신약 개발 후보물질 발굴을 위한 거대 화합물 라이브러리 탐색 최적화

중간보고서

지도교수: 송길태 교수님

A (소프트웨어/인공지능)

08 - 이성최

201924656 이정민

202155565 성가빈

201845928 최우영

목차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 과제 목표

본 과제의 목표는 거대 화합물 라이브러리에서 주어진 타겟 단백질과 결합 확률이 높을 것으로 예측되는 화합물을 효율적으로 탐색하고 선별하는 것이다. 사용자가 단백질 파일을 업로드하면, 해당 단백질과 결합 확률이 가장 높게 예측된 10개의 화합물을 알려준다. 사용자는 모든 화합물을 실험하는 대신 추천된 일부 화합물의 결합 여부만 실험해 필요한 화합물을 더 적은 시간에 찾아낼 수 있게 될 것이다. 이를 위해 다양한 알고리즘을 제시하고 구현하여, 거대한 라이브러리를 빠르게 탐색하고 예측의 정확도가 높으면서 추천된 화합물 간의 유사도가 낮도록 해야한다.

1.2 요구조건 및 제약 사항

- **반응성이 높은 물질 선별 및 표시**
  추천 시스템은 타겟 물질에 대해 반응성이 높은 화합물들을 효과적으로 예측해 결과를 표시할 수 있어야 한다.

- **프로세스의 속도**
  스크리닝 과정은 10억여개의 데이터를 빠르게 처리할 수 있어야 한다.

- **화합물의 다양성 보장**
  추천 시스템은 비슷한 화학적 성질을 가진 화합물들만을 제안하는 것이 아니라, 다양한 화학적 성질을 가진 화합물을 포괄적으로 추천할 수 있어야 한다. 이는 신약 개발 과정에서 예상치 못한 부작용으로 인해 대체 물질이 필요할 경우에 대비하기 위해서이다.

- **사용자 친화적 인터페이스**
  쉽게 접근하고 사용할 수 있는 직관적인 사용자 인터페이스를 제공하여야 한다.

- **물질 결합 시각화**
  타겟 물질과 화합물이 결합되는 모습을 직관적으로 확인하기 위해 그래픽을 제공한다.

- **처리 시간과 연산비**
  10억 개의 화합물을 모두 스크리닝하기 위한 시간과 비용은 매우 크다.

2. 설계 상세화 및 변경 내역

2.1 화합물 탐색 알고리즘

현재까지 세 가지 탐색 알고리즘을 구현하였고, 추가적인 탐색 알고리즘을 적용하고 평가할 예정이다.

1. **무작위 탐색**

   무작위 탐색 알고리즘은 전체 리간드 중 40개를 무작위로 선택한 후, 단백질과 도킹을 실시하여 결합력이 높은 상위 10개 리간드를 추천하는 방식으로 구체화, 구현하였다. 이 방법은 탐색 속도는 빠르지만, 탐색의 다양성과 정확성 측면에서 한계가 있다. 따라서 무작위 탐색은 따라서 탐색 방식으로 적합하지 않다는 결론을 내렸으며, 비교군으로만 활용될 예정이다.

2. **화합물 클러스터링 및 탐색**

   화합물 클러스터링 및 탐색 알고리즘은 두 가지 방식으로 구현하였다.

   - K-평균(K-means) 알고리즘

     K-평균 알고리즘은 주어진 데이터 포인트들을 K개의 클러스터로 나누는 비지도 학습 기법이다. 각 데이터 포인트는 가장 가까운 클러스터 중심점(센트로이드)으로 할당되며, 각 클러스터의 중심점을 반복적으로 업데이트하여 클러스터를 형성한다. 이 방법은 계산 속도가 빠르지만, 초기 클러스터 중심점 설정에 따라 결과가 달라질 수 있다.

   - 병합 군집 알고리즘

     병합 군집 알고리즘은 데이터 포인트들을 개별 클러스터로 시작하여, 가장 가까운 두 클러스터를 반복적으로 병합해 나가는 계층적 군집 기법이다. 이 방법은 클러스터 간의 거리 계산 방식을 다양하게 적용할 수 있어 유연성이 높으며, 계층적인 트리 형태의 군집 구조를 만들어낸다.

   두 방법을 비교한 결과, 병합 군집 알고리즘을 이용한 클러스터링이 더 나은 결과를 제공하였기에 이를 최종적으로 선택하였다. 병합 군집 알고리즘은 계층적인 트리 형태로 군집을 이루기 때문에 깊이 우선 탐색(DFS)과 유사한 방식으로 세부적인 탐색이 가능하여, 다양한 화합물 후보를 효율적으로 탐색할 수 있다.

3. **MEMES를 이용한 탐색**

MEMES(Machine learning framework for Enhanced MolEcular Screening)는 Sarvesh Mehta등이 제시한 머신러닝 기반의 가상 스크리닝 프레임워크로, 베이지안 최적화를 통해 대규모 약물 라이브러리에서 잠재적인 신약 후보 물질을 효율적으로 식별하는 알고리즘이다.

탐색 과정은 먼저 클러스터링을 통해 탐색 공간을 균형있게 커버할 수 있도록 각 클러스터에서 일정 수의 리간드를 무작위 선택하는 것으로 시작한다. 선택된 초기 리간드는 Surrogate model의 학습 데이터로 쓰이며, 이 모델을 이용해 리간드의 도킹 점수를 예측한다. 이후, 학습된 모델을 기반으로 Acquisition Function을 사용해 다음 탐색할 리간드를 선택한다. 여기서는 Surrogate Model로 GPR(Gaussian Process Regression)을 사용하였고, Acquisition Function으로 EI(Expected Improvement)를 사용하였다. 이 과정은 최적의 리간드를 반복적으로 탐색하며 모델을 업데이트하는 방식으로 진행된다. 각 반복에서 새로운 리간드를 선택하고 도킹 점수를 예측해 탐색이 점진적으로 개선된다. 최종적으로, 모델에서 예측 점수가 높은 리간드를 선택하여 최종 결과를 도출한다.

현재까지 테스트해본 결과, MEMES를 이용한 탐색 방식이 추천한 리간드들이 가장 높은 평균 도킹 스코어를 보이는 것으로 나타났다. 이러한 결과를 바탕으로, 우리 팀은 현재 MEMES 기반 탐색 알고리즘을 주요 탐색 방법으로 채택하였다.

2.2 배포 방식

현재 프로젝트는 윈도우 환경에서 원활한 실행이 어려워 플랫폼 종속성 문제가 존재한다. 이를 해결하기 위해 도커(Docker)를 이용하기로 하였다. 도커는 애플리케이션을 컨테이너로 묶어 일관된 실행 환경을 제공하는 도구로, 플랫폼에 상관없이 동일한 환경에서 실행할 수 있게 한다.

컨테이너는 리눅스 컨테이너를 사용할 예정이며, 이를 통해 리눅스 환경에서 테스트된 애플리케이션을 윈도우 및 기타 운영체제에서도 문제없이 실행할 수 있게 된다. 또한 도커를 이용함으로써 환경 설정 문제를 최소화할 수 있다.

2.3 사용자 인터페이스

CLI(Command Line Interface) 방식은 일반적인 사용자에게 불친절하고 사용이 어려운 단점

이 있다. 따라서 Django 프레임워크를 이용해 백엔드와 프론트엔드를 처리할 예정이다. Django는 파이썬 기반의 웹 프레임워크로, 빠르고 효율적인 웹 애플리케이션 개발을 지원한다.

사용자는 도커 컨테이너를 실행한 후 웹 브라우저를 통해 접근할 수 있는 GUI(Graphical User Interface)를 제공받게 된다. 이를 통해 사용자는 직관적이고 편리하게 시스템을 사용할 수 있으며, 복잡한 명령어 입력 없이도 쉽게 화합물 탐색 작업을 수행할 수 있다.

3. 갱신된 과제 추진 계획

3.1 갱신된 계획

- **개발 환경**

  1만개의 화합물을 이용할 계획이었으나, 1만개도 알고리즘을 테스트 하기에는 너무 많아 약 4천개의 화합물을 이용하는 것으로 수정하였다. 기존 계획과 마찬가지로 추후에 더 많은 화합물을 이용하여 적용할 것이다.

- **알고리즘 분석 및 개선**

  모든 화합물을 전부 도킹하기에는 너무 많은 시간과 전력이 소모되고, 그런 이유 때문에 알고리즘을 개선하는 것이므로 완전 탐색은 구현하지 않았다.

  무작위 탐색과 클러스터링 알고리즘 구현 이후 MEMES 알고리즘을 이용한 방법을 시도했다. 중간 보고서 이후 한 두가지의 알고리즘을 더 시도할 계획이다.

- **웹 애플리케이션 개발**

  알고리즘 분석에 좀 더 집중하기 위해 웹 애플리케이션 개발 일정을 기존보다 늦췄다. 웹 UI 개발에는 Django 프레임워크를 사용하고, Docker를 이용해 배포하기로 결정되었다.

  지금까지 전반적인 UI 디자인을 완성하였으며, 리간드를 추가하고 리간드의 목록을 볼 수 있는 기능을 구현하였다. 앞으로 단백질을 입력하고, 원하는 방법을 선택하여 결과를 시각적으로 볼 수 있도록 개발할 예정이다.

- **자문의견서 반영**

  전문가 피드백에는 효율적인 클러스터링 또는 스크리닝 탐구, unsupervised learning 적용 등의 내용이 있었다. 이를 반영해 계층적 클러스터링 외에 다른 방법이 있는지 찾고, MEMES와 같이 unsupervised learning 방식을 적용하고 개선을 시도해 볼 예정이다.

3.2 추진 일정

<table>
<tr><th colspan="2">5월</th><th colspan="4">6월</th><th colspan="5">7월</th><th colspan="5">8월</th><th colspan="4">9월</th><th colspan="3">10월</th></tr>
<tr><th>4</th><th>5</th><th>1</th><th>2</th><th>3</th><th>4</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>1</th><th>2</th><th>3</th><th>4</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td colspan="2">착수 보고서 작성</td><td colspan="4"></td><td colspan="5"></td><td colspan="5"></td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td></td><td colspan="3">개발 환경 구축</td><td colspan="4"></td><td colspan="5"></td><td colspan="5"></td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="4">무작위 탐색 구현</td><td colspan="5"></td><td colspan="5"></td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="3"></td><td colspan="4">클러스트링 구현</td><td colspan="5"></td><td colspan="5"></td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="5"></td><td colspan="3">MEMES 알고리즘 구현</td><td colspan="5"></td><td colspan="5"></td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="5"></td><td colspan="4">중간 보고서 작성</td><td colspan="5"></td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="5"></td><td colspan="5"></td><td colspan="3">알고리즘 분석 및 개선</td><td colspan="4"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="5"></td><td colspan="5"></td><td colspan="5">웹 애플리케이션 개발 & 시각화</td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="5"></td><td colspan="5"></td><td colspan="5"></td><td colspan="3"></td><td colspan="3">최종 보고서 작성</td></tr>
</table>

4. 구성원별 진척도

이름	진척도
이정민	보고서 작성 개발 환경 구축 및 관리 AutoDockTools, ChimeraX 등을 활용한 프로틴, 리간드 라이브러리 구축 spicy.cluster.hierarchy를 활용한 계층적 클러스터링 구현
성가빈	보고서 작성 rdkit을 이용한 clustering 구현 MO-MEMES 소스코드 수정 및 MEMES 알고리즘 테스트 웹 애플리케이션 UI 디자인 및 구현
최우영	보고서 작성 Docker와 Django를 이용해 환경 구축 리간드 데이터베이스 구축 및 리간드 추가 기능 구현

5. 과제 수행 내용 및 중간 결과

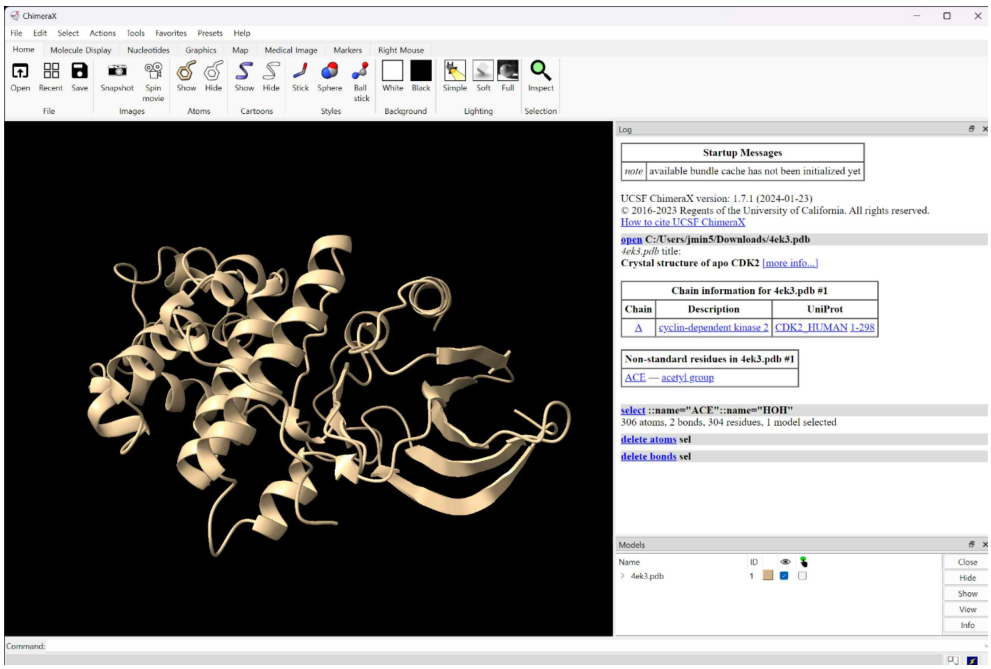
5.1 환경 구성

1. 단백질-리간드 결합 예측 모델 선택

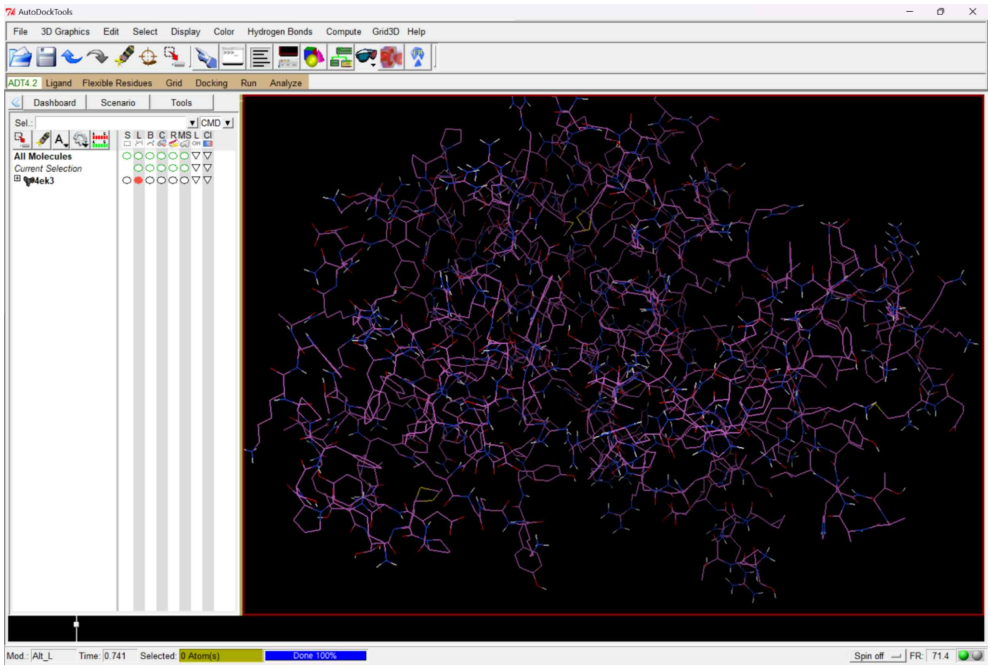
   WSL 환경에서 오픈 소스 분자 도킹 프로그램인 Autodock Vina 모델을 활용한다. Autodock Vina로 단백질과 리간드의 pdbqt 파일을 사용해 둘 사이의 결합 점수를 측정할 수 있다.

2. 단백질 파일 생성

   RCSB PDB 사이트에서 pdb 파일을 다운로드 한 후, ChimeraX 툴을 활용하여 pdb 파일을 정리한다. 비표준 잔류물과 추가 체인을 제거한다.

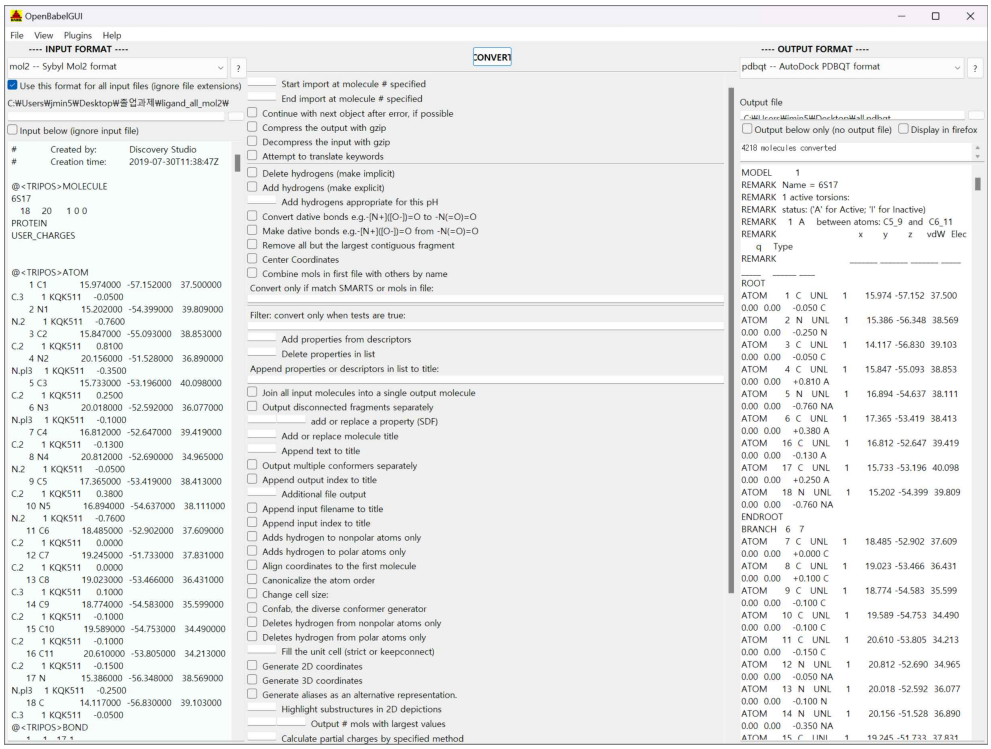


   AutoDockTools 툴을 활용하여 pdb 파일을 pdbqt 파일로 변환한다. 우선 물을 제거하고 수소를 추가한 후, 무극성 수소를 병합한다. 그리고 콜먼 전하를 추가하고, 마지막으로 AD4 타입 원자를 배치하여 pdbqt 파일로 변환한다.



3. 리간드 파일 생성

OpenBabel GUI 툴을 이용하여 다운로드 받은 mol2 파일을 pdbqt 파일로 변환한다.



위 변환으로 4412개의 리간드 정보가 담긴 하나의 pdbqt 파일이 생성된다. "vina_split --input all.pdbqt --ligand ligand"의 명령어를 사용하여 리간드 pdbqt파일을 분할해 4412개의 리간드 파일을 생성한다.

4. None 리간드 제거

분자를 표현하는 방법 중 하나인 SMILES(Simplified Molecular-Input Line-Entry System) 표기법에는 두가지 방식이 존재한다. 어떤 분자를 "c1(nc(c([nH]1)C1=[C][C]=N[C]=[C]1)C1=[C][C]=C([C]=[C]1)F)C1=[C][C]=C([C]=[C]1)[S@@](=O)[C]" 혹은 "CS(=O)c1ccc(-c2nc(-c3ccc(F)cc3)c(-c3ccncc3)[nH]2)cc1"로 표현할 수 있다.

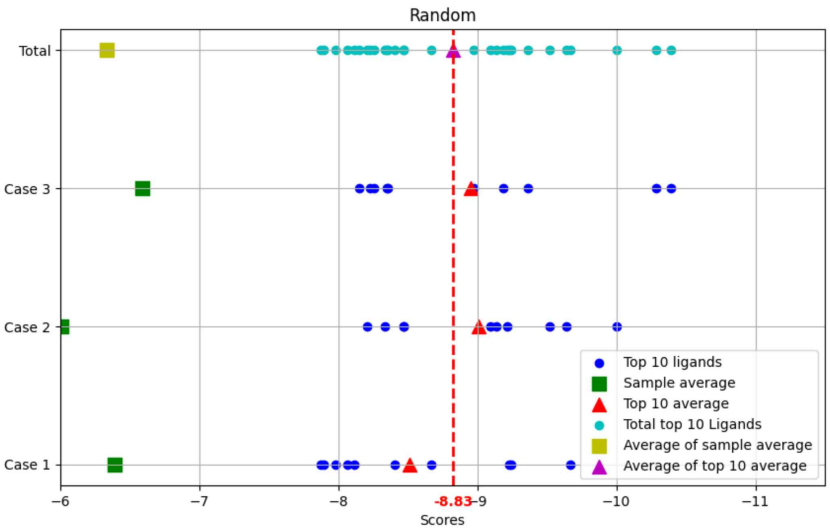
모든 리간드가 첫번째 방식으로 표현 가능하지만 두번째 방식으로 표현되지 않는 리간드들이 존재한다. 두번째 방식을 활용해 유사도를 측정할 예정이므로 해당 방식으로 변환되지 않는 리간드 162개를 제거하여 4250개의 리간드를 사용한다.

	평균 점수
4250개 중 3346개 도킹 결과	-6.72

5.2 무작위 탐색

무작위 40개의 리간드를 선별해 도킹 후 상위 10개를 추천한다.

Top10 점수	Case 1	Case 2	Case3	평균
무작위	-8.51	-9.01	-8.96	-8.83



Top10 유사도	Case 1	Case 2	Case3	평균
분자 지문 기반	0.48	0.41	0.44	0.443
최대 공통 부분구조 기반	0.21	0.24	0.21	0.220

5.3 화합물 클러스트링 및 탐색

- **K-means 클러스터링**

  클러스터링 결과 하나의 클러스터에 절반 이상의 리간드가 모이는 등의 불균형 문제가 발생하였기에, 크기가 작은 클러스터를 병합하여 15개의 균일한 클러스터를 구성해 진행했다.

  각각의 클러스터에서 리간드를 2개씩 선별해 도킹 후 평균 점수를 클러스터에 배정하고 이후 10번 최상위 점수의 클러스터에서 하나씩 뽑아 도킹 후 클러스터 점수 갱신하여, 총 40번의 도킹 후 상위 10개의 리간드를 추천한다. 그러나 작은 클러스터 병합 결과 클러스터링의 의미가 퇴색되는 문제가 발생했다.

Top10 점수	Case 1	Case 2	Case3	평균
k-means 클러스터링	-9.28	-8.63	-8.68	-8.86

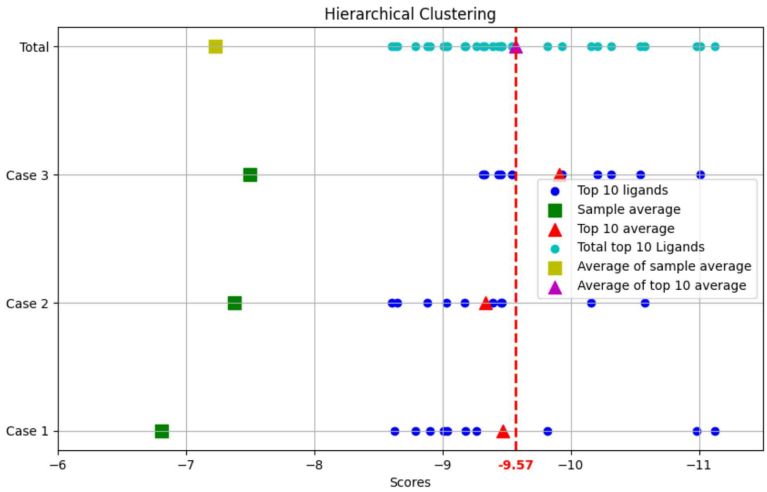
Top10 유사도	Case 1	Case 2	Case3	평균
분자 지문 기반	0.52	0.49	0.47	0.493
최대 공통 부분구조 기반	0.21	0.25	0.23	0.230

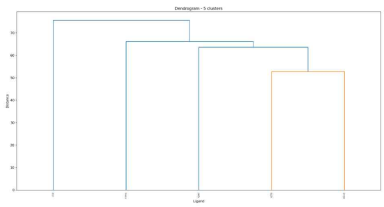
- **계층적 클러스터링**

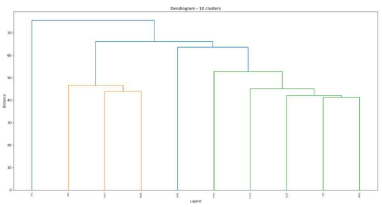
  클러스터 병합에 의한 클러스터링 성능 저하를 보완하기 위해 큰 클러스터에서 작은 클러스터로 내려가는 방법을 사용하여 클러스터링 성능이 저하되지 않도록 하였다. 계층적 클러스터링 메트릭스 Z를 만들고 지정된 개수에 따라 점진적으로 잘린 클러스터를 구성한다.

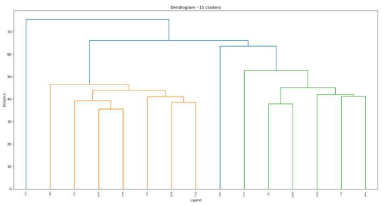
각 클러스터에서 리간드를 하나씩 뽑아 도킹 후, 최고 점수 클러스터에 속하지 못한 리간드를 배제하는 방식을 반복한다. 마지막 최고점 클러스터에서 총합 40번의 도킹이 될 때가지 도킹 후 상위 10개의 리간드를 추천한다.

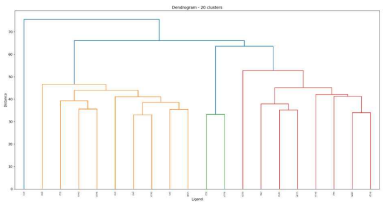
Top10 점수	Case 1	Case 2	Case3	평균
계층적 클러스터링	-9.47	-9.34	-9.91	-9.57

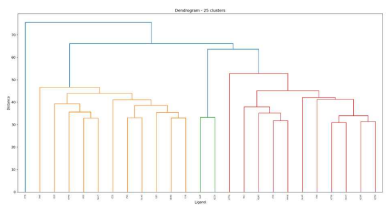


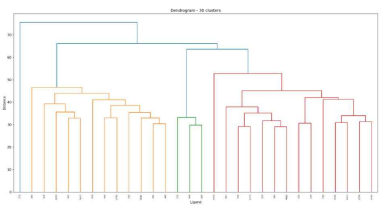












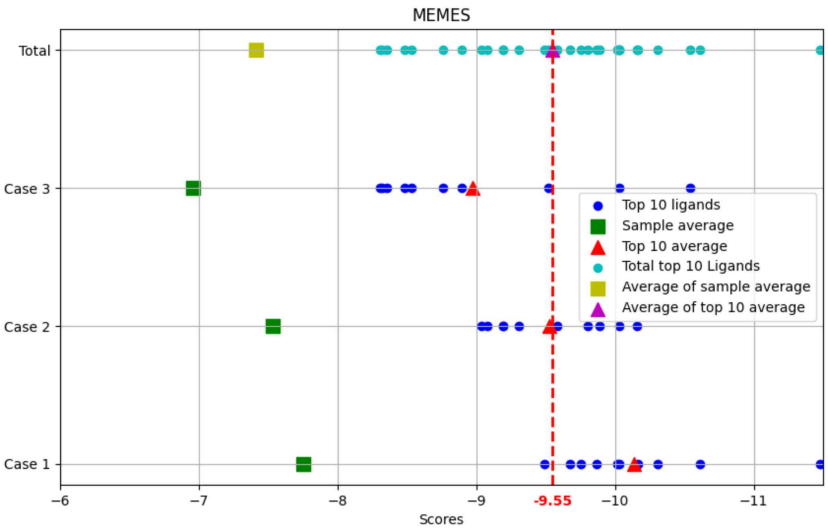
Top10 유사도	Case 1	Case 2	Case3	평균
분자 지문 기반	0.53	0.67	0.44	0.550
최대 공통 부분구조 기반	0.21	0.19	0.31	0.237

5.4 MEMES를 이용한 탐색

MEMES 알고리즘은 클러스터링을 활용해 초기 리간드의 점수를 계산 후 가우시안 회귀 기반 베이지안 최적화를 반복하면서 모델을 학습해 최적의 리간드를 찾는 방식이다. MEMES 저자의 GitHub에 공유된 MO-MEMES 소스코드를 상황에 맞게 수정하여 사용하였다.

아래의 결과는 10개 클러스터로 구성된 클러스터링 파일을 사용해 각 클러스터에서 1개씩 총 10개의 리간드를 초기 리간드로 사용하고, 5번의 반복에서 5개의 새로운 리간드를 선택한 결과이다. 마지막 GP모델에서 높은 예측치를 가지는 리간드를 탐색할 때, 앞에서 탐색하지 않았던 리간드를 5개 뽑을 때까지 상위 리간드를 순서대로 탐색하였다. 이로 인해 총 탐색한 리간드의 개수가 35~40개 사이로 일정하지 않게 되는 문제가 발생한다. 케이스 1과 2는 40개의 리간드를 탐색했고, 케이스 3는 39개의 리간드를 탐색했다. 또한 결과를 보면 실행별 편차가 큰 것을 확인할 수 있다. 탐색의 최대치로 정한 40이 너무 작아 발생한 것으로 판단된다.

Top10 점수	Case 1	Case 2	Case3	평균
MEMES	-10.14	-9.53	-8.98	-9.55



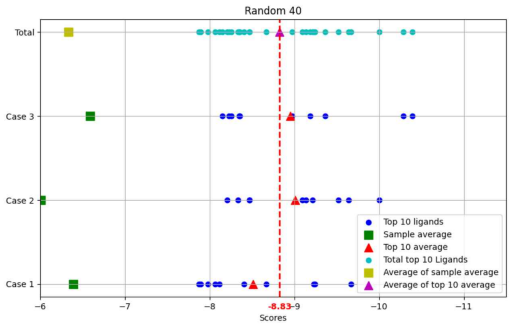
Top10 유사도	Case 1	Case 2	Case3	평균
분자 지문 기반	0.51	0.48	0.63	0.540
최대 공통 부분구조 기반	0.26	0.25	0.28	0.263

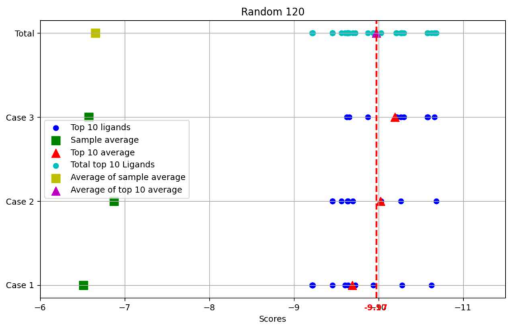
5.5 탐색의 최대치 확장

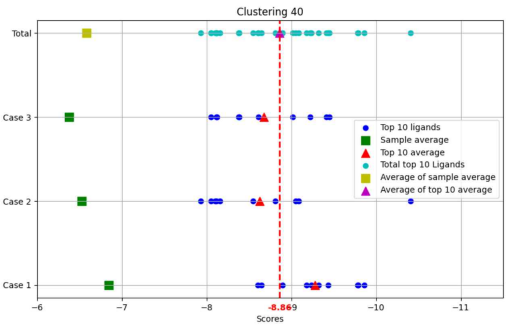
현재 리간드 라이브러리의 크기는 약 4000개이다. 초기 목표는 이보다 훨씬 큰 수십억개였으나 라이브러리 구성문제, 컴퓨팅파워 부족 문제 등의 문제로 우선 작은 라이브러리로 시작하였다. 수십억개의 1퍼센트는 수천만개이지만, 4000개의 1퍼센트는 40개이다. 최대한 적게 탐색하고자 하였으나, 40은 너무 작다고 판단하여 120으로 늘려 다시 탐색하였다.

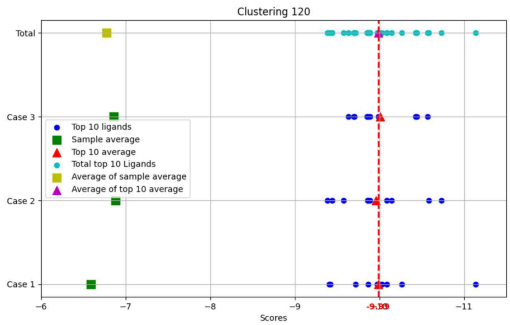
클러스터링은 각 클러스터에서 2개씩 선별하던 것을 7개씩 선별하였고 이후 15번은 그 순간의 최상위 클러스터에서 선별하였다. 계층적 클러스터링은 각 클러스터에서 1개씩 선별하던 것을 5개씩 선별하였고 마지막 레벨 최상위 클러스터에서 남은 횟수만큼 선별하였다. MEMES는 기존 10개의 클러스터에서 각 1개씩 선별 후 5번의 반복에서 5개씩 선별하던 방식에서 20개의 클러스터에서 각 2개씩 선별 후 4번의 반복에서 15개씩 선별하는 방식으로 변경하였다.

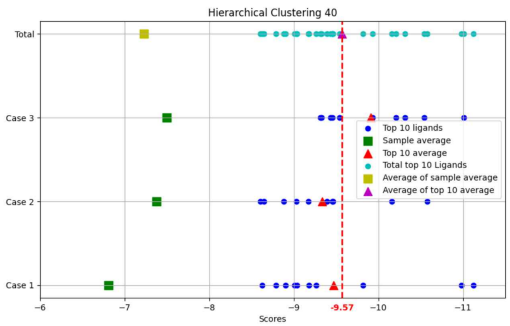
Top10 평균 점수	무작위	클러스터링	계층적 클러스터링	MEMES
40번 탐색	-8.83	-8.86	-9.57	-9.55
120번 탐색	-9.97	-9.99	-10.10	-10.34

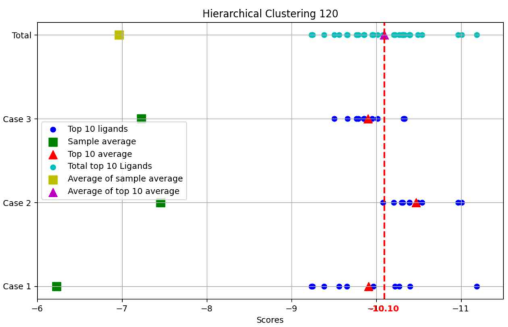


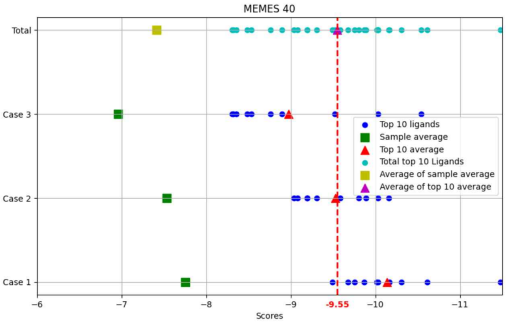


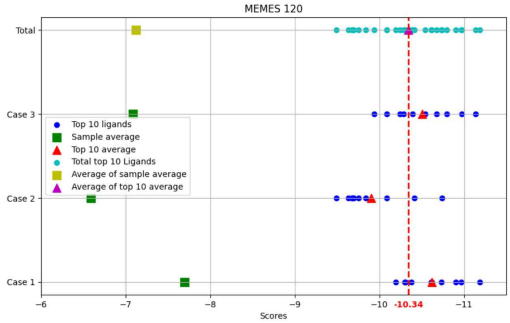












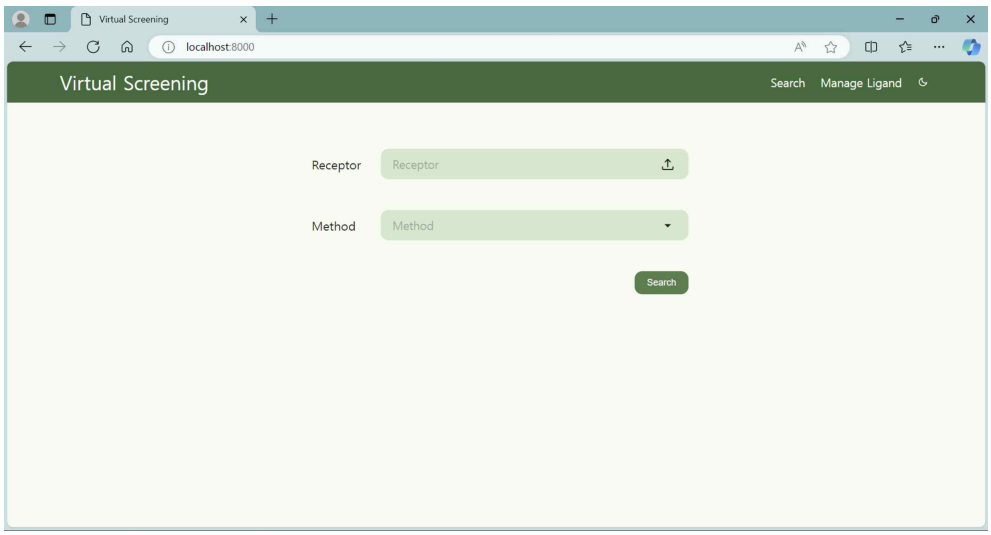
5.6 웹 애플리케이션 개발

사용자들이 쉽게 서비스를 이용할 수 있도록 웹 애플리케이션의 형태로 제공한다. Django 프레임워크로 개발하고 있으며 Docker를 이용해 배포할 예정이다.

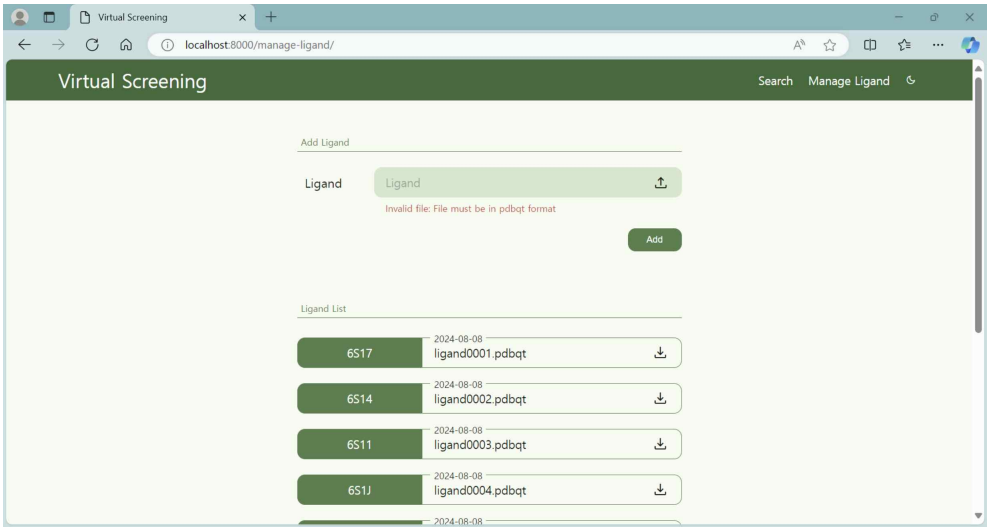
사용자는 필요한 라이브러리를 설치할 필요없이 GitHub 레포지토리를 로컬로 clone한 다음 "docker-compose up" 명령어를 실행하면 이용할 수 있다. 컨테이너 시작 시 Django에서 migrate가 이루어지고 서버가 실행된다.

```
bini@MY-DESKTOP:~/깃허브/PNUCSE_2024-1_team8/ligand_search$ docker-compose up
WARN[0000] /home/bini/깃허브/PNUCSE_2024-1_team8/ligand_search/docker-compose.yml: `version` is obsolete
[+] Running 2/2
 ✔ Network ligand_search_default  Created                                    0.1s
 ✔ Container ligand_search-app-1  Created                                    0.1s
Attaching to app-1
app-1  | Operations to perform:
app-1  |   Apply all migrations: admin, auth, contenttypes, sessions, virtual_screening
app-1  | Running migrations:
app-1  |   No migrations to apply.
app-1  | Watching for file changes with StatReloader
app-1  | Performing system checks...
app-1  |
app-1  | System check identified no issues (0 silenced).
app-1  | August 09, 2024 - 11:54:53
app-1  | Django version 5.0.7, using settings 'app.settings'
app-1  | Starting development server at http://0.0.0.0:8000/
app-1  | Quit the server with CONTROL-C.
app-1  |
```

로컬호스트로 접속하면 첫 화면에서 단백질 파일을 업로드하고 탐색 방법을 선택할 수 있다. 지금은 과제에서 구현한 random, clustering, MEMES를 선택할 수 있고, 앞으로 구현하는 알고리즘을 추가할 계획이다. 이후 버튼을 누르면 선택한 방법에 따라 결과를 보여준다. 추천하는 리간드를 확인하고, 업로드한 단백질과 상위의 리셉터의 결합을 시각화하여 볼 수 있도록 구현할 것이다.



상단 버튼을 눌러 데이터베이스에 저장된 리간드를 관리하는 페이지로 이동할 수 있다. 리간드 파일을 업로드하고 Add 버튼을 누르면 데이터베이스에 저장된다. 정확하지 않은 파일을 업로드할 경우 경고 메시지가 뜨고 저장할 수 없다. 그 아래로는 현재 데이터베이터에 저장된 리간드 리스트를 보여준다.



6. 참고 자료 및 출처

- MEMES
  : Chem. Sci., 2021,12, 11710-11721

- MO-MEMES
  : Mehta S, Goel M and Priyakumar UD (2022) MO-MEMES: A method for accelerating virtual screening using multi-objective Bayesian optimization. Front. Med. 9:916481. doi: 10.3389/fmed.2022.916481
  : MO-MEMES 소스코드

- 베이지안 최적화
  : Chem. Sci., 2021, 12, 7866

- AutoDock Vina
  : https://github.com/ccsb-scripps/AutoDock-Vina
  : J. Eberhardt, D. Santos-Martins, A. F. Tillack, and S. Forli. (2021). AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings. Journal of Chemical Information and Modeling.
  : O. Trott, A. J. Olson, AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading, Journal of Computational Chemistry 31 (2010) 455-461

- 사용한 단백질 파일
  : https://www.rcsb.org/structure/4ek3

- 단백질 파일 생성에 사용한 MGL tool
  : https://ccsb.scripps.edu/mgltools/downloads/
  : 사용 영상