):225–239, 2016.

[42] V. Limongelli. Ligand binding free energy and kinetics calculation in 2020. *WIREs Computational Molecular Science*, 10, 01 2020. doi: 10.1002/wcms.1455.

[43] J. L. Melville, E. K. Burke, and J. D. Hirst. Machine learning in virtual screening. *Combinatorial chemistry & high throughput screening*, 12(4):332–343, 2009.

[44] I. G. Metushi, A. Wriston, P. Banerjee, B. Oliver Gohlke, A. Michelle English, and A. Lucas. The workflow of the virtual screening protocol for screening of similar drugs to abacavir, 2015.

[45] J. J. Naveja and J. L. Medina-Franco. Finding constellations in chemical space through core analysis. *Frontiers in Chemistry*, 7, 2019. ISSN 2296-2646. doi: 10.3389/fchem.2019.00510. URL `https://www.frontiersin.org/article/10.3389/fchem.2019.00510`.

[46] R. Nussinov. Advancements and challenges in computational biology. *PLOS Computational Biology*, 11(1):1–2, 01 2015. doi: 10.1371/journal.pcbi.1004053. URL `https://doi.org/10.1371/journal.pcbi.1004053`.

[47] R. Pan, Y. Zhou, B. Cao, N. N. Liu, and R. Lukose. One-class collaborative filtering. doi: http://www.rongpan.net/publications/pan-oneclasscf.pdf.

[48] M. Pandey, M. Fernandez, F. Gentile, O. Isayev, A. Tropsha, A. C. Stern, and A. Cherkasov. The transformational role of gpu computing and deep learning in drug discovery. *Nature Machine Intelligence*, 4(3):211–221, Mar 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00463-x. URL `https://doi.org/10.1038/s42256-022-00463-x`.

[49] S. Philip, P. Shola, and O. Abari. Application of content-based approach in research paper recommendation system for a digital library. *International Journal of Advanced Computer Science and Applications*, 5, 10 2014. doi: 10.14569/IJACSA.2014.051006.

[50] V. W. Rodwell, D. A. Bender, K. M. Botham, P. J. Kennelly, and P. A. Weil. *Harper's illustrated biochemistry*. McGraw-Hill Education New York, NY, USA:, 2018.

[51] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 285–295, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133480. doi: 10.1145/371920.372071. URL `https://doi.org/10.1145/371920.372071`.

[52] Q. Shen, B. Xiong, M. Zheng, X. Luo, C. Luo, X. Liu, Y. Du, J. Li, W. Zhu, J. Shen, and H. Jiang. Knowledge-based scoring functions in drug design: 2. can the knowledge base be enriched? *Journal of Chemical Information and Modeling*, 51(2):386–397, 2011. doi: 10.1021/ci100343j. URL `https://doi.org/10.1021/ci100343j`. PMID: 21192670.

[53] T. Stow. The drug development and approval process is about much more than the final "okay", 2015. URL `https://catalyst.phrma.org/the-drug-development-and-approval-process-is-about-much-more-than-the-final-ok`

[54] D. Stumpfe and J. Bajorath. Similarity searching. *WIREs Computational Molecular Science*, 1(2):260–282, 2011. doi: https://doi.org/10.1002/wcms.23. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.23`.

[55] M. Su, Q. Yang, Y. Du, G. Feng, Z. Liu, Y. Li, and R. Wang. Comparative assessment of scoring functions: the casf-2016 update. *Journal of chemical information and modeling*, 59(2):895–913, 2018.

[56] U. O. Surrey. In silico cell for tb drug discovery. *ScienceDaily*. URL `www.sciencedaily.com/releases/2007/06/070624135714.htm`.

[57] Y. Tenorio, A. Hernandez-Santoyo, V. Altuzar, H. Vivanco-Cid, and C. Mendoza-Barrera. *Protein-Protein and Protein-Ligand Docking*, page 187. 05 2013. ISBN 978-953-51-1138-2. doi: 10.5772/56376.

[58] R. Thomsen and M. H. Christensen. Moldock: A new technique for high-accuracy molecular docking. *Journal of Medicinal Chemistry*, 49(11):3315–3321, 2006. doi: 10.1021/jm051197e. URL https://doi.org/10.1021/jm051197e. PMID: 16722650.

[59] P. H. Torres, A. C. Sodero, P. Jofily, and F. P. Silva-Jr. Key topics in molecular docking for drug design. *International journal of molecular sciences*, 20(18):4574, 2019.

[60] E. Vitali, D. Gadioli, G. Palermo, A. Beccari, C. Cavazzoni, and C. Silvano. Exploiting openmp & openacc to accelerate a molecular docking mini-app in heterogeneous HPC nodes. *CoRR*, abs/1901.06229, 2019. URL http://arxiv.org/abs/1901.06229.

[61] J. R. Wagner, C. T. Lee, J. D. Durrant, R. D. Malmstrom, V. A. Feher, and R. E. Amaro. Emerging Computational Methods for the Rational Discovery of Allosteric Drugs. *Chem Rev*, 116(11):6370–6390, 06 2016.

[62] W. P. Walters, M. T. Stahl, and M. A. Murcko. Virtual screening—an overview. *Drug discovery today*, 3(4):160–178, 1998.

[63] R. Wang, L. Lai, and S. Wang. Further development and validation of empirical scoring functions for structure-based binding affinity prediction. *J Comput Aided Mol Des*, 16(1):11–26, Jan 2002.

[64] O. J. Wouters, M. McKee, and J. Luyten. Estimated Research and Development Investment Needed to Bring a New Medicine to Market, 2009-2018. *JAMA*, 323 (9):844–853, 03 2020. ISSN 0098-7484. doi: 10.1001/jama.2020.1166. URL https://doi.org/10.1001/jama.2020.1166.

[65] U. Yadava. Search algorithms and scoring methods in protein-ligand docking. *Endocrinol Int J*, 6(6):359–367, 2018.

[66] L. Yang and M. Guo. High-performance computing: Paradigm and infrastructure. pages 1–778, 01 2006. doi: 10.1002/0471732710.

[67] R. Zhang, Q.-d. Liu, Chun-Gui, J.-X. Wei, and Huiyi-Ma. Collaborative filtering for recommender systems. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 301–308, 2014. doi: 10.1109/CBD.2014.47.

List of Figures

List of Tables