Primjena strojnog učenja u kemijskom inženjerstvu

Rapinac, Nela

SVEUČILIŠTE U ZAGREBU

FAKULTET KEMIJSKOG INŽENJERSTVA I TEHNOLOGIJE

SVEUČILIŠNI PREDDIPLOMSKI STUDIJ

Nela Rapinac

ZAVRŠNI RAD

Zagreb, rujan 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET KEMIJSKOG INŽENJERSTVA I TEHNOLOGIJE

SVEUČILIŠNI PREDDIPLOMSKI STUDIJ

Nela Rapinac

ZAVRŠNI RAD

Primjena strojnog učenja u kemijskom inženjerstvu

Voditelj rada: doc. dr. sc. Miroslav Jerković

Članovi ispitnog povjerenstva: doc. dr. sc. Miroslav Jerković

doc. dr. sc. Erna Begović Kovač

doc. dr. sc. Željka Ujević Andrijić

Zagreb, rujan 2023.

Ovaj rad izrađen je na Fakultetu kemijskog inženjerstva i tehnologije, Sveučilišta u Zagrebu, Zavod za matematiku, akademske godine 2022./2023. pod mentorstvom doc. dr. sc. Miroslava Jerkovića.
Zahvaljujem mentoru na pomoći i smjernicama te strpljenju i razumijevanju.

Sadržaj

Sažetak

Strojno učenje je jedan od najnovijih i trenutno najpopularnijih dijelova računalne znanosti - primjena algoritama koji stvaraju neki smisao od skupa podataka. Kreće se od vektora, a glavni već razvijeni algoritmi su regresija i klasifikacija. Od složenijih algoritama najpopularnije su neuralne mreže koje, kao što im i ime kaže, podsjećaju na rad ljudskog mozga te se sastoje od neurona i čvorova. Strojno učenje, uključujući i neuralne mreže, ima široku primjenu i postaje svakodnevnica uporabe, a da se to ni ne primjećuje.

Strojno učenje se može primjenjivati i u kemijskom inženjerstvu, a posebno je popularno razvijanje metoda za predviđanje molekularnih svojstava. Postoje razne metode i modeli, a jedna od metoda je i Uni-Mol metoda koja uzima u obzir 3D konfiguraciju molekula. Metoda se bazira na transformeru, vrsti dubokog učenja koje radi slično kao i neuralne mreže uz određena poboljšanja.

U ovome radu bit će objašnjeni spomenuti pojmovi i ključne riječi te algoritmi i metode, a cilj rada je uvidjeti veliki potencijal koje strojno učenje ima u konkretno kemijskom inženjerstvu, ali i općenito u ostalim industrijama.

Ključne riječi: strojno učenje, vektor, neuralne mreže, transformer, predviđanje svojstva molekula, Uni-Mol metoda, Python

Abstract

Machine learning is one of the newest and currently most popular parts of computer science - the application of algorithms that make some sense out of a set of data. It starts from vectors, and the main already developed algorithms are regression and classification. Among the more complex algorithms, the most popular are neural networks, which, as their name suggests, resemble the work of the human brain and consist of neurons and nodes. Machine learning, including neural networks, have a wide application and becoming common place without anyone realizing it.

Machine learning can also be applied in chemical engineering, and the development of methods for predicting molecular properties is particularly popular. There are various methods and models, and one of the methods is the Uni-Mol method, which takes into account the 3D configuration of molecules. The method is based on a transformer, a type of deep learning that works similarly to neural networks with some improvements.

In this paper, the mentioned terms and keywords as well as algorithms and methods will be explained, and the aim of the paper is to comprehend the great potential that machine learning has in chemical engineering in particular, but also in other industries in general.

Keywords: machine learning, vector, neural networks, transformer, molecular property prediction, Uni-Mol method, Python

Strojno učenje

Dio računalne znanosti

Strojno učenje je jedan od najnovijih i trenutno najpopularnijih dijelova računalne znanosti – primjena algoritama koji stvaraju neki smisao od skupa podataka. Baziraju se na velikom skupu primjera podataka koji mogu doći iz prirode, biti ručno izrađeni od strane ljudi ili su pak dobiveni nekim drugim algoritmima. Strojno učenje je proces skupljanja tih podataka i stvaranja nekog modela ili algoritama koji ih može sve povezati. Puno je efektivnije od ručne ljudske obrade podataka i, naravno, puno brže te nakon nekog vremena može samo predviđati podatke i donositi odluke. Strojno učenje je postala zapravo svakodnevnica u upotrebi tehnologije te nalazi primjenu u raznim područjima, a samo neki od primjera su softveri za prepoznavanje teksta i glasa, pretraživanja na internetu, filteri za neželjenu elektroničku poštu, programi za igranje šaha, itd.

Vrste strojnog učenja

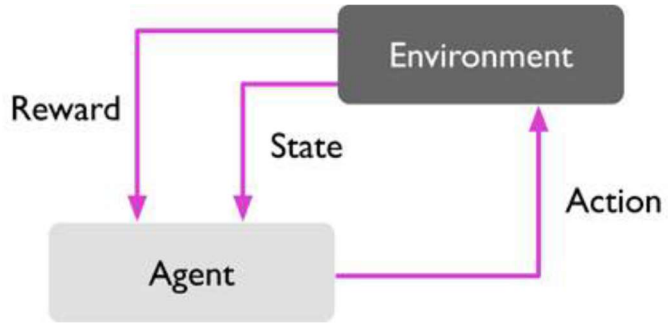
Strojno učenje dijeli se na nadzirano učenje, učenje bez nadzora, polunadzirano učenje i ojačano učenje.

Nadzirano učenje radi sa skupom označenih podataka, a svaki podatak se sastoji od određenog broja označenih vektora. Cilj ovakve vrste učenja je da na temelju dobivenih podataka napravi model po kojem može davati oznake kao izlazne informacije. Za predodžbu se uzima primjer modela koji na temelju ulaznih podataka pacijenta u bolnici (npr. visine, težine, spola i pripadajućih simptoma) daje vjerojatnost da osoba ima rak.

Kod učenja bez nadzora radimo sa skupom neoznačenih podataka i vektora, a cilj je napraviti model pretvorbe ulaznih vektora u neke druge vektore ili vrijednosti. Ovakvo učenje se koristi za određene praktične probleme.

Polunadzirano učenje se sastoji i od označenih i od neoznačenih podataka, a prevladavaju neoznačeni podatci. Cilj je identični kao i kod nadziranog učenja, ali model bi trebao biti bolji zbog neoznačenih podataka koji poboljšavaju učenje (puno je veći broj primjera).

Ojačano učenje je vrsta učenja u kojem stroj „živi“ u okolini i uočava stanje okoline kao označene vektore. Stroj obavlja određene radnje i za to dobiva nagrade. Cilj je da nauči pravila, tj. funkciju *f*, koja na temelju ulaznih podataka vraća optimalnu radnju za to stanje. Radnja je optimalna ako je nagrada koja se očekuje maksimalna.



Slika 1: Način rada ojačanog učenja [1]

Matematička podloga

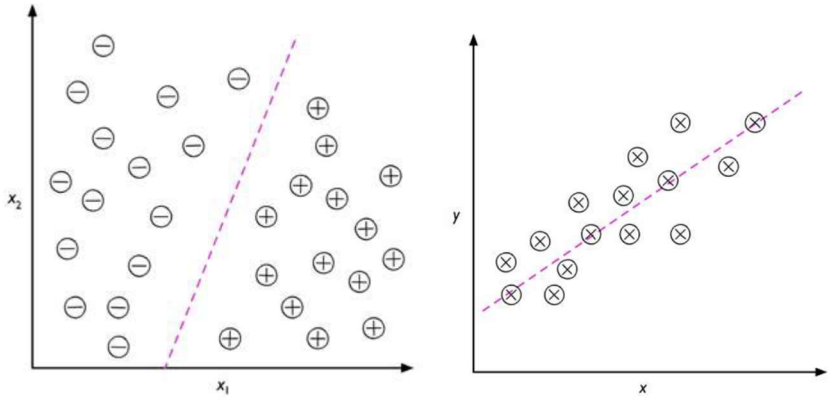
Strojno učenje temelji se na vektorima. Vektor je usmjerena dužina s određenom točkom početka i kraja. Označavaju se malim tiskanim slovima, isto kao i pravci. Prikazuju se kao dužine sa strelicom na jednoj strani koja označava smjer kretanja. Mogu biti u jednoj dimenziji ili više njih. Svaki vektor ima svoju duljinu (modul), smjer i orijentaciju, za razliku od skalara koji je određen samo svojom vrijednošću. Vektori se mogu međusobno zbrajati i oduzimati, a mogu se vršiti i matematičke operacije sa skalarima i matricama. Jedino se onda mora paziti da se dimenzije vektora i matrica slažu jer nije ih uvijek moguće kombinirati. Vektori služe za prikazivanje preslikavanja, translacija i rotacija u ravnini te se najviše koriste u računalnoj grafici.

Klasifikacija i regresija

Razlikuju se dva načina obrade podataka – klasifikacija i regresija.

Klasifikacija je podvrsta nadziranog učenja s ciljem predviđanja oznaka neoznačenih novih ulaznih podataka na temelju prethodnog učenja. Najlakši primjer za predočiti je algoritam za predviđanje neželjene elektroničke pošte; binarni sustav u kojem je elektronička pošta ili željena ili neželjena (eng. spam). Algoritam uči pravila po kojima ih razdvaja – isprekidana linija na Slici 2 (lijevo).

Cilj je pronaći pravilo po kojima su svi primjeri što udaljeniji od linije razdvajanja. Naravno, ovaj binarni problem je jedan od najlakših primjera, postoje i višeklasne klasifikacije.



Slika 2: Usporedba klasifikacije (lijevo) i regresije (desno) [1]

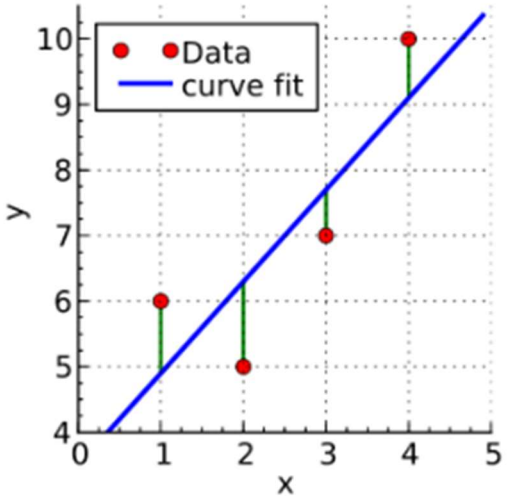
Regresija je predviđanje idućeg izlaznog podatka na temelju primjera parova zavisnih i nezavisnih varijabli. Pokušava se doći do pravila, tj. funkcije koja najbolje opisuje vezu između dva skupa podataka.

Cilj je da su svi podaci što bliže pravilu, tj. isprekidanoj liniji na Slici 2 (desno).

Najčešći algoritmi

Postoje određeni, već gotovi osnovni algoritmi kojima se riješavaju većina problema, ili se barem kreće od njih.

Jedan od njih je i već spomenuta regresija. Najpoznatija od svih regresijskih metoda je metoda linearne regresije. To je vrlo popularan algoritam kojime se traži linearna veza između ulaznih i izlaznih podataka, kao što je prikazano na Slici 3.



Slika 3: Linearna regresija [2]

Cilj je, naravno, da su svi podaci što bliže liniji linearne regresije kako bi i idući ulazni podaci davali točne izlazne podatke.

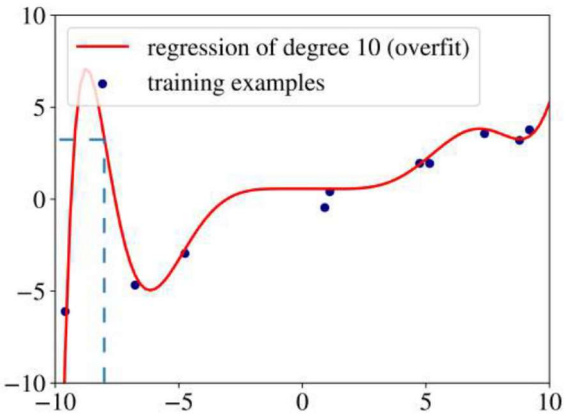
Ovaj model naziva se model jednostavne linearne regresije jer postoji jedna ulazna varijabla x i jedna izlazna varijabla y čije se predviđanje ovim modelom dobiva putem računanja formule $f_{w,b}(x) = wx + b$, gdje su w, x i b realne veličine, a w i b nazivaju se parametri modela. Parametar w predstavlja koeficijent smjera, tj. nagib pravca sa Slike 3, a parametar b predstavlja odsječak na y-osi.

Pored jednostavne linerane regresije, može se promatrati tzv. višestruka linearna regresija, u kojoj postoji više od jedne ulazne, tj. nezavisne varijable. Može se izraziti istom jednadžbom $f_{w,b}(x) = wx + b$, ali u kojoj w predstavlja vektor određene dimenzije;

$$w = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_n \end{bmatrix} \quad x = [x_1 \quad x_2 \quad . \quad . \quad . \quad x_n]$$

dok je b realan broj kao i kod linearne regresije. Množenjem w i x dobi se realan broj koji se onda može normalno zbrojiti s realnim brojem b.

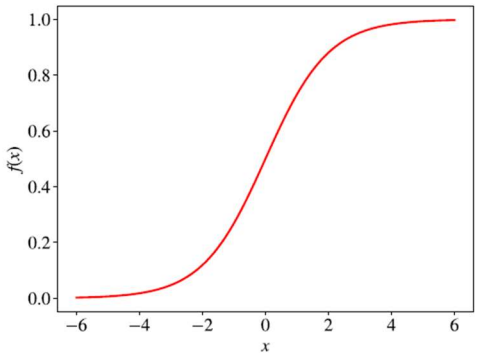
Linearna regresija se često koristi baš zbog svoje jednostavnosti. Može se i pokušati s višim polinomima, ali onda često dolazi do grešaka. Kod viših polinoma, podaci koji se koriste za pronalazak regresije vrlo dobro se poklapaju s funkcijom, tj. ako se pretjera sa stupnjem polinoma može se doći do tzv. problema *overfittinga*. Međutim, onda novi podaci mogu dosta odskakati od vrijednosti koje predviđa model (Slika 10). Stoga se ne preporuča korištenje polinoma visokog stupnja.



Slika 4: Polinom 10.-og stupnja koji ne daje dobre rezultate za nove ulazne podatke [3]

Drugi algoritam koji se koristi je logistička regresija – matematički je samo slična linearnoj regresiji pa se zato tako zove, ali zapravo nije regresija već klasifikacija. Najlakše se objasni preko modela u kojem *x* koordinata ide kao u prijašnjem primjeru (od minus beskonačno do plus beskonačno), a *y* koordinata ide od 0 do 1. Primjer logističke funkcije je sigmoidna funkcija;

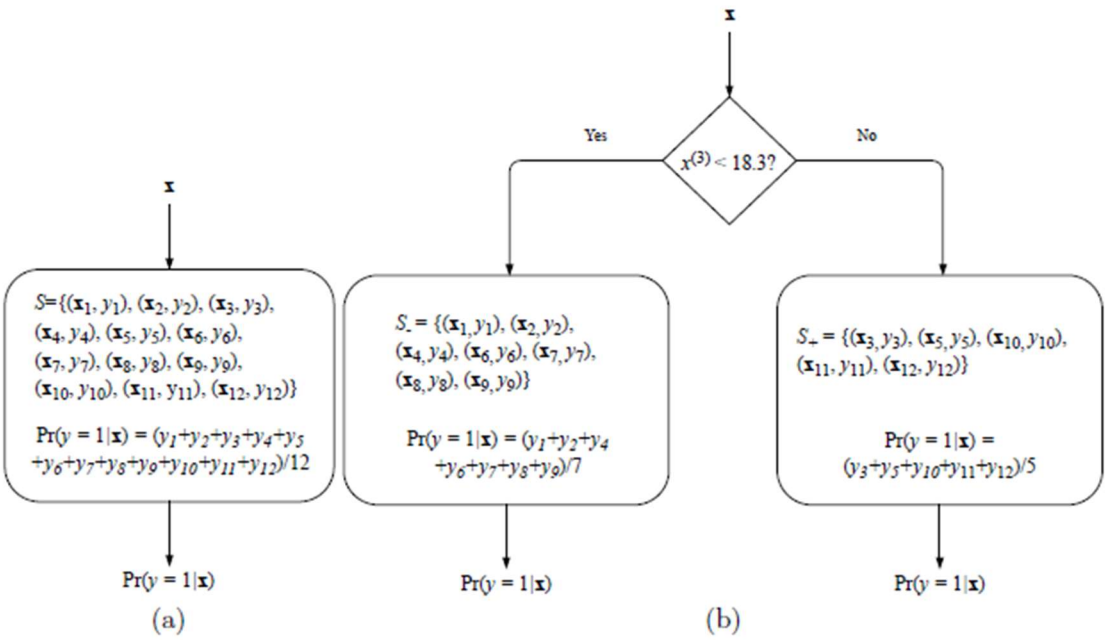
$$f(x) = \frac{1}{1+ e^{-x}}$$



Slika 5: Logistička funkcija [3]

Jednadžba se može preurediti da izgleda kao $f_{w,b}(x) = \frac{1}{1+ e^{-(wx+b)}}$ gdje se onda vidi poveznica s lineranom regresijom – ubačen dio wx+b.

Još jedan algoritam rješavanja problema je drvo odluke. To je grafički prikaz pomoću kojeg se donose odluke te na svakom koraku odluke se onda može prebaciti na lijevu ili desnu granu i nastaviti dalje.



Slika 6: Prikaz drva odluke [3]

Na Slici 6 prikazan je primjer drva odluke. Zadan je neki skup S koji sadrži 12 parova. Lijevo, u (a) dijelu, bez obzira koji ulazni podatak dođe, pretpostavka je uvijek ista, a time i izlazni podatak. U desnom dijelu Slike 6, (b) dio, postoji grananje. Program provjerava je li x dio trećeg para, koji dolazi kao ulazni podatak, strogo manji od 18,3. Ako je, nastavlja se lijevo po jednoj pretpostavci, a ako nije nastavlja se desno po drugoj pretpostavci.

Neuralne mreže

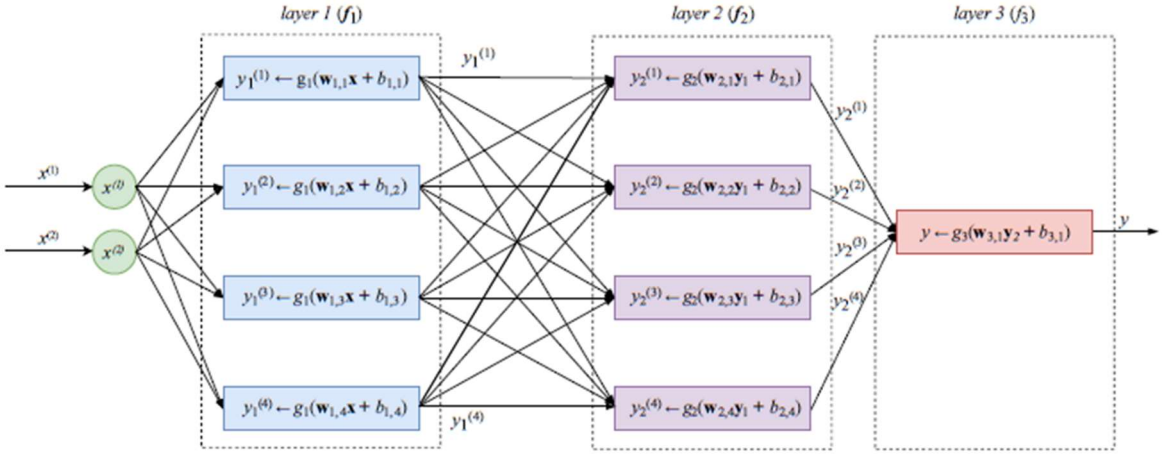
Skup neurona

Neuralne mreže su skup već razrađenih podataka i algoritama, a baziraju se na regresijskoj analizi. Kao što im i ime podsjeća na pojam iz biologije vezan uz ljudski mozak, rade na način koji oponaša radnje i procese u mozgu - skup neurona koji stvara čvorove i sinapse. U biologiji se pretpostavlja da ljudski mozak ima čak 100 milijardi neurona, a svaki od njih je povezan s 1000 do 10 000 drugih neurona oko sebe. Neuralne mreže u informatici uglavnom ipak ne koriste tako velik broj neurona te se pokušavaju složiti jednostavniji modeli koji samo podsjećaju na ljudski mozak.

Neuralne mreže mogu se pojednostaviti na samo tri dijela – ulazni podatak koji ulazi u mrežu i analizira se, procesni sloj koji obrađuje podatke na temelju prijašnjeg učenja i konačni produkt, izlazni podatak koji bi trebao odgovarati željenom. Neuroni se slažu u pravilne strukture, tj. mreže. Mreže se sastoje od slojeva, od kojih neki mogu biti skriveni te se ne vide, a svaka mreža koja ima dva ili više skrivenih slojeva naziva se duboka neuralna mreža. Postoje različite strukture mreža koje se onda podešavaju ovisno o tome koja je potrebna za određeni zadatak. To podešavanje i prilagođavanje zadatku koji se želi riješiti jako utječe na proces učenja mreže.

Kompozicije funkcija

Matematički, to su funkcije oblika: $y = f_{NN}(x) = f_3(f_2(f_1(x)))$, takozvana kompozicija funkcija s više slojeva. Kreće se od funkcije 1 čiji izlazni podatak postaje ulazni od funkcije 2 te isto tako izlazni podatak funkcije 2 je ulazni od funkcije 3. Ukupni konačni željeni izlazni podatak je izlazni podatak funkcije 3. Funkcije mogu biti vektorske ili skalarne, o čemu onda ovisi prikaz izlaznih podataka. U sebi sadržavaju argumente sastavljene od matrica ili vektora. Bitno je da nisu sve kompozicije funkcija linearne jer onda će, bez obzira koliko slojeva ima, cijela funkcija biti linearna te se neće moći predvidjeti kompliciranije funkcije.



Slika 7: Primjer neuralne mreže od tri sloja [3]

Na skici je prikazan primjer jedne neuralne mreže sastavljene od tri sloja; ulaznog, skrivenog i izlaznog. Sve funkcije sastavljene su od dva parametara – *w* i *b*, gdje *w* predstavlja matrični dio, a *b* vektorski. Parametar *g* je zapravo aktivacijska funckija određena prije nego što započne proces učenja.

Indeksi kod svakog parametara označavaju kojem sloju i podsloju pripadaju. Svaki sloj može biti sastavljen od različitih brojeva podsloja, a da skupa čine cjelovitu neuralnu mrežu. Na skici se sa strelicama jasno vidi smjer putovanja informacija te čiji izlazni podaci postaju čiji ulazni. Unutar krugova se informacija ne mijenja, već se samo podaci šalju dalje, a unutar pravokutnika se odvijaju zadane matematičke operacije te se stvaraju novi izlazni podaci koji onda putuju dalje. Prije svakog pravokutnika dolazi do čvora u kojem se skupe svi ulazni podaci u vektor kako bi onda se funkcija mogla lakše izvršiti, a obično radi po istom principu kao i linerana regresija. Bitna je aktivacijska funkcija *g* u zadnjem sloju koja onda određuje cjelokupnu funkciju. Najčešće su dva slučaja – aktivacijska funkcija je linearna pa je cijela mreža po regresijskom modelu ili je aktivacijska funkcija logistička pa je mreža po modelu binarne klasifikacije. Zato je bilo bitno prethodno definirati razlike između regresija i klasifikacija.

Postoji više vrsta razvijenih neuralnih mreža, a najpoznatije su mreže u kojima informcaije putuju prema naprijed, takozvane jednosmjerne ili nepovratne neuralne mreže (eng. feed-forward). Koriste se zbog svoje jednostavnosti jer je smjer putovanja podataka samo jedan, nema „skretanja“ od ulaznih do izlaznih podataka. Mogu koristiti i skrivene slojeve koji rade onda u pozadini, a jedan od primjera njihove upotrebe je u tehnologijama za raspoznavanje lica. Još neke razvijene neuralne mreže su one koje konstantno uče i vraćaju podatke u samu mrežu,

zatim postoje mreže koje raspoređuju podatke u određene kategorije te one čiji slojevi rade zasebno i međusobno neovisno.

Prednosti i mane

Neuralne mreže imaju puno prednosti te su zato postale i popularne. Obzirom da se radi o strojnom algoritmu, mogu raditi dulje i produktivnije od ljudi, biti isprogramirane i unaprijed mijenjati algoritme kako bi se poboljšale te se konstantno razvijaju nove mreže koje se mogu primijeniti u sve više područja.

Ipak, postoje i određene mane. Moraju biti fizički negdje pohranjene, a to zahtijeva dodatan rad ljudi te bez obzira što se temelje na jednostavnim algoritmima, ukupno su komplicirane i potrebno je mnogo vremena, kodova i algoritama da bi se razvile nove mreže koje rade za točno određena područja namjene.

Bez obzira na spomenute mane, to nisu neke velike prepreke u odnosu na ono što mogu pružiti pa se neuralne mreže sve više razvijaju i stavljaju u tehnologije za svakodnevnu upotrebu.

Transformer

Što je transformer?

Transformer je vrsta dubokog učenja s naglaskom na mehanizmu pozornosti, a glavna mu je prednost što zahtijeva puno manje vremena treninga od mnogih drugih neuralnih mreža. Najveća mu je upotreba u prevođenju jezika i treniranju baza podataka različitih riječi u svjetskim jezicima.

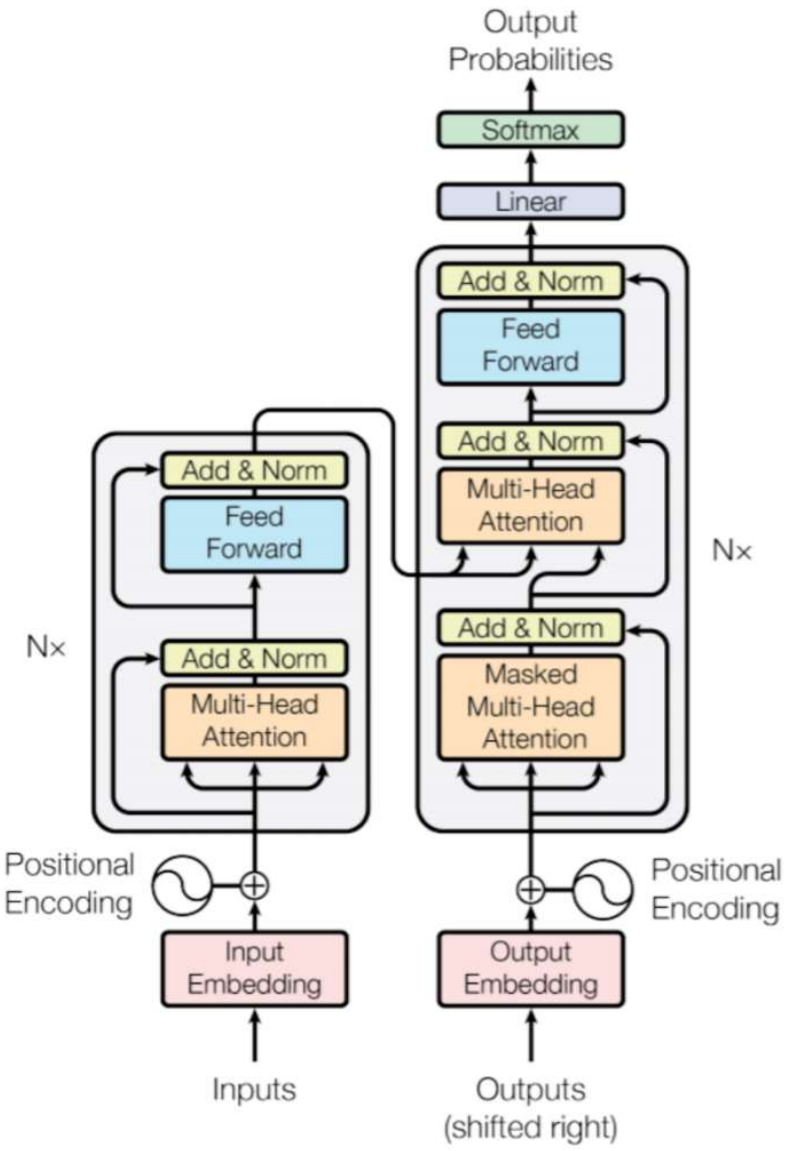
Napredak u odnosu na prethodne arhitekture neuralnih mreža

Već su prije bile razvijene takozvane ponavljajuće neuralne mreže (eng. recurrent neural networks – RNN), poboljšane neuralne mreže u odnosu na već spomenute jednosmjerne neuralne mreže (eng. feed-forward). Glavno je poboljšanje RNN-a bilo to što nisu pamtile samo tijekom treninga, već su učile i pamtile tijekom obrade novih ulaznih i izlaznih podataka. Postoje tri modela po kojem rade; model vektor-sekvenca u kojem je i ulazni i izlazni podatak vektor točno određene veličine, model sekvenca-vektor gdje je ulazni podatak vektor bilo koje veličine, a izlazni podatak vektor točno određene veličine te model sekvenca-za-sekvencom koji je najpopularniji i najčešći, a uzima sekvencu kao ulazni podatak i za izlazni je neka druga sekvenca različite veličine. Ipak, i s tim poboljšanim neuralnim mrežama, postojala su dva glavna problema; RNN su bile jako spore i dugo je trajao proces treninga te nisu mogle pamtiti stare konekcije, tj. memorija im nije bila dovoljno jaka. Stvarao se veliki problem u prevođenju riječi i predviđanju idućih riječi ako su povezane riječi između sebe imale veliki razmak.

Taj se problem riješio s dugom kratkoročnom memorijom (eng. long short-term memory – LSTM), posebnom vrstom RNN-a. Mogu pamtiti puno duže te njihovi neuroni preskaču određene informacije i slojeve koji nisu bitni u tom trenutku kako bi podaci putovali brže. To je moguće uz naglasak na pozornosti u neuralnim mrežama, vrlo slično kao i kod ljudi dok je potrebno uzeti određene podatke unutar velikih grupa informacija – fokus je samo na određenim bitnim informacijama, a ostalo se u tom trenutku zanemaruje. Ovaj mehanizam pozornosti je značajno razvio prevođenje u odnosu na do tad popularni model sekvenca-za-sekvencom jer se prenosi puno više informacija odjednom, a ne samo zadnji dio.

Način rada

U 2017. godini predstavljen je novi model koji se bazira upravo na slojevima pozornosti te je nazvan transformer. [8]Glavna razlika je bila što su se ulazni podaci mogli prenositi paralelno, tj. više njih odjednom pa je vrijeme treninga značajno bilo skraćeno i cijeli je model općenito radio brže od prethodnih, a i dalje je jednako dobro pamtio. Transformer koristi model pozornosti bez RNN-a te računa težinu svakog vektora pozornosti između slojeva.



Slika 8: Skica načina rada transformera [8]

Najlakše je objasniti način rada preko primjera prevođenja riječi i rečenica s jednog na drugi jezik, obzirom da se za to najviše i koristi. Prvo se riječi pretvaraju u vektore te se riječi grupiraju po sličnom značenju i svakoj je riječi dodijeljena određena vrijednost. Potrebno je još riječima dodati vektore koji daju kontekst, ovisno gdje se riječ nalazi u rečenici. Tako spremne riječi pretvorene u vektore postaju ulazni podaci. Nakon toga slijedi zapravo najbitniji dio, vektori pozornosti koji pojašnjavaju koliko je određena riječ bitna u rečenici i odnose s drugim riječima unutar rečenice. Svaki vektor je nezavisan o ostalima te se mogu više njih paralelno procesirati što značajno ubrzava cijeli proces. Tijekom treninga potrebno je u isto vrijeme dati iste rečenice na oba jezika – i onog s kojeg se prevodi i onog na koji se prevodi. Još jedna bitna stvar na koju se mora paziti je da postoji mogućnost maskiranja, tj. skrivanja idućih riječi kako bi program mogao sam predvidjeti iduću riječ pa onda usporediti s utreniranim podacima i na taj način učiti. Nakon usporedbe svih mogućnosti, pregledanih povezanih riječi preko modela pozornosti i općenito obrađenih svih vektora, program vraća najbolju verziju prijevoda rečenice. Ovakav način rada transformera se koristi u najpoznatijim programima za prijevode, uključujući i onaj na Google internet mreži.

Predviđanje molekularnih svojstava (MPP)

Povezanost kemije i informatike

Predviđanje molekularnih svojstava (eng. molecular property prediction – MPP) je područje koje povezuje kemiju i informatiku, a bazira se na računalnim metodama pomoću kojih je moguće predvidjeti fizikalna i kemijska svojstva, položaje i karakteristike različitih molekula. Područje je popularno za razvijanje zbog svojeg velikog potencijala u proizvodnji i otkrivanju novih lijekova, a primjenu nalazi i u znanosti o materijalima te znanosti o okolišu.

Veliki potencijal

Postoje različita svojstva molekula koja se mogu predviđati; molekularna masa, polarnost molekula, ionizacijski potencijal, afinitet elektrona, absorpcijski spektar, točku vrenja, toksičnost kemikalija i sigurnost za okoliš, utjecaj struktura na biološku aktivnost, a kod razvoja lijekova može se pridvidjeti kako lijek reagira u tijelu, tj. kako se apsorbira i distribuira po metabolizmu.

Razvijene su već različite metode za predviđanje raznih svojstava molekula, svaka sa svojim prednostima i manama. Jedna od prvih je teorija funkcijske gustoće koja predviđa dosta točno, ali je računalno preteška za velike molekule. Ostale teorije koriste npr. brojevne predodžbe molekularnih struktura ili pojednostavljenu potencijalnu energijsku funkciju kako bi napravile model molekula. Često se koristi strojno učenje, regresija i klasifikacija te neuralne mreže. Duboke neuralne mreže mogu obraditi kompliciranе odnose i uzorke, ali zahtijevaju veliki skup podataka koji nije uvijek moguće nabaviti.

Različite metode se fokusiraju na različita svojstva, a predviđanja se mogu odvijati na temelju molarne mase molekula, broju atoma, vrsti atoma, vrsti veza između atoma unutar molekule, broju slobodnih elektronskih parova, obliku, volumenu i specifičnoj površini molekula i sl.

Kao što je već spomenuto, glavna je primjena u istraživanju novih lijekova gdje se proučava koje proteine molekule lijeka mogu ciljati, toksičnost lijekova i općenito potencijalni kandidati za nove lijekove ili daljnji njihov razvoj. U dizajnu materijala gleda se kako spojevi djeluju na katalizu i razvoj uvijek popularnih polimera. U znanosti o okolišu bitan je utjecaj raznih novih spojeva na okoliš te sama sigurnost, dok je u biologiji zanimljivo proučavati interakcije između proteina i liganada, stabilnost proteina te općenito biološke procese.

Međutim, postoje razne prepreke u razvijanju dobre metode koja može točno predvidjeti svojstva. Njačešće su potrebne velike baze podataka koje nije uvijek moguće dobiti ili su

ograničene, metode moraju dobro predviđati nove, nikad viđene molekule i spojeve što je nekad teško za programirati, pogotvo za kompliciranije molekule. Velike molekule zahtijevaju kompliciraniju i dugotrajniju obradu te se ne mogu uvijek sasvim točno analizirati. Modeli dubokog učenja daju dobre rezultate, ali nekada nije jasno na koji način se dođe do tih rezultata.

Uni-Mol metoda

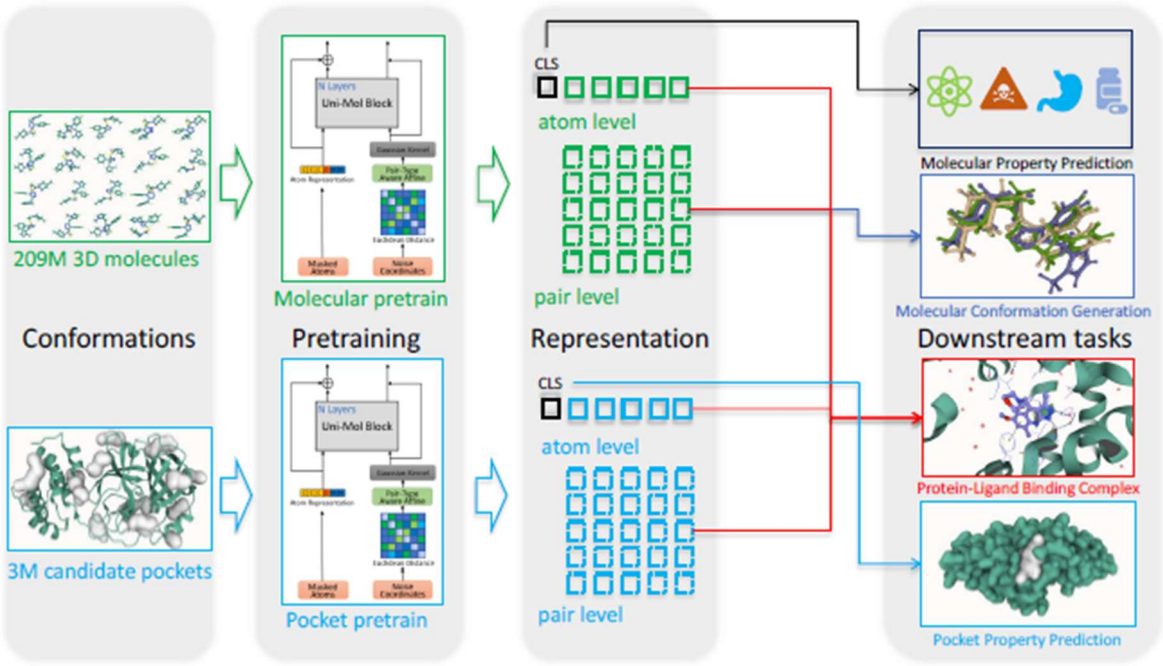
Zašto je potrebna?

Učenje molekularne prezentacije, tj. izgleda i položaja molekula i atoma, postaje sve popularnije zbog primjene u dizajnu lijekova te se istražuju nove metode kako bi se položaji molekula mogli predvidjeti što bolje. Međutim, većina metoda kreće od toga da se molekule predstavlaju kao 1D sekvence ili 2D topografski grafovi čime se gubi jako puno bitnih informacija jer se uopće ne uzima u obzir geometrija u prostoru što čini veliku razliku u kemijskim spojevima.

Naime, u kemiji se razlikuju izomeri – vrsta molekula koje imaju iste atome i molekularnu masu, ali je raspored atoma u prostoru drugačiji što onda mijenja sva svojstva molekula; kemijska, fizička i biološka. Od samo jednog spoja postoji više različitih izomera, a broj izomera eksponencijalno raste kako se molekule povećavaju. Tako npr. za ugljikovodike koji imaju 5 C atoma postoji 3 izomera, za 9 C atoma već ima 36 različitih izomera, a za 20 C atoma postoji čak 366319 izomera.[12] Izomeri mogu biti različitih vrsta, a glavna je podjela na konstitucijske ili strukturne i stereoizomere. Razlike u svojstvima izomera istog spoja mogu biti tako velike da su neki izomeri ljekoviti i upotrebljivi, dok su drugi otrovni ili eksplozivni. Jasno je onda da geometrija u kemiji čini veliku razliku te ako se zanemaruje u predviđanju molekula, može doći do velikih pogrešaka.

Dijelovi metode

Uni-Mol metoda predstavlja univerzalni okvir za učenje 3D molekularne reprezentacije uzimajući u obzir spomenuti problem geometrije.



Slika 9: Ilustracija sheme rada Uni-Mol metode [13]

Uni-Mol sadrži dva modela po kojima radi; molekularno predtreniranje koje radi na bazi od 209 milijuna 3D molekula i „džepno“ predtreniranje koje radi na bazi od 3 milijuna. Modeli su međusobno nezavisni i svaki radi sam da bi se na kraju za rezultat kombinirali i spojili u jedno. Uzima 3D pozicije i kao ulazne i kao izlazne podatke. Sastoji se od tri glavna dijela; oslonca, predtreniranja i podešavanja. [13]

Većina prijašnjih metoda prikazivala je molekule kao grafove dok se tu molekule tretiraju kao 3D čvorovi gdje svaki čvor ima odgovrajuću vrstu atoma i 3D koordinatu. Kako bi se u potpunosti mogli povezati ti čvorovi, glavni dio (spomenuti oslonac) cijele metode je Transformer. Jedini je problem što je Transformer originalno postavljen za prevođenje jezika, a ne za 3D geometriju pa se moraju učiniti određene preinake kako bi ulazni podaci bili povoljni za obradu. Potrebno je programirati točno određene pozicije i položaje u prostoru koji su jednaki za sve molekule kako bi Transformer mogao izvršiti sve zadatke. Iz tog razloga se određuju udaljnosti između atoma na temelju Euklidske udaljenosti (najkraći razmak između dvije točke) i Gaussove gustoće.

U predtreningu koriste se dvije baze podataka; 3D strukture organskih molekula i 3D strukture proteina te se zatim oba modela treniraju koristeći cijelu bazu podataka. Proteini su vrlo bitni u industriji lijekova pa se taj dio metode koristi u predviđanju struktura i interakcija proteina i liganada. Za predtrening se miču duplicirane molekule i iznimke te u bazi podataka ostaje 19

milijuna molekula. Kako bi se pokrilo što više različitih mogućnosti, za svaku se molekulu generiraju 10 različitih konformacija i dodaju se još neke 2D konformacije (bazirane na grafičkom prikazu) za molekule za koje se ne mogu generirati 3D konformacije. Bitno je da program može učiti tijekom treninga i kasnije pa se atomi maskiraju po istom principu kao što su se u Transformeru maskirale iduće riječi kako bi mogao sam predviđati i onda usporediti dobiveno. Za svaku se molekulu postave koordinate centralnog atoma što onda predstavlja ukupnu cijelu molekulu. Međutim, onda je programu prejednostavno za predvidjeti gdje su susjedni atomi jer zna kolike su udaljenosti između atoma pa ne može učiti na taj način. Zbog toga se dodaju šumovi na 15% slučajno odabranih koordinata atoma kako bi se program konstantno sam unaprijeđivao. Koordinate atoma sa šumovima se pretpostavljaju na dva načina – preko predviđanja udaljenosti između atoma i predviđanjem točnih koordinata te se opet na kraju onda uspoređuju rezultati.

U završnom dijelu metode, podešavanju, se unaprijeđuju 3D prikazi s istom bazom podataka kao i u predtreningu, a dijele se na one koje uzimaju 3D zadatke u predviđanju i one kojima 3D nije potreban. 3D predviđanja nisu potrebna ako je moguće problem riješiti linearno, tj. na taj način prikazati molekulu. To se koristi kod predviđanja određenih svojstava, usporedbe sličnosti molekula i predviđanje afiniteta između proteina i liganada. S druge strane, nekada je iznimno potrebno koristiti 3D predviđanja kao npr. kod konformacije molekula gdje je glavni naglasak na rasporedu atoma u prostoru. Inače to zahtijeva puno vremena pa se često koriste i podaci iz grafičkih prikaza kako bi se ubrzao prijenos informacija. Također, vrlo bitna predviđanja koja koriste prednosti 3D-a su u izradi lijekova, konkretno kompleksne strukture izgrađene od proteina i molekularnih liganada. U obzir treba uzeti razne konformacije struktura, a obično postoje za svaku tako kompleksnu molekulu 3 rotacije i 3 translacije. Prvo se uzimaju podaci iz predtreninga, a zatim se u metodi dodaju 4 dodatna sloja koja podešavaju i uče udaljenost atoma u molekuli. Nakon toga se na slučajno odabrano mjesto postavi ligand i gledaju se sve mogućnosti i udaljenosti od susjednih atoma. Ovakav proces je produktivan i vrlo brz u odnosu na neke tradicionalnije metode.

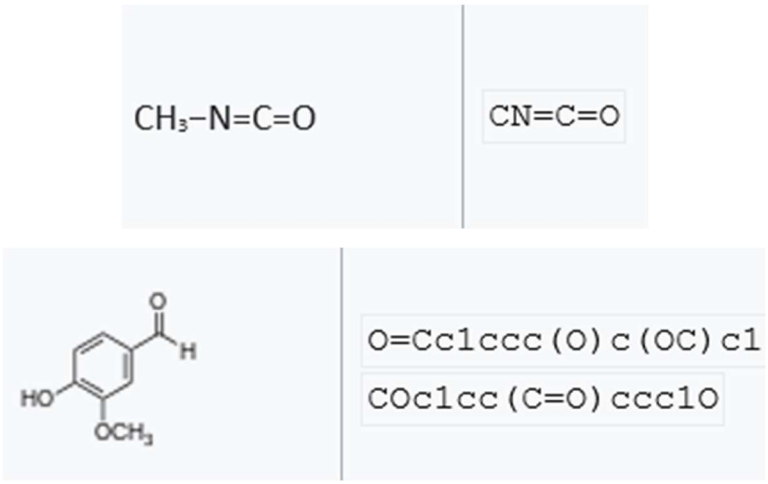
Računalni dio

Python

Uni-Mol metoda može se isprogramirati u programskom jeziku Pythonu. Ima visoke razine ugrađenih podatkovnih struktura, podržava module i pakete te je općenito jednostavan jezik za naučiti jer ima par ključnih riječi koje su vizualno naglašene različitim bojama i jasno određene sintakse. Popularan je za brzo korištenje i razvoj aplikacija, a u njemu se mogu i spajati već postojeće komponente od prije. Pogreške se lako prikazuju, može se mijenjati kod unutar samog programa red po red, a program razlikuje globalne i lokalne razine.

SMILES

Spomenuti kod u Pythonu koristi SMILES (eng. Simplified Molecular Input Line Entry Specification) što je specifikacija koja omogućava jednostavan unos i opisivanje kemijskih vrsta i struktura. SMILES se koristi za dvodimenzionalne crteže molekula kao i za trodimenzionalne modele. Atomi se označavaju simbolima kao u periodnom sustavu te se posebno naglašavaju različite vrste veza, prstenaste strukture i aromatičnost. Sustav zna koliko bi svaki atom trebao imati valenciju pa se atomi vodika (H) mogu i ne trebaju pisati, baš kao i u organskoj kemiji na papiru.



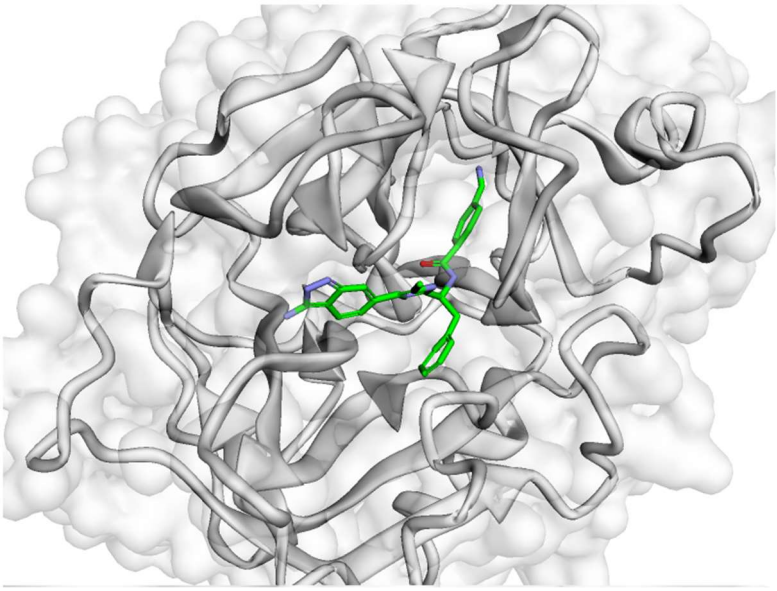
Slika 10: Primjer dviju molekula prikazanih u SMILES [18]

Na Slici 10 se može primijetiti da se atomi vodika posebno ne naglašavaju te da se prate sva uobičajena pravila iz organske kemiju.

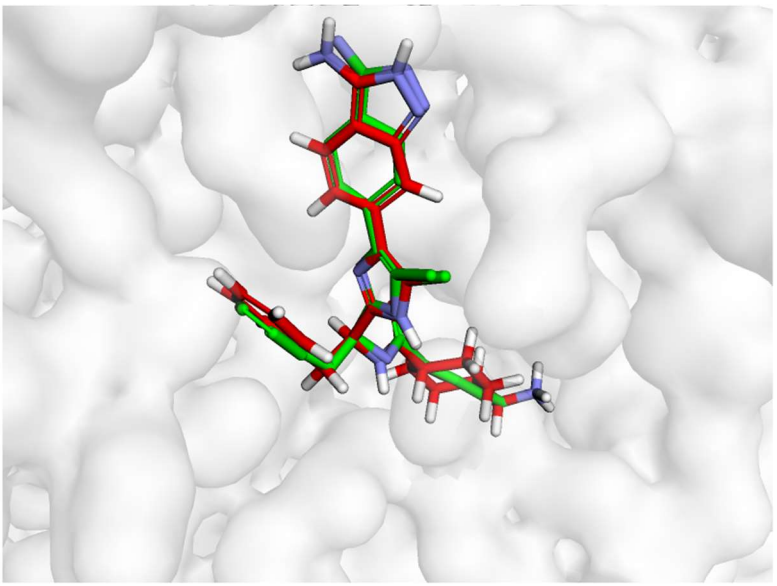
Vizualizacija molekula

Sami kod preuzet je sa web stranice[14]. Kod najprije uvozi sve potrebne podatke i mape iz zadanih biblioteka, a zatim se definiraju konstante, glavni atomi te se uklanjaju numeričke vrijednosti iz naziva samih atoma. Potom se definiraju različite funkcije za očitavanje podataka o strukturi proteina i liganada, generiranje različitih konformacija molekula, računanje sila i polja između molekula, određivanje koordinata, obrađivanje podataka, spajanje proteina i liganada te konačno funkcije za predviđanja molekularnih spajanja i interakcija proteina i liganada.

Na kraju se može vidjeti vizualni prikaz dobivenih molekula. Najprije se vidi molekula unutar veće strukture oko nje, a zatim sama molekula koja je, radi jasnijeg prikaza, označena u više boja. Dobivene Slike 11 i 12 su vrlo praktične jer se mogu okretati u svim smjerovima, približavati i udaljavati te se cijela struktura može rotirati i vidjeti iz svakog kuta.



Slika 11: Molekula unutar strukture [14]



Slika 12: Dobivena molekula [14]

Zaključak

Strojno učenje od početka svog razvoja pa do danas sve više dobiva na popularnosti zbog uviđenih prednosti i mogućnosti primjene. Općenito računalni programi mogu zamijeniti ljudski rad jer rade efikasnije, računaju točnije i brže te mogu obraditi puno više podataka. Strojno učenje prikuplja i obrađuje različite podatke te traži algoritme po kojima bi ih moglo povezati, uz cilj da tijekom tog procesa uči kako bi se novi ulazni podaci mogli obraditi sami bez ponovnog programiranja. Neuralne mreže su skup već razrađenih podataka i algoritama, a baziraju se na regresijskoj analizi. Svojim načinom rada podsjećaju na ljudski mozak pa su po tome i dobile ime, a matematički se baziraju na kompoziciji funkcija te upotrebi vektora i matrica. Neuralne mreže ipak ne mogu pamtiti dugoročno ili ako je između podataka preveliki razmak. Stoga je razvijen transformer, poboljšana verzija neuralnih mreža. Transformer koristi metodu pozornosti u kojoj stavlja fokus na samo trenutno bitne informacije, a ostale zanemaruje. Najviše se koristi za prevođenje jezika, ali određenim modifikacijama ima primjenu i u drugim područjima.

U kemijskom inženjerstvu strojno učenje može se primijeniti za predviđanje različitih svojstava molekula. Razvijene su različite metode, a Uni-Mol metoda je dobra jer uzima u obzir geometriju i 3D položaje molekula bez kojih se gubi veliki broj informacija te rezultati u suprotnome nisu toliko točni. Uni-Mol metoda može se isprogramirati u programskom jeziku Pythonu, a cilj je dobiti jasan vizualni prikaz predviđenog spajanja proteina i liganada. Metode poput Uni-Mola od iznimne su važnosti u proizvodnji lijekova i predviđanju novih jer program može predvidjeti potencijalni lijek koji još nije fizički napravljen u laboratoriju.

Prilozi

U prilogu su kodovi korišteni na web stranici [14] za predviđanje položaja molekula na temelju Uni-Mol metode kao što je opisano.

Za instalirati sve povezane dokumente i mape je potrebno:

```
%%bash
#@title Install dependencies

GIT_REPO='https://github.com/dptech-corp/Uni-Mol'
UNICORE_URL='https://github.com/dptech-corp/Uni-Core/releases/download/0.0.2/unicore-0.0.1+cu116torch1.13.1-cp39-cp39-linux_x86_64.whl'
DOCKING_DATA_URL='https://github.com/dptech-corp/Uni-Mol/releases/download/v0.1/CASF-2016.tar.gz'
DOCKING_WEIGHT_URL='https://github.com/dptech-corp/Uni-Mol/releases/download/v0.1/binding_pose_220908.pt'
if [ ! -f UNIMOL_READY ]; then
  wget -q ${UNICORE_URL}
  pip3 -q install "https://github.com/dptech-corp/Uni-Core/releases/download/0.0.3/unicore-0.0.1+cu117torch2.0.0-cp310-cp310-linux_x86_64.whl"
  rm -rf ./Uni-Mol
  git clone -b main ${GIT_REPO}
  pip3 install -q ./Uni-Mol/unimol
  pip install -q rdkit
  pip install -q biopandas
  wget -q ${DOCKING_DATA_URL}
  tar -xzf "CASF-2016.tar.gz"
  wget -q ${DOCKING_WEIGHT_URL}
  pip install -q py3Dmol
  touch UNIMOL_READY
fi
```

Glavni dio koda – pokretanje Uni-Mol

```
import os
import sys
import numpy as np
import pandas as pd
import biopandas
import lmdb
from biopandas.pdb import PandasPdb
from rdkit import Chem
from rdkit.Chem import AllChem
from sklearn.cluster import KMeans
from rdkit.Chem import rdMolTransforms
from rdkit.Chem.rdMolAlign import AlignMolConformers
from unimol.utils.docking_utils import docking_data_pre, ensemble_iterations
from tqdm import tqdm
import pickle
import re
import json
import copy

CASF_PATH = "CASF-2016"
main_atoms = ["N", "CA", "C", "O", "H"]
```

Normalizacija atoma:

```
def load_from_CASF(pdb_id):
    try:
        pdb_path = os.path.join(CASF_PATH, "casf2016", pdb_id + "_protein.pdb")
        pmol = PandasPdb().read_pdb(pdb_path)
        pocket_residues = json.load(
            open(os.path.join(CASF_PATH, "casf2016.pocket.json"))
        )[pdb_id]
        return pmol, pocket_residues
    except:
        print("Currently not support parsing pdb and pocket info from local files.")


def normalize_atoms(atom):
    return re.sub("\d+", "", atom)


def single_conf_gen(tgt_mol, num_confs=1000, seed=42, removeHs=True):
    mol = copy.deepcopy(tgt_mol)
    mol = Chem.AddHs(mol)
    allconformers = AllChem.EmbedMultipleConfs(
        mol, numConfs=num_confs, randomSeed=seed, clearConfs=True
    )
    sz = len(allconformers)
    for i in range(sz):
        try:
            AllChem.MMFFOptimizeMolecule(mol, confId=i)
        except:
            continue
    if removeHs:
        mol = Chem.RemoveHs(mol)
    return mol
```

Micanje nepotrebno naglašenih atoma vodika:

```
def clustering_coords(mol, M=1000, N=100, seed=42, removeHs=True):
    rdkit_coords_list = []
    rdkit_mol = single_conf_gen(mol, num_confs=M, seed=seed, removeHs=removeHs)
    noHsIds = [
        rdkit_mol.GetAtoms()[i].GetIdx()
        for i in range(len(rdkit_mol.GetAtoms()))
        if rdkit_mol.GetAtoms()[i].GetAtomicNum() != 1
    ]
    ### exclude hydrogens for aligning
    AlignMolConformers(rdkit_mol, atomIds=noHsIds)
    sz = len(rdkit_mol.GetConformers())
    for i in range(sz):
        _coords = rdkit_mol.GetConformers()[i].GetPositions().astype(np.float32)
        rdkit_coords_list.append(_coords)

    ### exclude hydrogens for clustering
    rdkit_coords_flatten = np.array(rdkit_coords_list)[:, noHsIds].reshape(sz, -1)
    ids = (
        KMeans(n_clusters=N, random_state=seed)
        .fit_predict(rdkit_coords_flatten)
        .tolist()
    )
    coords_list = [rdkit_coords_list[ids.index(i)] for i in range(N)]
    return coords_list
```

```
def parser(pdb_id, smiles, seed=42):
    pmol, pocket_residues = load_from_CASF(pdb_id)
    pname = pdb_id
    pro_atom = pmol.df["ATOM"]
    pro_hetatm = pmol.df["HETATM"]
```

```
pro_atom["ID"] = pro_atom["chain_id"].astype(str) + pro_atom[
    "residue_number"
].astype(str)
pro_hetatm["ID"] = pro_hetatm["chain_id"].astype(str) + pro_hetatm[
    "residue_number"
].astype(str)

pocket = pd.concat(
    [
        pro_atom[pro_atom["ID"].isin(pocket_residues)],
        pro_hetatm[pro_hetatm["ID"].isin(pocket_residues)],
    ],
    axis=0,
    ignore_index=True,
)

pocket["normalize_atom"] = pocket["atom_name"].map(normalize_atoms)
pocket = pocket[pocket["normalize_atom"] != ""]
patoms = pocket["atom_name"].apply(normalize_atoms).values.tolist()
pcoords = [pocket[["x_coord", "y_coord", "z_coord"]].values]
side = [0 if a in main_atoms else 1 for a in patoms]
residues = (
    pocket["chain_id"].astype(str) + pocket["residue_number"].astype(str)
).values.tolist()
```

Stvaranje konformacija liganada:

```
    # generate ligand conformation
    M, N = 100, 10
    mol = Chem.MolFromSmiles(smiles)
    mol = Chem.AddHs(mol)
    AllChem.EmbedMolecule(mol, randomSeed=seed)
    latoms = [atom.GetSymbol() for atom in mol.GetAtoms()]
    holo_coordinates = [mol.GetConformer().GetPositions().astype(np.float32)]
    holo_mol = mol
    coordinate_list = clustering_coords(mol, M=M, N=N, seed=seed, removeHs=False)
    mol_list = [mol] * N

    return pickle.dumps(
        {
            "atoms": latoms,
            "coordinates": coordinate_list,
            "mol_list": mol_list,
            "pocket_atoms": patoms,
            "pocket_coordinates": pcoords,
            "side": side,
            "residue": residues,
            "holo_coordinates": holo_coordinates,
            "holo_mol": holo_mol,
            "holo_pocket_coordinates": pcoords,
            "smi": smiles,
            "pocket": pname,
        },
        protocol=-1,
    )
```

```python
def write_lmdb(pdb_id, smiles_list, seed=42, result_dir="./results"):
    os.makedirs(result_dir, exist_ok=True)
    outputfilename = os.path.join(result_dir, pdb_id + ".lmdb")
    try:
        os.remove(outputfilename)
    except:
        pass
    env_new = lmdb.open(
        outputfilename,
        subdir=False,
        readonly=False,
        lock=False,
        readahead=False,
        meminit=False,
        max_readers=1,
        map_size=int(10e9),
    )
    for i, smiles in enumerate(smiles_list):
        inner_output = parser(pdb_id, smiles, seed=seed)
        txn_write = env_new.begin(write=True)
        txn_write.put(f"{i}".encode("ascii"), inner_output)
    txn_write.commit()
    env_new.close()


# @title Run Uni-Mol Binding Pose Prediction

# @markdown Currently this scripts only support CASF-2016 dataset with given pockets res

# @markdown You can input multiple SMILES, split by ','.

# @markdown If SMILES is not given, the default one in the complex will be used.
```

```python
pdb_id = "4ty7"  # @param {type:"string"}
pdb_id = pdb_id.lower()
casf_collect = os.listdir(os.path.join(CASF_PATH, "casf2016"))
casf_collect = list(set([item[:4] for item in casf_collect]))
if pdb_id not in casf_collect:
  warning_str = "{} is not int CASF-2016 dataset, Please select from \n".format(pdb_id)
  for i in range(15):
    warning_str += "{}\n".format(','.join(casf_collect[20*i:20*(i+1)]))
  raise Exception(warning_str)
supp = Chem.SDMolSupplier(os.path.join(CASF_PATH, "casf2016", pdb_id + "_ligand.sdf"))
mol = [mol for mol in supp if mol][0]
ori_smiles = Chem.MolToSmiles(mol)
smiles = ""  # @param {type:"string"}
seed = 42  # @param {type:"number"}
data_path = "./CASF-2016"
results_path = "./results/"
weight_path = "/content/binding_pose_220908.pt"
batch_size = 8
dist_threshold = 8.0
recycling = 3
```

Provjera SMILES pravila:

```
if smiles.split(",") == 0 or smiles == "":
    print("No other smiles inputs")
    smiles_list = [ori_smiles]
else:
    print("Docking with smiles: {}".format(smiles))
    smiles_list = smiles.split(",")

write_lmdb(pdb_id, smiles_list, seed=seed, result_dir=data_path)

!python ./Uni-Mol/unimol/unimol/infer.py --user-dir ./Uni-Mol/unimol/unimol $data_path -
       --results-path $results_path \
       --num-workers 8 --ddp-backend=c10d --batch-size $batch_size \
       --task docking_pose --loss docking_pose --arch docking_pose \
       --path $weight_path \
       --fp16 --fp16-init-scale 4 --fp16-scale-window 256 \
       --dist-threshold $dist_threshold --recycling $recycling \
       --log-interval 50 --log-format simple
```

```
def generate_docking_input(
    predict_file, reference_file, tta_times=10, output_dir="./results"
):
    (
        mol_list,
        smi_list,
        pocket_list,
        pocket_coords_list,
        distance_predict_list,
        holo_distance_predict_list,
        holo_coords_list,
        holo_center_coords_list,
    ) = docking_data_pre(reference_file, predict_file)
    iter = ensemble_iterations(
        mol_list,
        smi_list,
        pocket_list,
        pocket_coords_list,
        distance_predict_list,
        holo_distance_predict_list,
        holo_coords_list,
        holo_center_coords_list,
        tta_times=tta_times,
    )
    for i, content in enumerate(iter):
        pocket = content[3]
        output_name = os.path.join(output_dir, "{}.{}.pkl".format(pocket, i))
        try:
            os.remove(output_name)
        except:
            pass
        pd.to_pickle(content, output_name)
```

```
predict_file = os.path.join(results_path, "content_" + pdb_id + ".out.pkl")
reference_file = os.path.join(data_path, pdb_id + ".lmdb")
generate_docking_input(
    predict_file, reference_file, tta_times=10, output_dir=results_path
)
for i, smiles in enumerate(smiles_list):
    print("Docking {}".format(smiles))
    input_path = os.path.join(results_path, "{}.{}.pkl".format(pdb_id, i))
    ligand_path = os.path.join(results_path, "docking.{}.{}.sdf".format(pdb_id, i))
    cmd = "python ./Uni-Mol/unimol/unimol/utils/coordinate_model.py --input {} --output-
        input_path, ligand_path
    )
    os.system(cmd)
```

Završni dio koda potreban za vizualizaciju molekula u programu:

```
import py3Dmol
import matplotlib.pyplot as plt
pdb_path = os.path.join(CASF_PATH, 'casf2016', pdb_id+'_protein.pdb')
ligand_path = os.path.join(results_path, "docking.{}.{}.sdf".format(pdb_id,0))
gt_ligand_path = os.path.join(CASF_PATH,'casf2016',pdb_id+'_ligand.sdf')
view = py3Dmol.view()
view.removeAllModels()
pdb_path = os.path.join(CASF_PATH, 'casf2016', pdb_id+'_protein.pdb')
view.addModel(open(pdb_path,'r').read(),format='pdb')
view.setStyle({'cartoon': {'arrows':True, 'tubes':False, 'style':'oval', 'color':'white'
view.addSurface(py3Dmol.VDW,{'opacity':0.5,'color':'white'})

view.addModel(open(ligand_path,'r').read(),format='sdf')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'greenCarbon','radius':0.2}})

view.zoomTo(viewer=(100,0))
view.show()

view.removeAllModels()
view.addModel(open(ligand_path,'r').read(),format='sdf')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'greenCarbon','radius':0.2}})
```

```
view.addModel(open(gt_ligand_path,'r').read(),format='sdf')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'redCarbon','radius':0.2}})

view.zoomTo(viewer=(100,0))
view.show()
```

Literatura

[1] Raschka S., Mirjalili C., Python Machine Learning

[2] Linear regression https://en.wikipedia.org/wiki/Linear_regression, pristupljeno 07.09.2023.

[3] Burkov A., The Hundred-Page Machine Learning Book

[4] Sigmoid function https://www.learndatasci.com/glossary/sigmoid-function/, pristupljeno 07.09.2023.

[5] What Is a Neural Network? https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=A%20neural%20network%20is%20a,organic%20or%20artificial%20in%20nature, pristupljeno 13.08.2023.

[6] neuronska mreža https://www.enciklopedija.hr/natuknica.aspx?ID=43562, pristupljeno 13.08.2023.

[7] Transformer (machine learning model) https://en.wikipedia.org/wiki/Transformer_(machine_learning_model), pristupljeno 12.08.2023.

[8] Transformer Neural Networks: A Step-by-Step Breakdown https://builtin.com/artificial-intelligence/transformer-neural-network, pristupljeno 12.08.2023.

[9] Lu C., Liu Q., Wang C., Huang Z., Lin P., He L., Molecular Property Prediction: A Multilevel Quantum Interactions Modeling Perspective

[10] Fang X., Liu L., Lei J., He D., Zhang S., Zhou J., Wang F., Wu H., Wang H., Geometry-enhanced molecular representation learning for property prediction

[11] Shen J., Nicolaou C. A., Molecular property prediction; recent trends in the era of artificial intelligence

[12] Alkanes https://www.angelo.edu/faculty/kboudrea/molecule_gallery/01_alkanes/00_alkanes.htm#:~:text=There%20are%20two%20geometric%20isomers,the%20cis%20and%20trans%20form., pristupljeno 20.08.2023.

[13] Zhou G., Gao Z., Ding Q., Zheng H., Xu H., Wei Z., Zhang L., Ke G., "Uni-Mol: A Universal 3D Molecular Representation Learning Framework. " ChemRxiv (2022)

[14] Uni-Mol Binding Pose Prediction Colab https://colab.research.google.com/github/dptech-corp/Uni-Mol/blob/main/unimol/notebooks/unimol_binding_pose_demo.ipynb, pristupljeno 07.08.2023.

[15] Python - Overview https://www.tutorialspoint.com/python/python_overview.htm, pristupljeno 10.08.2023.

[16] What is Python? Executive Summary https://www.python.org/doc/essays/blurb/, pristupljeno 10.08.2023.

[17] Python (programming language)
https://en.wikipedia.org/wiki/Python_(programming_language), pristupljeno 10.08.2023.

[18] Simplified molecular-input line-entry system
https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system, pristupljeno 21.08.2023.