Pusan National University
Computer Science and Engineering
Technical Report 2024-10

신약 개발 후보물질 발굴을 위한 거대 화합물 라이브러리 탐색 최적화

201924656 이정민

202155565 성가빈

201845928 최우영

지도교수 송길태

목 차

1. 서론 1

1.1. 연구 배경 1

1.2. 기존 문제점 1

1.3. 연구 목표 2

2. 연구 배경 3

2.1. 배경 지식 3

2.1.1. 가상 스크리닝 3

2.1.2. AutoDock Vina 3

2.1.3. k-means 클러스터링 4

2.1.4. 병합 군집 클러스터링 4

2.1.5. 베이지안 최적화 5

2.1.6. MEMES 알고리즘 6

2.1.7. Diffdock 6

2.2. 연구 일정 및 구성원 역할 8

2.2.1. 연구 일정 8

2.2.2. 구성원 별 역할 9

3. 연구 내용 10

3.1. 개발 환경 구축 10

3.2. 4000개의 리간드 사용 13

3.2.1. 무작위 탐색 13

3.2.2. k-means 클러스터링 14

3.2.3. 계층적 클러스터링 14
3.2.4. MEMES 알고리즘 16
3.3. 210만개의 리간드 사용 18
3.3.1. 무작위 탐색 18
3.3.2. k-means 클러스터링 19
3.3.3. MEMES 알고리즘 19
3.3.4. 샘플 표본 축소 21
3.3.5. 웹 애플리케이션 탑재 22
4. 연구 결과 분석 및 평가 24
5. 결론 및 향후 연구 방향 27
6. 참고 문헌 28

1. 서론

1.1. 연구 배경

신약 개발은 막대한 비용과 오랜 기간이 소요되는 어려운 과정이다. 그중에서도 가장 중요하면서도 큰 난관으로 작용하는 단계는 효과적인 후보물질을 발굴하는 것이다. 수억 개가 넘는 거대한 화합물 라이브러리에서 특정 타겟 단백질과 잘 결합할 수 있는 화합물을 찾아내는 일은 실로 시간이 많이 소요되는 과정이다.

전통적으로는 모든 화합물과 타겟 단백질 간의 결합 여부를 실험실에서 하나씩 실험을 통해 검사하는 방식을 사용해왔다. 하지만 이러한 접근 방법은 비용과 시간이 지나치게 많이 들어 효율성이 현저히 낮은 문제가 있다. 수억 개의 화합물을 모두 실험해보기에는 현실적인 한계가 있었다.

이런 상황에서 컴퓨터 기반의 가상 스크리닝 기술과 인공지능 기법이 등장하면서 상당한 진전이 있었다. 가상 스크리닝 기술은 화합물과 타겟 단백질 간의 상호작용을 컴퓨터 시뮬레이션을 통해 예측하는 기술로, 기존 실험실 기반 방식에 비해 시간과 비용 측면에서 획기적인 절감 효과가 있었다. 그러나 이러한 절감에도 불구하고 거대한 규모의 화합물 라이브러리를 매번 전부 탐색하기에는 여전히 오랜 시간이 소요되는 문제가 있다.

1.2. 기존 문제점

1.2.1. 거대한 화합물 라이브러리의 크기

화합물 라이브러리의 규모가 수억 개에 달하기 때문에, 이를 전부 탐색하는 것은 현실적으로 불가능하다. 따라서 효과적으로 탐색 공간을 축소하고 유망한 화합물을 빠르게 찾아내는 전략이 필요하다.

1.2.2. 탐색 결과의 다양성 확보

탐색 과정에서 발견된 화합물들이 서로 유사한 구조를 가질 경우, 실제 실험 단계에서 한 화합물이 효과가 없으면 유사한 화합물들도 효과가 없을 가능성이 높다. 따라서 구조적 다양성을 유지하면서도 결합 확률이 높은 화합물을 선별하는 것이 중요하다.

1.2.3. 알고리즘의 확장성과 효율성

사용되는 인공지능 및 최적화 알고리즘은 거대한 화합물 라이브러리에 대해서도 확장 가능하고 효율적으로 동작해야 한다. 따라서 알고리즘의 시간 및 공간 복잡도를 고려하여 설계하고 구현해야 한다.

1.3. 연구 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.

2. 연구 배경

2.1. 배경 지식

2.1.1. 가상 스크리닝

단백질 구조 기반의 신약 개발 과정은 다음과 같이 진행된다. 먼저 질병의 원인이 되는 타겟 단백질의 구조를 결정해야 한다. 그리고 타겟 단백질과 결합하여 원하는 효과를 내는 화합물을 찾는다. 다음으로 찾아낸 화합물을 선도 물질(lead)화 하고, 선도 물질을 최적화한다.[1]

타겟 단백질과 결합하는 리간드를 찾는 과정에서 가상 스크리닝이 적용된다. 가상 스크리닝은 화합물과 타겟 단백질 간의 상호작용을 컴퓨터 시뮬레이션을 통해 예측하는 기술이다. 가상 스크리닝에서는 많은 화합물 라이브러리에서 점차적으로 수를 줄여간다. 처음에 타겟 단백질의 특성을 고려해 화합물을 선별하고, 거짓 양성을 주는 화합물을 제외한다. 그 다음으로 도킹등의 방법을 통해 후보 물질을 찾는다. 분자 도킹은 화합물이 타겟 단백질과 결합하는 방법을 알아보는 것으로, 역장(forcefield) 기반이나 경험(empirical) 기반의 스코어링 함수가 사용된다.[1]

2.1.2. AutoDock Vina

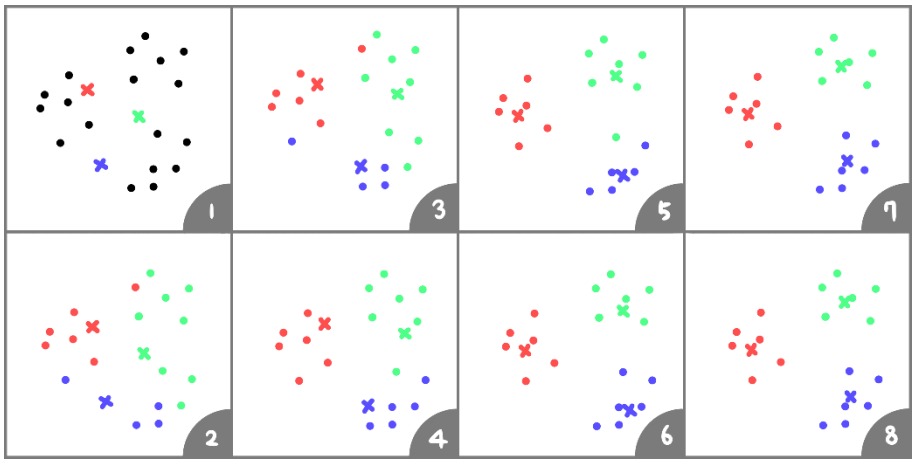
AutoDock Vina[2][3]는 분자 도킹을 위한 오픈 소스 프로그램이다. Oleg Trott가 설계하고 구현하였으며, 지금은 Forli Laboratory에서 유지 보수 및 개발 중이다.[4]

AutoDock Vina로 단백질과 리간드의 결합을 예측하기 위해서는 타겟 단백질(리셉터)와 리간드의 PDBQT 파일이 필요하다. Vina forcefield를 사용할 경우, 결합도 그리드 맵을 계산할 수 있도록 탐색 공간의 중심과 사이즈를 지정해야 한다.[4]

Vina에서 결합 점수(도킹 점수)는 반데르발스 힘, 수소 결합, 수소성, conformational entropy penalty등을 고려한 자유에너지로 계산된다.[3] 결합 점수가 낮을수록 결합할 확률이 더 높다고 할 수 있다. 점수는 리간드가 리셉터와 어떤 구조(포즈)로 결합하는지에 따라 달라진다. 도킹 시뮬레이션을 실행하면, 몬테카를로 샘플링에서 지정된 만큼의 포즈를 얻고 각 포즈에서의 점수가 계산된다. 최종적으로 에너지가 낮은 순서대로 순위가 매겨져 출력된다.[4]

2.1.3. k-means 클러스터링

K-means 알고리즘은 주어진 데이터 포인트들을 K개의 클러스터로 나누는 비지도 학습 기법이다. 비지도 학습에서는 데이터 포인트들의 라벨을 지정하지 않고, 데이터의 패턴을 파악하게 한다.



[그림 1] k-means 클러스터링 과정

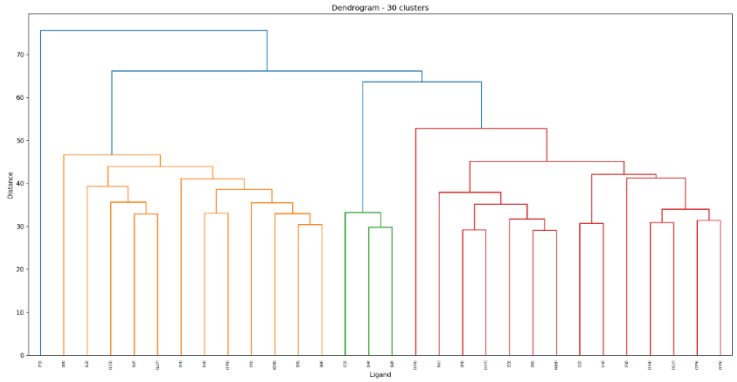
먼저 k개의 초기 클러스터 중심점(centroid)을 설정한다. 각 데이터 포인트는 가장 가까운 클러스터 중심점으로 할당되고, 나눠진 클러스터에서 중심점이 업데이트된다. 해당 과정을 중심점의 위치가 바뀌지 않을 때까지 반복한다. 이 방법은 계산 속도가 빠르지만, 초기 클러스터 중심점 설정에 따라 결과가 달라질 수 있다.

2.1.4. 병합 군집 클러스터링

병합 군집 알고리즘은 데이터 포인트들을 개별 클러스터로 시작하여, 가장 가까운 두 클러스터를 반복적으로 병합해 나가는 계층적 군집 기법이다.

클러스터 간의 거리 계산 방식에는 single, complete, average, centroid linkage 등이 존재한다. single linkage는 두 클러스터 간의 가장 가까운 거리를 사용하며, complete linkage는 가장 먼 거리를 사용한다. 그리고 average linkage는 각 클러스터의 모든 포인트 간의 거리의 평균을 사용한다. centroid linkage는 클러스터의 중심점 간의 거리를 사용하는 방식이다.

이처럼 다양한 거리 계산 방식을 적용할 수 있어 병합 군집 알고리즘은 유연성이 높다. 병합 군집 알고리즘의 결과로는 계층적인 트리 형태의 군집 구조를 만들어진다.



[그림 2] 병합 군집 클러스터링으로 만들어진 군집 구조

2.1.5. 베이지안 최적화

베이지안 최적화(bayesian optimization)는 알려지지 않은 목적 함수의 값을 최대 또는 최소로 만드는 입력 값을 찾는 것을 목표로 한다. 이 방법은 하이퍼파라미터 최적화에 주로 사용되며, 이 경우 하이퍼파라미터가 입력 값이 된다.

베이지안 최적화는 surrogate model과 acquisition function으로 구성되고, 이 두 과정을 반복하면서 목적 함수가 최대 또는 최소가 되는 지점을 찾게 된다. Surrogate model은 이전의 입력 값과 그에 따른 목적 함수의 값들을 이용해 목적 함수를 확률적으로 추정하며, GPR(Gaussian Process Regression)이 주로 사용된다. Acquisition function은 Surrogate model의 확률적 추정을 바탕으로 다음에 사용할 입력 값들을 추천한다.[5][6]

Acquisition function으로는 EI(Expected Improvement), PI(Probability of Improvement), UCB(Upper Confidence Bound) 등이 사용될 수 있다.[5][7] EI는 기대할 수 있는 목적 함수 값의 개선 정도가 가장 큰 다음 입력 값을 찾는다. f^*가 현재 가장 좋은 값일 때, $EI(x) = exp[max(0, f(x) - f^*)] = (\mu(x) - f^* - \zeta)CDF(\frac{\mu(x)-f^*-\zeta}{\sigma(x)}) + \sigma(x)PDF(\frac{\mu(x)-f^*-\zeta}{\sigma(x)})$ 로 계산된다.[5][7][8][9][10] PI는 선택했을 때 목적 함수 값이 개선될 가능성이 가장 큰 다음 입력 값을 찾는다. $PI(x) = P(max(0, f(x) - f^*) > 0) = CDF(-\frac{f^*-\mu(x)}{\sigma(x)})$ 로 나타낼 수 있다.[5][7][8][9][10] UCB는 $UCB(x) = \beta\sigma(x) + \mu(x)$으로 계산되며, β값이 클수록 이전의 좋은 값 근처보다 아직 잘 모르는 값에 비중을 두고 입력 값을 선택한다.[5][7]

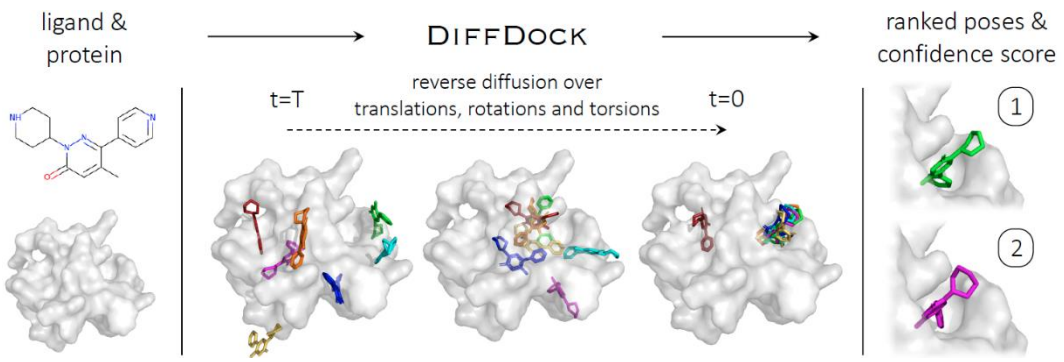
2.1.6. MEMES 알고리즘

MEMES(Machine learning framework for Enhanced MolEcular Screening)[11]는 Sarvesh Mehta등이 제시한 머신러닝 기반의 가상 스크리닝 프레임워크로, 베이지안 최적화를 통해 대규모 약물 라이브러리에서 잠재적인 신약 후보 물질을 효율적으로 식별하는 알고리즘이다.[11]

탐색 과정은 탐색 공간을 균형있게 커버할 수 있도록, 클러스터링된 리간드들을 이용해 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 GPR을 사용하였고, Acquisition Function으로 EI를 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.[11]

2.1.7. Diffdock

Diffdock[12]은 프로틴-리간드 도킹을 위한 딥러닝 기반 방법론으로 확산 모델(diffusion model)을 사용하여 단백질 결합 부위에 리간드의 3D 구조와 위치를 생성할 수 있다. Autodock Vina와 같이 분자 도킹 예측 모델이지만 둘은 접근 방식에 차이가 존재한다. Vina는 전통적인 분자 도킹 알고리즘을 사용하며, 주로 물리 기반 스코어링 함수와 최적화 알고리즘을 활용하지만 Diffdock은 확산 모델을 사용해 결합 예측을 더 정확하게 할 수 있다. [12]



[그림 3] DiffDock

확산 모델은 생성 모델(generate model)의 일종으로, forward diffusion process와 reverse diffusion process를 통해 학습이 이루어진다. forward diffusion process는 원본 데이터에 가우시안 노이즈를 점차 더해가는 과정이고, reverse diffusion process는 반대로 노이즈가 있는 데이터에서 원본 데이터를 찾아가는 과정이다. forward diffusion process $q(x_t|x_{t-1})$를 기반으로, $q(x_{t-1}|x_t)$를 근사하는 reverse diffusion precess $p(x_{t-1}|x_t)$를 학습하게 된다. [13][14][15]

중간 보고 이후 확산 모델을 사용해보라는 피드백을 받아 과제에 맞는 확산 모델 Diffdock을 찾아 적용을 시도하였지만, Diffdock 자체는 최적화 알고리즘을 직접 지원하지 않아 과제의 방향성과는 맞지 않다고 생각하였다.

2.2. 연구 일정 및 구성원 역할

2.2.1. 연구 일정

5월		6월				7월					8월					9월				10월		
4	5	1	2	3	4	1	2	3	4	5	1	2	3	4	5	1	2	3	4	1	2	3
착수 보고서 작성																						
	개발 환경 구축																					
		무작위 탐색 구현																				
				클러스터링 구현																		
							MEMES 알고리즘 구현															
									중간 보고서 작성													
													탐색 공간 확장									
									웹 애플리케이션 개발 & 시각화													
																			최종 보고서 작성			

2.2.2. 구성원 별 역할

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 MEMES에 Acquisition Function 추가 구현
성가빈	보고서 작성 rdkit을 이용한 clustering 구현 MO-MEMES 소스코드 수정 및 MEMES 알고리즘 테스트 웹 애플리케이션 UI 디자인 및 구현 MEMES에 Acquisition Function 추가 구현
최우영	보고서 작성 Docker와 Django를 이용해 환경 구축 리간드 데이터베이스 구축 및 리간드 추가 기능 구현 웹에서 탐색을 실행할 수 있도록 구현

3. 연구 내용

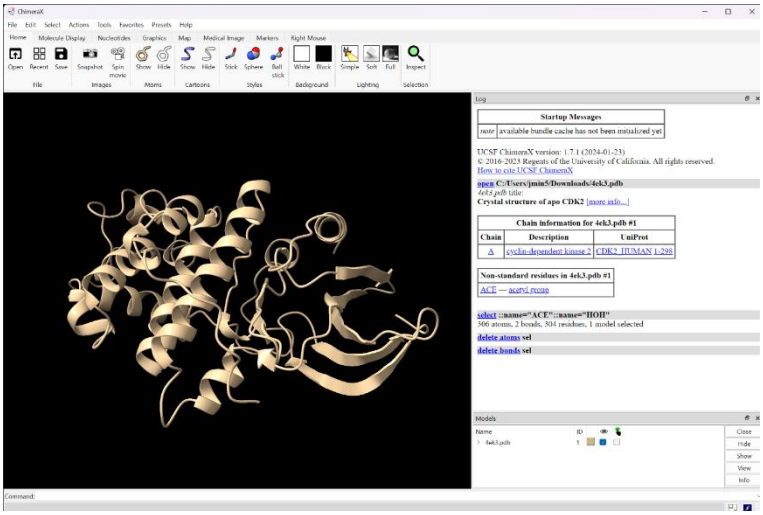
3.1. 개발 환경 구축

3.1.1. 단백질-리간드 결합 예측 모델 선택

가장 먼저 목표 단백질과 리간드 사이의 결합 점수를 확인할 모델을 선택해야 한다. WSL 환경에서 오픈 소스 분자 도킹 프로그램인 Autodock Vina[2][3] 모델을 사용하였다.

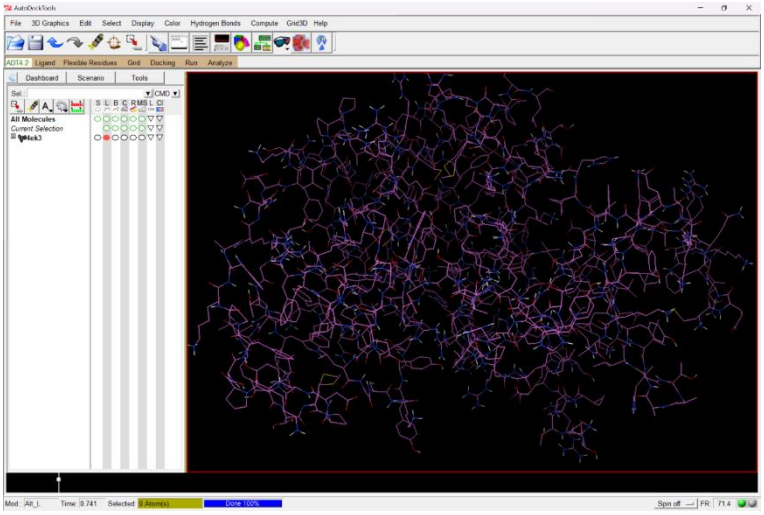
3.1.2. 단백질 파일 생성

Autodock Vina 모델의 입력으로 사용하기 위해, 단백질의 PDBQT 파일이 필요하다. RCSB PDB 사이트[16][17]에서 PDB 파일을 다운로드 한 후 PDBQT 파일로 변환하였다.



[그림 4] ChimeraX를 이용한 PDB 파일 정리

PDBQT 파일로 변환하기 전에, ChimeraX 툴과 AutoDockTools를 이용해 PDB 파일을 정리하는 작업이 필요하다. 우선 ChimeraX를 활용하여 비표준 잔류물과 추가 체인을 제거해주었다. AutoDockTools로는 정리한 PDB 파일에서 물을 제거하고 수소를 추가한 후, 무극성 수소를 병합한다. 그리고 콜먼 전하를 추가하고, 마지막으로 AD4 타입 원자를 배치하여 PDBQT 파일로 변환하였다.[18]

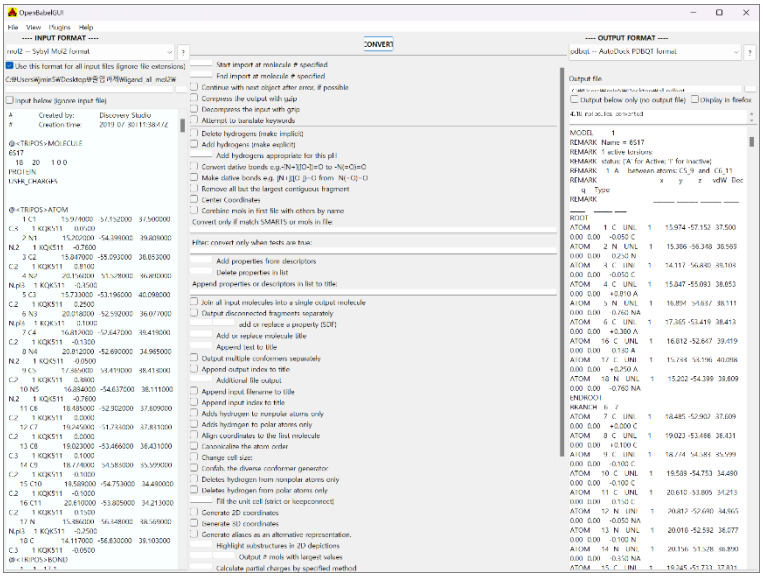


[그림 5] AutoDockTools를 이용한 PDB 파일 변환

3.1.3. 리간드 파일 생성

대용량 화합물 데이터를 사용하기에는 컴퓨팅 자원이 부족하기에 약 4000개의 작은 양의 화합물 데이터를 우선 테스트 한 후, 나중에 더 큰 데이터를 사용해 테스트하는 방향을 선택하였다.

사용할 4000개의 리간드의 mol2 파일을 다운로드 하고, OpenBabel GUI 툴[19][20]을 이용하여 mol2 파일들을 PDBQT 파일로 변환한다. 해당 변환으로 4412개의 리간드 정보가 담긴 하나의 pdbqt 파일이 생성되었다. "vina_split --input all.pdbqt --ligand ligand"의 명령어를 사용하여 리간드 PDBQT 파일을 분할해 4412개의 리간드 파일을 생성했다.



[그림 6] OpenBabel GUI를 이용한 리간드 파일 변환

3.1.4. None 리간드 제거

분자를 표현하는 방법 중 하나인 SMILES(Simplified Molecular-Input Line-Entry System) 표기법에는 두가지 방식이 존재한다. 예를 들어 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.

3.1.5. 유사도 측정 방법 선택

추천된 리간드 사이의 유사도를 측정할 방법으로 Tanimoto 유사도를 선택하였다. Tanimoto 유사도 혹은 Jaccard 유사도는 $\frac{|분자1 \cap 분자2|}{|분자1 \cup 분자2|}$ 로 계산된다. 유사도는 0~1 사이의 수이며, 1에 가까울수록 두 분자는 유사하다. 파이썬 RDKit 라이브러리[21]에서 제공되는 TanimotoSimilarity함수를 사용하여 유사도를 계산할 것이다.[22]

3.2. 4000개의 리간드 사용

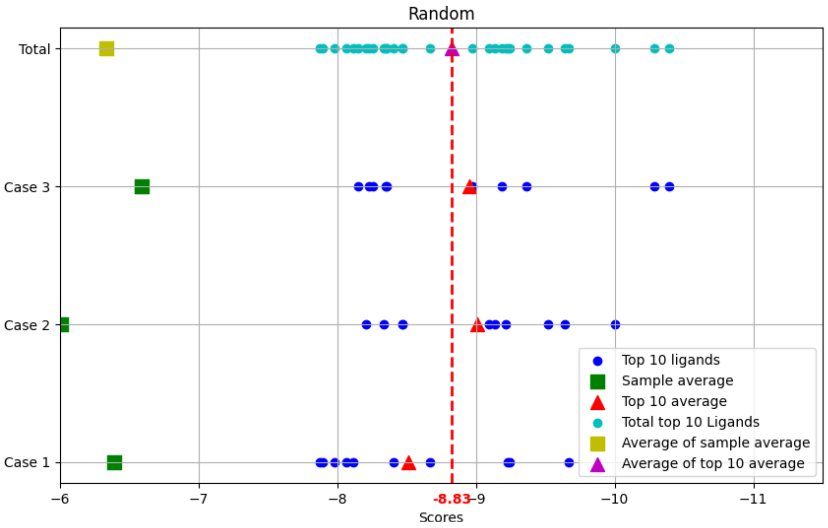
우선적으로 약 4000개의 리간드를 사용하여 연구 목표에 도달할 수 있는 방법을 테스트했다. Autodock Vina를 이용한 결합 점수 측정을 4000개의 리간드 중 1%인 40번으로 제한하여 그 결과를 바탕으로 비교를 할 예정이다. 타겟 단백질로는 4EK3[23]가 사용되었다. 각 알고리즘에서는 40개의 리간드 중 가장 점수가 높은 10개의 리간드를 추천한다.

	평균 점수
4250개 중 3346개 도킹 결과	-6.72

[표 1] 4000개 리간드의 평균 도킹 점수

3.2.1. 무작위 탐색

다른 방법들과의 비교를 위해 무작위 탐색을 가장 먼저 테스트 해 보았다. 무작위 탐색에서는 무작위로 40개의 리간드를 선별해 도킹 후 상위 10개를 추천한다.



[그림 7] 4000개 리간드의 무작위 탐색 결과

Top10 평균	Case 1	Case 2	Case3	평균
결합 점수	-8.51	-9.01	-8.96	-8.83
분자 지문 기반 유사도	0.48	0.41	0.44	0.44

[표 2] 4000개 리간드의 무작위 탐색 결과

3.2.2. k-means 클러스터링

클러스터링 알고리즘을 사용하여 전체 리간드를 몇 개의 클러스터로 나누고, 그 중 상위 클러스터를 찾아 해당하는 클러스터의 리간드를 추천하는 방식이다.

처음에는 4000개의 리간드를 15개의 클러스터로 나누는 클러스터링을 진행했다. 하지만 그 결과 하나의 클러스터에 절반 이상의 리간드가 모이는 등의 불균형 문제가 발생하였기에, 크기가 작은 클러스터를 병합하여 15개의 균일한 클러스터를 만드는 방향으로 바꾸었다.

먼저 각각의 클러스터에서 리간드를 2개씩 선별해 도킹 후 평균 점수를 클러스터에 배정하여 상위 클러스터를 찾는다. 최상위 점수의 클러스터에서 리간드를 하나 뽑아 도킹 후, 클러스터 점수를 계산하는 것을 10번 반복한다. 그렇게 결합 점수를 확인한 40개의 리간드 중 상위 10개의 리간드를 추천한다.

그러나 클러스터의 불균형 문제를 해결하기 위해 채택한 작은 클러스터를 병합하는 방식은, 같은 클러스터에서 리간드 사이의 유사도를 낮게 만들어 클러스터링의 의미가 퇴색되는 문제가 있다.

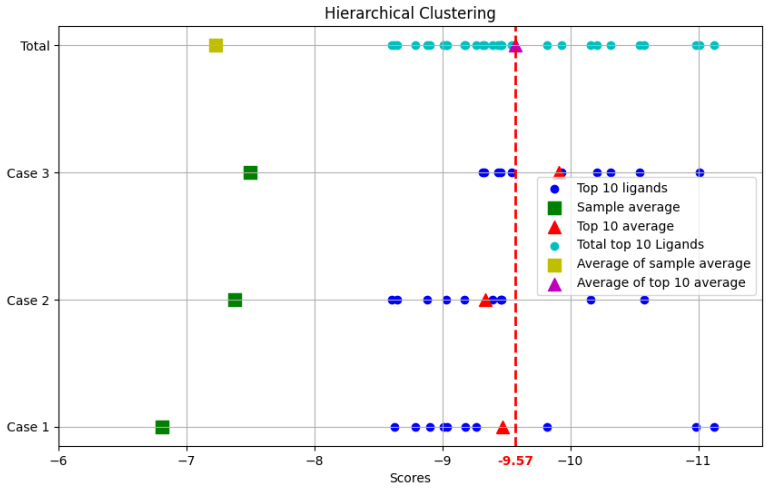
Top10 평균	Case 1	Case 2	Case3	평균
결합 점수	-9.28	-8.63	-8.68	-8.86
분자 지문 기반 유사도	0.52	0.49	0.47	0.493

[표 3] 리간드 4000개의 k-means 클러스터링 결과

3.2.3. 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.

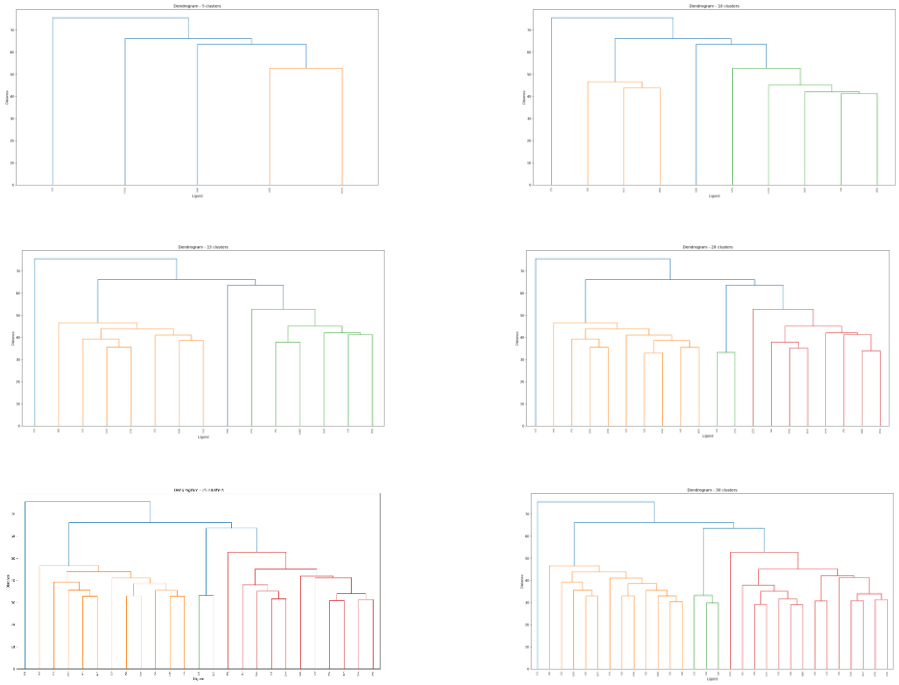
각 클러스터에서 리간드를 하나씩 뽑아 도킹 후, 최고 점수 클러스터에 속하지 못한 리간드를 배제하는 방식을 반복한다. 마지막 최고점 클러스터에서 총합 40번의 도킹이 될 때가지 도킹 후 상위 10개의 리간드를 추천한다.



[그림 8] 리간드 4000개의 계층적 클러스터링 결과

Top10 평균	Case 1	Case 2	Case3	평균
결합 점수	-9.47	-9.34	-9.91	-9.57
분자 지문 기반 유사도	0.53	0.67	0.44	0.550

[표 4] 리간드 4000개의 계층적 클러스터링 결과



[그림 9] 리간드 4000개의 계층적 클러스터링 결과로 생성된 덴드로그램

3.2.4. MEMES 알고리즘

MEMES 알고리즘을 테스트하기 위하여 MEMES 저자의 GitHub에 공유된 MO-MEMES 소스코드를 상황에 맞게 수정하여 사용하였다.[24][25]

계층적 클러스터링으로 리간드를 10개의 클러스터로 나누고, 각 클러스터에서 1개씩 총 10개의 리간드를 초기 리간드로 사용한다. 이후 5개의 새로운 리간드를 선택하는 것을 5번 반복한다. 마지막으로 선택된 35개의 리간드로 GPR 모델을 학습시켜 높은 점수가 예측되는 리간드를 탐색한다. 이전에 선택된 적 없는 리간드를 5개 뽑을 때까지 상위 리간드를 탐색하여 총 40개의 리간드를 탐색할 수 있도록 하였다.

모델을 학습할 때 사용할 리간드 표현 방법으로는 fingerprint와 mol2vec을 사용하였다. fingerprint는 분자를 비트 벡터로 표현한 것이다. 각 원자에 식별자를 할당하고, 이웃 원자에 따라 식별자를 업데이트한다. 지정된만큼 해당 과정을 반복해 비트 벡터를 얻는다. 비트는 분자에서 특정 구조의 존재 여부를 나타낸다.[11] 우리는 rdkit 라이브러리를 이용해 각 리간드를 1024비트의 fingerprint로 변환하여 사용하였다. mol2vec은 word2vec을 분자에 적용한 방법이다. mol2vec 라이브러리를 이용해 분자를 변환하였다. 먼저 분자를 각 단어가 morgan 식별자인 sentence로 나타낸다. 그리고 훈련된 mol2vec 모델을 이용해 벡터로 변환한다.[11]

Acquisition function으로는 EI, PI, UCB를 적용하여 비교해 보았다.

Top10 결합 점수		Case 1	Case 2	Case3	평균
finger print	EI	-10.14	-9.53	-8.98	-9.55
	PI	-9.16	-9.25	-9.07	-9.16
	UCB	-8.77	-8.80	-8.96	-8.84
mol2vec	EI	-9.70	-9.47	-8.97	-9.38
	PI	-8.67	-9.99	-9.17	-9.28
	UCB	-9.02	-8.57	-8.79	-8.79

[표 5] 4000개 리간드의 MEMES 결합 점수

분자 지문 기반 유사도		Case 1	Case 2	Case3	평균
finger print	EI	0.51	0.48	0.63	0.54
	PI	0.49	0.40	0.56	0.48
	UCB	0.49	0.34	0.47	0.43
mol2vec	EI	0.59	0.47	0.39	0.48
	PI	0.53	0.40	0.46	0.46
	UCB	0.44	0.49	0.68	0.54

[표 6] 4000개 리간드의 MEMES 유사도

3.3. 210만개의 리간드 사용

교수님께서 과제를 처음 설명해주실 때 교수님께선 라이브러리 크기를 수십억개까지 생각하셨고, 중간 보고 이후 라이브러리 크기를 더 키우자고 말씀하셨다. 더 큰 리간드 라이브러리는 구축부터 운용까지 막대한 시간이 소요된다. 십억의 1%인 천만개가 1초씩만 소요된다 하여도 약 116일이 소요된다. 도저히 우리의 힘만으론 진행이 어렵다 생각하였고, 중간 보고 때 교수님과 조교님께 여쭤본 결과 조교님이 사용하는 라이브러리를 받을 수 있었다. 해당 파일은 2104318개의 리간드 스마일과 해당 스마일이 4UNN[26] 단백질에 결합한 결과로 이루어져 있었다. 해당 라이브러리를 사용하여 리간드 중 1%인 21000번으로 제한하여 그 결과를 바탕으로 비교를 할 예정이다. 각 알고리즘에서는 21000개의 리간드 중 가장 점수가 높은 10개의 리간드를 추천한다.

	평균 점수
2104318개 리간드	-7.86

[표 7] 210만개 리간드의 평균 점수

3.3.1. 무작위 탐색

다른 알고리즘의 결과와 비교를 위해서, 무작위로 21000개의 리간드를 선별 후 상위 10개의 평균 결과를 얻는다.

Top10 평균	Case 1	Case 2	Case3	Case4	Case5	평균
결합 점수	-11.15	-11.10	-11.24	-11.11	-11.44	-11.208
분자 지문 기반 유사도	0.49	0.47	0.57	0.43	0.39	0.470
선별된 리간드 전체 평균	-7.86	-7.87	-7.86	-7.86	-7.87	-7.866

[표 8] 210만개 리간드의 무작위 탐색 결과

3.3.2. k-means 클러스터링

기존의 방식처럼 리간드들을 군집화하여 클러스터 별로 무작위로 리간드를 뽑아 클러스터에 점수를 매긴다. 8000개로 이루어진 클러스터에서 각각 2개씩 뽑았고, 이후 가장 높은 점수의 클러스터에서 2개의 리간드를 선별해 피드백하는 방식을 반복하였다. 그러나 라이브러리의 크기가 증가함에 따라 메모리 부족에 의해 기존의 방식으로 클러스터를 만드는 것이 불가능하게 되었다.

클러스터 인덱스 파일을 만드는 것은 MiniBatchKMeans, Dask 등을 통해 구현이 가능하였지만, 계층적 클러스터링은 실패했다. 기존에 사용하였던 scipy.cluster.hierarchy 방식이 특히 메모리를 많이 사용한다고 하여 근사적 계층적 클러스터링 방법인 BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies)와 nn_chain(PyClusterKit)을 사용해 보았고, 데이터 형식 변환, 차원 축소 등의 방식을 적용해 보았지만 모두 작동하지 않아 결국 200만 라이브러리에서 계층적 클러스터링 구현은 포기하였다.

Top10 평균	Case 1	Case 2	Case3	Case4	Case5	평균
결합 점수	-11.63	-11.49	-11.66	-11.50	-11.49	-11.554
분자 지문 기반 유사도	0.48	0.45	0.49	0.48	0.52	0.484
선별된 리간드 전체 평균	-8.21	-8.21	-8.22	-8.22	-8.22	-8.216

[표 9] 210만개 리간드 k-means 클러스터링 결과

3.3.3. MEMES 알고리즘

우선 k-means 클러스터링을 통해 전체 리간드를 400개의 클러스터로 나누었다. 초기 리간드는 8000개로, 400개의 클러스터에서 각각 20개의 리간드를 선택하였다. 그리고 6번의 반복에서 새로운 리간드를 2000개씩 선택하여 모델을 학습시킨다.

기존 4000개의 리간드를 사용할 때는 학습에 사용되는 리간드가 몇십개 정도이므로 한 번에 사용할 수 있었다. 하지만 몇천개의 리간드는 메모리 부족으로 한 번에 학습시킬 수 없는 문제가 발생하였다. 그래서 배치 사이즈를 1000으로 하여 미니 배치 방식을 이용해 모델을 학습시키게 되었다.

6번의 반복 이후 선택된 20000개의 리간드로 GPR 모델을 학습시키고, 예측된 점수가

높은 1000개의 리간드를 탐색한다. 그렇게 해서 총 2100개의 리간드가 탐색되고, 그 중 가장 점수가 높은 10개의 리간드를 추천한다.

acquisition function으로는 EI, PI, UCB에 더해 greedy와 EBAF(Entropy-Based Acquisition Function), HAF(Hybrid Acquisition Function)를 사용하였다. greedy는 단순하게 이미 찾은 높은 값 근처에서 다음 입력 값을 선택한다. EBAF는 엔트로피가 높은 리간드를 다음 입력으로 선택한다. 엔트로피는 평균 정보량으로, 일어날 확률이 낮은 일일수록 높다. 가우시안 분포에서 엔트로피는 $E[-\log P(x)] = E[-\log \mathcal{N}(\mu, \sigma^2)] = \frac{1}{2}\log 2\pi\sigma^2 + \frac{1}{2}$로 계산된다.[27] HAF는 EI와 UCB를 합한 것이다. EI와 UCB에 가중치를 곱해 더한다. 여기서는 가중치로 0.5, 0.5를 사용해 $HAF = 0.5 * EI + 0.5 * UCB$로 계산하였다.[5][7][8][9][10]

Top10 결합 점수		Case 1	Case 2	Case 3	Case 4	Case 5	평균
finger print	EI	-12.13	-12.12	-12.08	-12.09	-12.10	-12.10
	PI	-12.11	-12.09	-12.09	-12.08	-12.12	-12.10
	UCB	-12.15	-12.15	-12.15	-12.07	-12.04	-12.11
	Greedy	-12.07	-12.15	-12.10	-12.13	-12.01	-12.09
	HAF	-12.01	-12.15	-12.15	-12.02	-12.13	-12.09
	EBAF	-11.66	-11.41	-11.50	-11.61	-11.35	-11.51
mol2 vec	EI	-12.19	-12.19	-12.19	-12.18	-12.19	-12.19
	PI	-12.19	-12.19	-12.19	-12.19	-12.12	-12.18
	UCB	-12.19	-12.19	-12.19	-12.19	-12.19	-12.19
	Greedy	-12.19	-12.19	-12.19	-12.19	-12.12	-12.18
	HAF	-12.19	-12.19	-12.19	-12.19	-12.19	-12.19
	EBAF	-11.16	-11.08	-11.18	-10.95	-11.28	-11.13

[표 10] 210만개 리간드 MEMES 결과

Top10 유사도		Case 1	Case 2	Case 3	Case 4	Case 5	평균
finger print	EI	0.49	0.43	0.47	0.44	0.44	0.454
	PI	0.45	0.44	0.43	0.41	0.45	0.436
	UCB	0.46	0.44	0.42	0.44	0.43	0.438
	Greedy	0.42	0.44	0.43	0.43	0.41	0.426
	HAF	0.44	0.44	0.44	0.43	0.45	0.440
	EBAF	0.45	0.44	0.41	0.42	0.39	0.422
mol2 vec	EI	0.44	0.43	0.42	0.44	0.42	0.430
	PI	0.42	0.42	0.44	0.44	0.42	0.428
	UCB	0.42	0.42	0.42	0.42	0.42	0.420
	Greedy	0.42	0.42	0.42	0.42	0.41	0.418
	HAF	0.42	0.42	0.42	0.42	0.42	0.420
	EBAF	0.40	0.37	0.43	0.43	0.45	0.416

[표 11] 210만개 리간드의 MEMES 유사도

3.3.4. 샘플 표본 축소

기존 전체 모집단의 1%인 21000개 선별 결과 mol2vec 방식의 EBAF을 제외한 나머지 acquisition function의 경우 92%의 높은 확률로 전체 리간드 중 스코어가 가장 높은 리간드를 찾는데 성공했다. Acquisition function 중 가장 높은 점수를 리턴하는 것을 찾기 위해 샘플 크기를 모집단의 0.07%인 1500개로 줄여서 확인해 보았다.

Top10 결합 점수	Case 1	Case 2	Case3	평균
EI	-11.92	-11.94	-11.99	-11.950
PI	-12.06	-12.03	-12.06	-12.050
UCB	-12.14	-12.18	-12.15	-12.157
Greedy	-11.95	-12.02	-12.02	-11.997
HAF	-12.06	-12.09	-12.08	-12.077

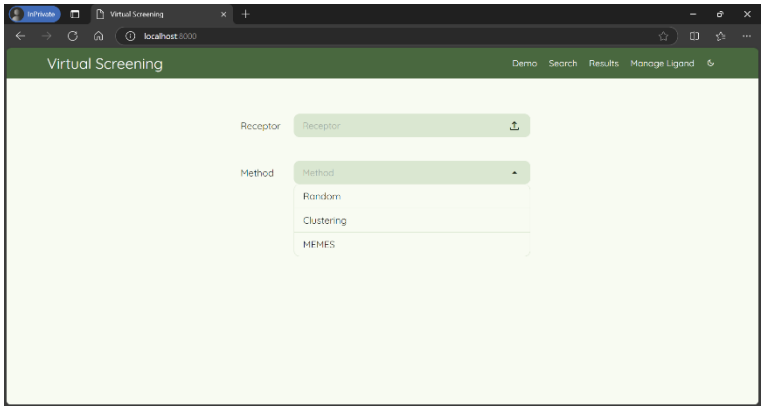
[표 12] 210만개 리간드 중1500개를 탐색했을 때 MEMES 결과

3.3.5. 웹 애플리케이션 탑재

화합물 라이브러리 탐색의 접근성을 높이고 사용자 친화적인 환경을 제공하기 위해, Docker와 Django를 활용하여 편리하게 설치하고 이용할 수 있는 웹 애플리케이션을 개발하였다.

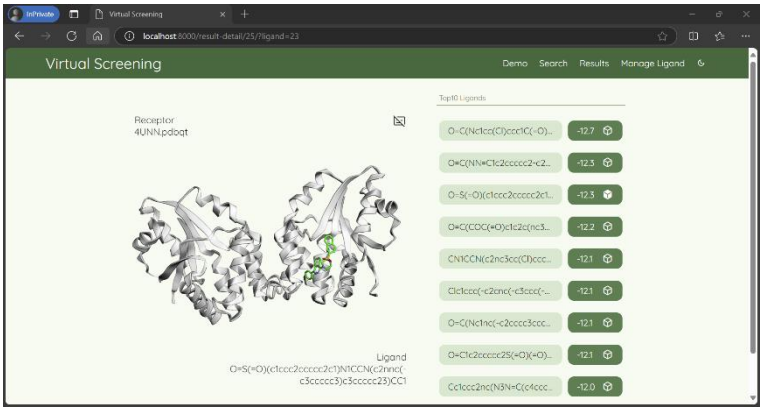
웹 애플리케이션에서 제공하는 주요 기능은 다음과 같다:

- Manage Ligand 페이지: 사용자는 이 페이지를 통해 팀이 수집한 리간드 데이터를 데이터베이스에 쉽게 추가할 수 있다. 또한, 직접 추가하고자 하는 리간드의 SMILES 문자열을 입력하여 데이터베이스에 저장할 수 있다.
- Search 페이지: 사용자는 자신의 컴퓨터에서 PDBQT 파일 형식의 리셉터를 선택하고, Random, Clustering, MEMES 세 가지 메소드 중 하나를 선택하여 탐색을 시작할 수 있다. MEMES 메소드를 선택할 경우 사용할 Acquisition Function도 지정할 수 있다.



[그림 10] 웹 애플리케이션 Search 페이지

- Demo 페이지: 미리 준비된 리셉터와 리간드들의 도킹 스코어를 활용하여, 실제 도킹 과정을 생략하고 각 메소드의 동작을 빠르게 비교할 수 있다. 이를 통해 사용자는 알고리즘의 성능과 특성을 직관적으로 파악할 수 있다.
- Results 페이지: 이전에 수행한 탐색 결과들의 목록을 확인하고, 각 결과의 세부 정보를 열람할 수 있다. 결과의 세부 정보 페이지에서는 추천된 리간드의 도킹 점수를 볼 수 있다. 그리고 Autodock Vina를 통해 얻은 도킹 포즈를 3Dmol.js[28]로 시각화하여 3D 모델을 볼 수 있게 하였다.



[그림 11] 웹 애플리케이션 결과 상세 페이지

웹 애플리케이션은 복잡한 설치 과정 없이도 사용자들이 쉽게 화합물 라이브러리 탐색을 수행할 수 있도록 설계되어 신약 후보물질 발굴 과정에서의 효율성 향상이 기대된다.

4. 연구 결과 분석 및 평가

해당 연구의 목표에 따르면 추천되는 리간드는 결합 점수가 높으면서 유사도가 낮아야 한다. 두 지표를 모두 포함하여 비교하기 위해 결합 점수를 0~1 사이로 정규화하고, 가중치를 곱해 더해서 종합 점수를 계산하였다. 유사도보다 결합 점수가 더 중요하다고 판단하여 결합 점수의 가중치를 0.6, 유사도의 가중치를 0.4로 하였다.

Top10 평균		결합 점수	유사도	종합 점수
무작위 탐색		-8.83 (8)	0.44 (2)	0.34 (7)
k-means 클러스터링		-8.86 (6)	0.49 (6)	0.33 (8)
계층적 클러스터링		-9.57 (1)	0.55 (9)	0.57 (1)
finger print	**EI**	-9.55 (2)	0.54 (8)	0.57 (2)
	PI	-9.16 (5)	0.48 (4)	0.45 (5)
	UCB	-8.84 (7)	0.43 (1)	0.35 (6)
mol2vec	**EI**	-9.38 (3)	0.48 (4)	0.53 (3)
	PI	-9.28 (4)	0.46 (3)	0.50 (4)
	UCB	-8.79 (9)	0.54 (7)	0.29 (9)

[표 13] 4000개 리간드의 알고리즘 별 Top10 평균 점수

4000개의 리간드를 사용했을 때의 점수를 확인해보면, 유사도에 큰 차이가 없었기 때문인지 종합 점수의 순위는 결합 점수의 순위와 크게 다르지 않았다. 가장 좋은 결과를 보인 것은 계층적 클러스터링을 사용했을 때이다. 그리고 fingerprint와 EI를 사용한 MEMES의 결과가 거의 비슷하게 좋았다. 의외로 무작위 탐색보다 k-means클러스터링 결과와 MEMES에서 mol2vec과 UCB를 사용한 결과가 더 나빴다.

MEMES에서 분자 표현 방법을 비교해보면, acquisition function으로 PI를 사용했을 때를 제외하고는 mol2vec보다 fingerprint로 표현했을 때 더 좋은 결과를 보였으나 큰 차이는 아니었다. acquisition function을 비교해보면 EI를 사용했을 때가 가장 점수가 높고, UCB를 사용했을 때가 가장 낮았다. mol2vec으로 표현했을 때, 특히 UCB의 점수가 다른 두 acquitision function보다 큰 차이로 낮았다.

Top10 평균		결합 점수	유사도	종합 점수
무작위 탐색		-11.208 (13)	0.470 (13)	0.337 (13)
k-means 클러스터링		-11.554 (11)	0.484 (14)	0.499 (12)
finger print	EI	-12.104 (7)	0.454 (12)	0.777 (9)
	PI	-12.098 (8)	0.436 (9)	0.781 (8)
	UCB	-12.112 (6)	0.438 (10)	0.787 (6)
	Greedy	-12.092 (9)	0.426 (6)	0.782 (7)
	HAF	-12.092 (9)	0.440 (11)	0.777 (10)
	EBAF	-11.506 (12)	0.422 (5)	0.500 (11)
mol2vec	EI	-12.188 (3)	0.430 (8)	0.827 (3)
	PI	-12.176 (4)	0.428 (7)	0.822 (5)
	UCB	-12.190 (1)	0.420 (3)	0.832 (1)
	Greedy	-12.176 (4)	0.418 (2)	0.826 (4)
	HAF	-12.190 (1)	0.420 (3)	0.832 (1)
	EBAF	-11.130 (14)	0.416 (1)	0.321 (14)

[표 14] 210만개 리간드의 알고리즘별 Top10 평균 점수

210만개의 리간드를 사용한 경우도 4000개를 사용한 경우와 같이 결합 점수 순위와 종합 점수 순위가 비슷하다. MEMES에서 mol2vec으로 분자를 표현하고, acquisition function으로 UCB와 HAF를 사용했을 때 결과가 가장 좋았다. 가장 나쁜 결과를 보인 것은 MEMES에서 mol2vec과 EBAF를 사용했을 때이고, 그 다음으로는 무작위 탐색의 결과가 안좋았다. k-means 클러스터링은 유사도가 아닌 유클리드 거리로 클러스터링을 했음에도 무작위 탐색보다 좋은 결과를 보였다.

이전과 다르게 MEMES에서 fingerprint를 사용했을 때와 mol2vec을 사용했을 때의 차이가 명확하게 난다. EBAF를 제외하면 결합 점수 약 0.8, 종합 점수 약 0.4정도 차이로 mol2vec을 사용했을 때의 결과가 더 좋다. acquisition function으로 EBAF를 사용한 결과는 무작위 탐색과 비슷할 정도로 좋지 않다. Greedy 함수를 포함한 다른 함수의 결과는 차이가 거의 없으며, 근소한 차이로 UCB가 가장 점수가 높다. 이것은 4000개 리간드를 사용했을 때 UCB를 사용한 점수가 낮았던 것과는 반대되는 결과이다.

210만개의 리간드 중 1500개의 리간드만 탐색한 경우에서도 mol2vec으로 분자를 나타내고 UCB를 사용한 MEMES의 결과가 가장 좋았다. UCB > HAF ≥ PI > Greedy ≥ EI 순으로 결합 점수가 높았다. 다만 0.07%밖에 탐색하지 않았는데도 불구하고 지나치게 좋은 결과가 나왔다. 여러가지 가능성을 생각해 보았으나, 원인을 파악하지는 못했다. 미리 계산된 점수를 사용한 것 때문에 높은 점수가 나왔을 가능성이 있다.

5. 결론 및 향후 연구 방향

이번 과제에서 신약 개발 후보물질 발굴을 위해 거대 화합물 라이브러리 탐색을 최적화하는 방법을 탐구하였다. 클러스터링 방식을 계층적으로 적용해 보았고, 베이지안 최적화 기법의 내부 목적함수에 변화를 주는 등의 방식으로 결합 예측도가 높을 것으로 추정되는 리간드를 선별해 보았다. 이외에도 과제에 도움이 될 만한 다양한 모델을 적용해 보았다. 결과적으로 전체 리간드의 1%를 탐색한 결과, 75%의 확률로 가장 결합도가 높은 리간드를 성공적으로 찾아내는 성과를 얻을 수 있었고 가장 좋은 성적을 내는 방식의 경우 샘플의 크기를 모집단의 0.07%까지 낮춰도 90%의 확률로 가장 결합도가 높은 리간드를 성공적으로 찾아낼 수 있었다.

하나 특이한 점은 200만개 라이브러리를 사용하였을 때, 결합 예측도와 유사도가 원하는 방향으로 같이 갔다는 것이다. 교수님께선 높은 결합 예측도와 낮은 유사도를 얻는 방향으로 가면 좋겠다고 말씀해주셨고, 4천개 라이브러리에선 예측도가 올라갈수록 유사도도 올라가는 트레이드 오프 관계였다. 의도치 않게 유사도 문제가 해결되어 아쉬움이 남는다. 이는 화합물 다양성을 확보하면서도 최적의 결합성을 갖춘 후보물질을 발굴하는 데 중요한 요소이므로, 향후 연구에서는 이러한 부분을 보완할 필요가 있다.

향후 연구 방향으로는 현재 급속히 발전하고 있는 신기술을 적극적으로 도입하고 대응해야 할 필요가 있다. 예를 들어, Google DeepMind의 AlphaProteo는 단백질 구조 예측을 최적화하는 AI 모델이며, NVIDIA의 BioNemo는 AI 기반의 신약 개발 플랫폼으로, 이러한 빅테크 기업들의 최신 인공지능 기반 기술은 신약 후보물질 발굴 과정의 효율성을 극대화할 수 있는 잠재력을 가지고 있다. 이들 외에도 다양한 기술들이 빠른 주기로 공개되고 있으며, 이는 신약 개발 분야의 패러다임을 바꿀 것이다. 따라서 향후 연구에서는 본 과제와 같이 최신 기술을 바탕으로 탐색 방법을 지속적으로 개선하는 접근이 필요하다. 신기술과의 결합이 단순히 새로운 도구를 사용하는 것에 그치지 않고, 각 기술의 강점을 최대한 활용하여 보다 다양하고 고도화된 신약 후보물질을 발굴하는 전략이 중요할 것이다. 이러한 방향은 향후 신약 개발의 성공 가능성을 높이는 데 큰 기여를 할 것으로 기대된다.

6. 참고 문헌

[1] 남궁석, *알파폴드: AI 신약개발 혁신*, biospectator, 2024

[2] Eberhardt, J., Santos-Martins, D., Tillack, A.F., Forli, S., "AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings", *Journal of Chemical Information and Modeling,* Vol. 61, No. 8, pp. 3891-3898, 2021

[3] Trott, O., & Olson, A. J., "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading.", *Journal of computational chemistry*, Vol. 31, No. 2, pp. 455-461, 2010

[4] Center of Computational Structural Biology (CCSB) - Scripps Research(2021), AutoDock Vina [Online] Available: https://autodock-vina.readthedocs.io/en/latest/index.html

[5] Graff, David E. and Shakhnovich, Eugene I. and Coley, Connor W., "Accelerating high-throughput virtual screening through molecular pool-based active learning", *Chem. Sci.*, Vol. 12, No.22, pp. 7866-7881, 2021

[6] Brain_93 (2021), 베이지안 최적화(Bayesian Optimization) [Online]. Available: https://data-scientist-brian-kim.tistory.com/88

[7] Stathis Kamperis (2021), Acquisition functions in Bayesian Optimization [Online]. Available: https://ekamperi.github.io/machine%20learning/2021/06/11/acquisition-functions.html

[8] Meta Platforms (2024), BoTorch [Online]. Available: https://botorch.org/api

[9] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimizatio", *Advances in Neural Information Processing Systems 33*, 2020

[10] Jay Han (2022), Bayesiaan Optimization [Online]. Available: https://otzslayer.github.io/ml/2022/12/03/bayesian-optimization.html

[11] Sarvesh Mehta, Siddhartha Laghuvarapu, Yashaswi Pathak, Aaftaab Sethi, Mallika Alvala, U. Deva Priyakumar, "MEMES: Machine learning framework for Enhanced MolEcular Screening", *Chem. Sci.*, Vol.12, No.35, pp. 11710-11721, 2021

[12] Corso, Gabriele and Stärk, Hannes and Jing, Bowen and Barzilay, Regina and Jaakkola, Tommi, "DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking", International Conference on Learning Representations (ICLR), 2023

[그림 3] https://github.com/gcorso/DiffDock

[13] yunhuijang (2022), Diffusion model 설명 (Diffusion model 이란? Diffusion model 증명) [Online]. Available: https://process-mining.tistory.com/182

[14] Weng, Lilian. (2021). What are diffusion models? [Online]. Available: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

[15] 알파고라니 (2024), [만들면서 배우는 생성 AI] 8 장 - 확산 모델(diffusion model) [Online]. Available: https://velog.io/@running_learning/%EB%A7%8C%EB%93%A4%EB%A9%B4%EC%84%9C-%EB%B0%B0%EC%9A%B0%EB%8A%94-%EC%83%9D%EC%84%B1-AI-8%EC%9E%A5-%ED%99%95%EC%82%B0-%EB%AA%A8%EB%8D%B8diffusion-model

[16] RCSB PDB [Online]. Available: https://www.rcsb.org/

[17] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, "The Protein Data Bank", Nucleic Acids Research 28, pp. 235-242, 2000

[18] BioinformaticsCopilot (2022), Molecular Docking for Beginners | Autodock Full Tutorial [Online]. Available: https://www.youtube.com/watch?v=ZVKKsK5DsCY

[19] Open Babel Team (2023), Open Babel [Online], Available: http://openbabel.org

[20] Noel M. O'Boyle, Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, Geoffrey R. Hutchison, "Open Babel: An open chemical toolbox.", *J. Cheminf.*, Vol.3, No. 33, 2011

[21] RDKit: Open-source cheminformatics. [Online], Available: https://www.rdkit.org

[22] CHML (2023), 파이썬 RDKit 을 이용한 분자 유사도 (Tanimoto Similarity) 계산 [Online]. Available: https://untitledtblog.tistory.com/189

[23] Kang, Y.N., Stuckey, J.A. (2013), Crystal structure of apo CDK2 [Online]. Available: https://doi.org/10.2210/pdb4EK3/pdb

[24] Mehta S, Goel M and Priyakumar UD, "MO-MEMES: A method for accelerating virtual screening using multi-objective Bayesian optimization", *Front. Med*, 9:916481, 2022

[25] Mehta S, Goel M and Priyakumar UD (2022), MO-MEMES [Online]. Available: https://github.com/devalab/MO-MEMES

[26] Naik, M., Raichurkar, A., Bandodkar, B.S., Varun, B.V., Bhat, S., Kalkhambkar, R., Murugan, K., Menon, R., Bhat, J., Paul, B., Iyer, H., Hussein, S., Tucker, J.A., Vogtherr, M., Embrey, K.J., Mcmiken, H., Prasad, S., Gill, A., Ugarkar, B.G., Venkatraman, J., Read, J., Panda, M. (2015), Structure Guided Lead Generation for M. Tuberculosis Thymidylate Kinase (Mtb Tmk): Discovery of 3-Cyanopyridone and 1,6-Naphthyridin-2-One as Potent Inhibitors. [Online]. Available: https://doi.org/10.2210/pdb4UNN/pdb

[27] Gregory Gundersen (2020), Entropy of the Gaussian [Online]. Available: https://gregorygundersen.com/blog/2020/09/01/gaussian-entropy/

[28]Nicholas Rego and David Koes, 3Dmol.js: molecular visualization with WebGL, Bioinformatics, 31, 8, pp.1322-1324, 2015