Umjetna inteligencija, entropija i jezični modeli

Luka Mikec[1]

Uvod

Umjetna inteligencija (engl. *artificial intelligence*, AI) označava područja matematike i računarstva koja se, ugrubo, bave algoritmima inspiriranim ljudskim mozgom, primjerice, logičkim zaključivanjem ili neuronskim mrežama. Tipičan je primjer problema rješivog ovakvim algoritmima strojno prepoznavanje rukom napisanog teksta. Taj se problem može sa zadovoljavajućom točnošću riješiti koristeći neuronske mreže konfigurirane i optimizirane ('naučene') za prepoznavanje simbola na slikama. Fraza 'umjetna inteligencija' ima i bitno drugačije značenje izvan znanosti (primjerice, u znanstvenoj fantastici): pokušaj strojne simulacije ljudskog razmišljanja bez neke posredne svrhe. Posljednjih nekoliko godina potonje je neformalno značenje ušlo u domenu znanosti, i to umjetne inteligencije. Pokazalo se da možemo dovoljno efikasno i jeftino simulirati ljudsko razmišljanje da, primjerice, provjera gramatičke ispravnosti teksta simulacijom ljudskog razmišljanja postane otprilike jednako efikasna kao dosadašnji pristupi temeljeni na zadanim pravilima i jednostavnijim heuristikama. Za temeljno otkriće na ovom putu, tzv. transformeri, zaslužna je grupa istraživača iz Googlea ([9]). Najpoznatija su primjena transformera tzv. veliki jezični modeli koji pokreću alate poput Googleovog Geminija, OpenAI-jevog ChatGPT-a, ili Anthropicovog Claudea. Svi su navedeni alati besplatno dostupni na Internetu, ili nude besplatan probni period.

Spomenimo da u umjetnoj inteligenciji okvirno razlikujemo dva pristupa. **Simbolički pristup** temelji se na sažimanju ljudskog znanja u jednostavne činjenice, i potom korištenju tih činjenica za odgovaranje na složena pitanja. Primjerice, medicinske ustanove mogle su konstruirati tzv. ekspertne sisteme koji opisuju povezanost simptoma i bolesti. Unošenjem pacijentovog zdravstvenog stanja, sistem bi mogao navesti relevantne bolesti. Iako su ovakvi sistemi doista korišteni u praksi, imaju lako uočljiv nedostatak: ljudsko je znanje uglavnom formulirano u terminima vjerojatnosti i statistike, a ne crno-bijelih činjenica.

Drugi je pristup **strojno učenje**. Umjesto pisanja pravila, ovaj pristup koristi algoritme za automatiziranu izradu modela na temelju podataka. Pritom "model" ovdje označava bilo kakav postupak koji pretvara opis problema u rješenje tog problema. Treniranje na podacima najčešće znači da se model formira na temelju (obično velikog broja) primjera problema i rješenja. Tipičan primjer algoritma koji stvara modele, konkretno neuronske mreže, jest *backpropagation*, algoritam koji na temelju velikog broja primjera ulaza i izlaza stvara neuronsku mrežu. Primjerice, ako imamo velik broj fotografija rukom napisanih slova te uz svaku sliku priložimo slovo koje se zaista nalazi na slici, backpropagation može stvoriti neuronsku mrežu koja će za bilo koju sliku pokušati pogoditi koje se slovo nalazi na slici. Uspješnost pogađanja ovisi prvenstveno o broju slika korištenih u treniranju.

Veliki jezični modeli mogu se promatrati kao specifično strukturirane neuronske mreže. "Problem" koji rješavaju jest pitanje: "Koja bi riječ najbolje nastavljala dani tekst?". Primjerice, koja riječ najbolje nastavlja tekst "Danas je lijep".

[1] e-pošta: luka.mikec@gmail.com

Predviđanje i entropija jezika

Teorijski temelji predviđanja nastavka teksta dolaze iz sredine 20. stoljeća kad je američki matematičar Claude Shannon postavio temelje matematičke teorije informacija. Tu je veliku ulogu imao članak *Prediction and Entropy of Printed English* [5]. Shannon se bavio pitanjem predvidljivosti engleskog jezika, odnosno mjerom koju je nazvao **entropija**. Ona, ugrubo, mjeri količinu 'iznenađenja' u tekstovima nekog jezika, uzimajući u obzir sve 'prirodno' nastale tekstove.

Primjerice, ako imamo neki tekst na engleskom jeziku koji završava slovom "Q", vrlo je vjerojatno da je sljedeće slovo "U". To je stoga što u tekstovima na engleskom jeziku nakon slova "Q" gotovo uvijek slijedi slovo "U". Što je predvidljivost viša, to je entropija niža. Minimalnu entropiju postigao bi jezik u kojem je svaki mogući tekst unaprijed determiniran, tj. svaki početak teksta ima jedinstven nastavak. Primjerice, jezik u kojem nakon 'A' uvijek dolazi 'B', nakon 'B' uvijek dolazi 'C' itd. Očito je takav jezik potpuno predvidljiv (i beskoristan). Maksimalnu entropiju postiže potpuno nepredvidljiv jezik, a to je jezik u kojem je svaki simbol jednako vjerojatan kao i svaki drugi. Primjerice, nakon "Ž" može doći samoglasnik, ali jednako je vjerojatno da se može pojaviti i bilo koje drugo slovo, poput još jednog slova "Ž". U jeziku s maksimalnom entropijom svaki je niz znakova smislena riječ. Takav jezik omogućava kraće zapisivanje informacija, jer je za istu duljinu teksta dostupan puno veći broj mogućih tekstova te duljine. Primjerice, kad stvorite ZIP datoteku, efektivno ste preveli tekst iz (primjerice) hrvatskog jezika u neki drugi (ljudima izravno nečitljiv) jezik s vrlo visokom entropijom. Što je algoritam komprimiranja efikasniji, entropija spomenutog internog jezika bit će veća.[2]

Shannon je na umu imao vrlo praktične primjene entropije. Koristeći informacije o frekvencijama pojavljivanja simbola u jeziku možemo izgraditi statistički model jezika. Ipak, ovakav model nije u tom obliku posebno koristan, jer frekvencije slova zapravo ne govore puno o jeziku. Primjerice, engleski i francuski ne bi se bitno razlikovali u ovom modelu. Modeli jezika postaju puno zanimljiviji uz malu promjenu. Umjesto da gledamo koje slovo slijedi nakon kojeg slova, radije analiziramo ***n*-grame**, odnosno nizove od n slova. Primjerice, tražimo vjerojatnost pojavljivanja simbola ne samo s obzirom na prethodni simbol, već na temelju $n-1$ prethodnih simbola.

Označimo s $P(x|y)$ vjerojatnost da simbol x slijedi nakon niza simbola y. Pretpostavimo da smo nekako (primjerice, analizom svih knjiga na hrvatskom i engleskom jeziku) aproksimirali $P(x|y)$ za sve 3-grame y i svako slovo x. Drugim riječima, znamo da je vrlo vjerojatno da nakon "čaš" slijedi "a", i znamo da je vrlo malo vjerojatno da nakon "čaš" slijedi "ž". Sada možemo stvoriti jednostavan algoritam generiranja teksta. Uzmimo za primjer tekst "Danas je lijep d". Pogledajmo posljednja tri slova (točnije, simbola) u tekstu: to su "p", razmak i "d". Sada pogledajmo za koji smo simbol x ranije pronašli da

[2] Na ovom se principu temelji Huffmanovo kodiranje, jednostavan algoritam za kompresiju koji je temelj gotovo svih danas korištenih algoritama za kompresiju. Ideja je da za dani tekst 'izmislimo' umjetan jezik za koji vrijedi da je prijevod danog teksta u taj jezik najkraći mogući, među svim mogućim jezicima. Umjesto da svaki simbol koristi fiksan broj bitova kao što, primjerice, ASCII kod koristi 7 ili 8 bitova za svaki simbol, česti simboli u ovom umjetnom jeziku koristit će manji broj bitova. Primjerice, slova "A" i "E" koristit će nekoliko bitova, dok će slovo "Ž" koristiti veći broj bitova. Ukupan broj bitova u prijevodu gotovo bilo kojeg prirodno nastalog teksta bit će značajno manji nego u ASCII kodiranju istog teksta, jer će većina znakova biti kodirana s manje od 8 bitova. Nizove bitova potrebno je odabrati na način da i dalje možemo pročitati cijeli tekst. Naime, s ovim pristupom ne možemo prilikom čitanja razlomiti tekst na dijelove od po 8 bitova kao u slučaju ASCII zapisa teksta, već početak i kraj niza bitova pojedinog slova mora biti drugačije prepoznatljiv. Konkretno, jezik mora imati tzv. *prefix-free* svojstvo.

je vrijednost $P(x|pd)$ najviša. Pretpostavimo da je to slovo "a". Sad dodajemo slovo "a" u naš tekst i dobivamo "Danas je lijep da". Potom određujemo za koje je slovo x vrijednost $P(x|da)$ najviša (možda je to slovo "n") i ponavljamo postupak proizvoljno dugo.

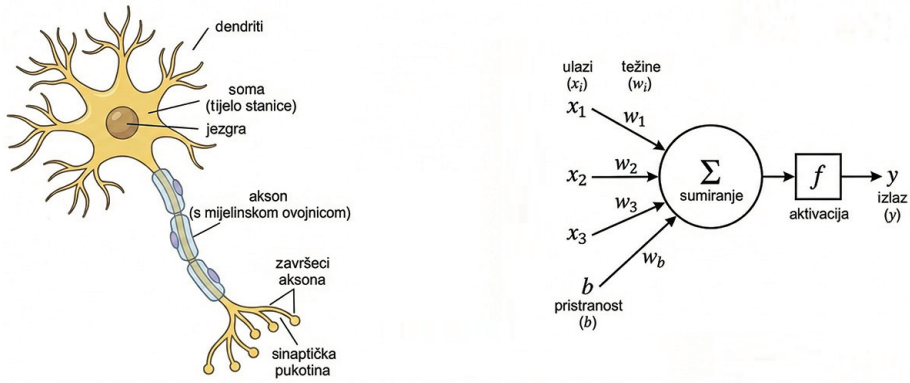
Tekstovi generirani na temelju 3-grama obično nemaju puno smisla. No, izdaleka se mogu činiti kao ispravni tekstovi na engleskom (ili hrvatskom) jeziku. Ovdje je primjer tako generiranog teksta:

> *From what is a day caes the maleuolence of griefe due to disgest his way unequall match and goes hence this I pray god of watchfull tyranny is begun macb and wisely and tell him three to the faire world.*

Kako se n povećava, tekstovi će imati sve više smisla. U slučaju da je n veći od duljine generirane rečenice, cijela će rečenica zasebno imati smisla. U slučaju da je n veći od duljine teksta, i sve će rečenice zajedno imati smisla. Ako smo frekvencije proizveli na temelju istinitih tekstova, ovakav generator rečenica možemo iskoristiti za izradu sveznajućeg *chatbota*: zadajmo bilo koje pitanje kao početak teksta, i pustimo generator da dovrši taj tekst. U tom nastavku teksta pisat će (istinit) odgovor na naše pitanje. Naravno, ovaj pristup predviđanja temeljen na frekvencijama n-grama funkcionira samo uz pretpostavku da su naše aproksimacije $P(x|y)$ savršeno točne, što bi zahtijevalo analizu frekvencija svakog mogućeg istinitog teksta. U nastavku ćemo se baviti metodom generiranja teksta (i drugog sadržaja) koja daje iznimno dobre rezultate, a pritom se ne bazira na nerealističnoj pretpostavci.

Umjetne neuronske mreže

Umjetne neuronske mreže (engl. *artificial neural networks*, ANN) jedan su od pristupa stvaranju modela u umjetnoj inteligenciji, konkretno u strojnom učenju.



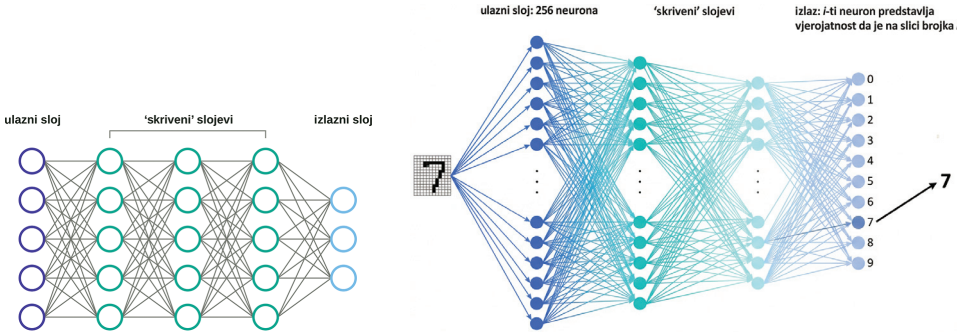
Slika 1. Lijevo: biološki neuron. Desno: tipičan umjetni neuron.

Biološki neuron komunicira sa svojim susjedima kroz dendrite i aksone koristeći elektrokemijske signale. S druge strane, "neuron" u umjetnoj mreži jednostavna je matematička operacija. Neurone u oba slučaja možemo zamišljati kao male kalkulatore, ili logičke sklopove u računalu, koji primaju neke ulazne vrijednosti i proizvode izlaznu vrijednost. Ovdje otprilike i završava analogija između bioloških i umjetnih neurona. Primjerice, algoritam učenja (tj. način odabira broja neurona, njihovih spojeva i funkcija koje pojedini neuroni računaju) bitno se razlikuje. Unatoč razlici, i biološki i umjetni neuroni sposobni

su biti podloga ljudskom razmišljanju odnosno razmišljanju nalik ljudskom razmišljanju. Važno je spomenuti da neuronske mreže nisu zanimljive zbog do sada ispričanih svojstava; sve što smo do sada rekli odnosi se i na logičke sklopove u modernom hardveru.

Umjetne neuronske mreže zanimljive su zbog iznenađujuće iznimne efikasnosti algoritma kojim se stvaraju (*backpropagation*), tj. uče, a koji je efikasan prvenstveno zbog *oblika* funkcije koju svaki umjetni neuron računa. Naime, iako svaki neuron računa vlastitu funkciju, sve te funkcije moraju imati isti oblik: $f(w_1x_1 + w_2x_2 + \ldots + w_nx_n + b)$. Pritom je f obično funkcija $\sigma(x) = \dfrac{e^x}{1+e^x}$ gdje je $e = 2.71828\ldots$, a simboli $x_1, x_2, \ldots, x_n$ predstavljaju ulazne varijable, po jedna varijabla za svaki ulazni neuron. To znači da je jedini stupanj slobode odabir konstanti $w_1, \ldots, w_n$ (*weights*) te konstante b (*bias*).[3] Svaki neuron ima vlastite vrijednosti ovih konstanti.

Izračunamo li $w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$ za neki konkretan neuron i neke konkretne ulazne vrijednosti, dobit ćemo neki realan broj. Vrijednost b ovisi o neuronu i može biti proizvoljno velika ili mala, tako da i čitav izraz može biti proizvoljan realan broj. Taj je broj ulaz za funkciju f. Funkcija f šalje nulu u broj 0.5, negativne brojeve u brojeve bliske nuli, a pozitivne brojeve u brojeve bliske jedinici. Neuronske mreže najčešće se organiziraju u obliku slojeva bez petlji: ulazni sloj prima vanjske informacije (primjerice, kod analize slika, po jedan neuron za svaki piksel na fotografiji). Na izlazu postavljamo jedan ili više neurona. Njihove izlazne vrijednosti (brojevi između 0 i 1) najčešće interpretiramo kao vjerojatnosti. Mreža za prepoznavanje teksta trenirana je na način da sedmi neuron ima visoku vrijednost (blizu 1) ako slika izgleda kao zapis broja 7.



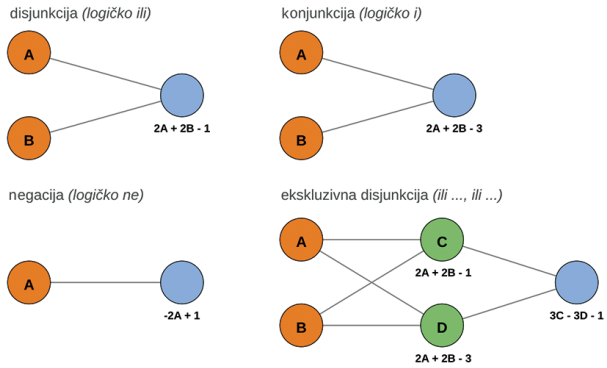
Slika 2. Lijevo: općenita neuronska mreža. Desno: mreža koja prepoznaje rukom napisanu znamenku na slici rezolucije $16 \cdot 16$ *piksela.*

Neuroni kao logički sklopovi

Vidjeli smo kako izgleda funkcija koju računaju umjetni neuroni, no na prvi pogled nije jasno kako iskoristiti takve funkcije za rješavanje stvarnih problema. Mogu li umjetni neuroni reprezentirati dovoljno kompleksne računske operacije?

[3] Često se definira 'ulazna vrijednost' x_0 koja je uvijek jednaka 1, pa umjesto konstante b koristimo w_0. Tada nije potrebno uvoditi posebnu 'bias' vrijednost, jer w_0 ima njenu ulogu. Označimo li $\mathbf{x} = (x_0, x_1, \ldots, x_n)$ te $\mathbf{w} = (w_0, w_1, \ldots, w_n)$, funkciju neurona sad možemo skraćeno zapisati kao $f(\mathbf{w} \cdot \mathbf{x})$. Operacija $\mathbf{w} \cdot \mathbf{x}$ predstavlja skalarni produkt vektora, tj. sumu produkata koordinata s jednakim indeksima.

Poznato je da logičkim sklopovima možemo reprezentirati proizvoljnu operaciju nad fiksnim brojem bitova. Ako pokažemo da neuronske mreže mogu simulirati logičke sklopove, onda je jasno da su neuronske mreže barem toliko izražajne koliko i logički sklopovi.



Slika 3. Neuroni kao logički sklopovi.

Na prethodnoj slici dane su težine kojima možemo simulirati neke tipične logičke sklopove. Primjerice, na prvoj slici težine su $w_1 = 2$, $w_2 = 2$ te $b = -1$. Za ulaznu vrijednost $A = 0$ i $B = 1$, neuron za disjunkciju vraća $f(0 + 2 - 1) = f(1)$, tj. broj veći od 0.5 (istina). Za ulaznu vrijednost $A = 0$ i $B = 1$, neuron za konjunkciju vraća $f(-1)$, tj. broj manji od 0.5 (neistina).

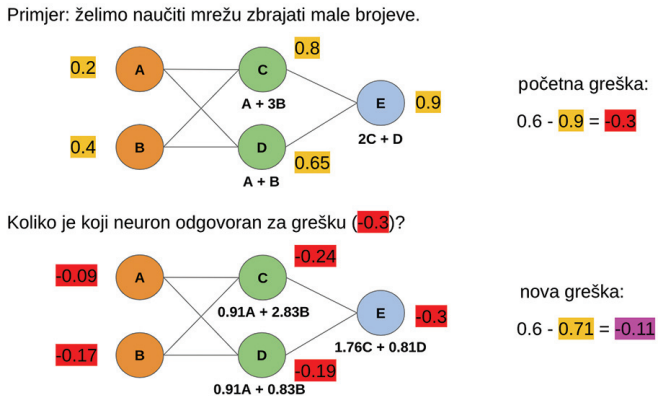
Probajte se sami uvjeriti da jedan neuron ne može biti dovoljan za simulaciju ekskluzivne disjunkcije, tj. pronađite argument da bilo koji odabir brojeva w_1, w_2 i b neće rezultirati izrazom $w_1x_1 + w_2x_2 + b$ koji je pozitivan ako i samo ako $x_1 \neq x_2$ (za $x_1, x_2 \in \{0, 1\}$). Ovaj primjer pokazuje da su skriveni slojevi neuronskih mreža barem ponekad nužni za postizanje potrebne ekspresivnosti.

Dakle, neuronske mreže izražajne su barem koliko i logički sklopovi. Štoviše, one su sposobne neke operacije predstaviti s puno manje operacija (neurona) nego što je to slučaj kod logičkih sklopova. To je posljedica nelinearnosti funkcije f.

Backpropagation

Algoritam **propagacije pogreške unatrag** (engl. *backpropagation*) postao je popularan u drugoj polovici dvadesetog stoljeća, kad je hardver omogućio pokretanje neuronskih mreža razumnom brzinom. Više je ljudi neovisno došlo do sličnih formulacija algoritma, a svima je zajedničko da se temelje na Leibnizovom lančanom pravilu za derivacije. Ovdje nećemo ulaziti u matematičke detalje lančanog pravila niti u sve detalje algoritma, no pokušat ćemo opisati sve glavne korake u algoritmu, te dati pojednostavljen primjer računskog dijela algoritma.

Zamislimo jednostavnu mrežu koja treba naučiti zbrajati brojeve čiji je zbroj jednak broju između 0 i 1. Promatramo taj primjer jer nam je tu dovoljan jedan izlazni neuron čiju izlaznu vrijednost možemo direktno interpretirati kao rezultat zbrajanja, bez drugih transformacija.



Slika 4. Vizualizacija procesa učenja zbrajanja. Zbog jednostavnosti ignoriramo bias, tj. svaki naš neuron određen je samo s neke dvije težine w_1 i w_2.

Proces formiranja mreže započinje odabirom broja slojeva neuronske mreže, te broja neurona u svakom sloju. Za većinu primjena dovoljno je samo nekoliko skrivenih slojeva, a broj neurona po sloju često ne prelazi broj ulaznih i izlaznih neurona. Potom slijedi zanimljiviji dio: odabir brojeva $w_1, w_2, \ldots, w_n$ te broja b za svaki neuron. Na početku možemo sve ove konstante postaviti na slučajne vrijednosti. Proces odabira ovih konstanti sastoji se od jedne velike petlje. U svakoj iteraciji (prolazu) petlje uzimamo jedan istinit testni primjer, primjerice $0.2 + 0.4 = 0.6$, i cilj nam je malo popraviti postojeću mrežu. Ta promjena trebala bi biti takva da neuronska mreža nakon ove iteracije ima malo manju grešku na trenutnom testnom primjeru. Primjerice, možda mreža na početku iteracije tvrdi $0.2 + 0.4 = 0.9$, a nakon ove iteracije tvrdi $0.2 + 0.4 = 0.71$, što je puno bliže točnom rezultatu. Jednom kad smo prošli velik broj primjera i pod pretpostavkom da smo odabrali dovoljno velik broj slojeva i neurona, možemo očekivati da će greška postati zanemariva. Svaka se iteracija sastoji od sljedećih koraka:

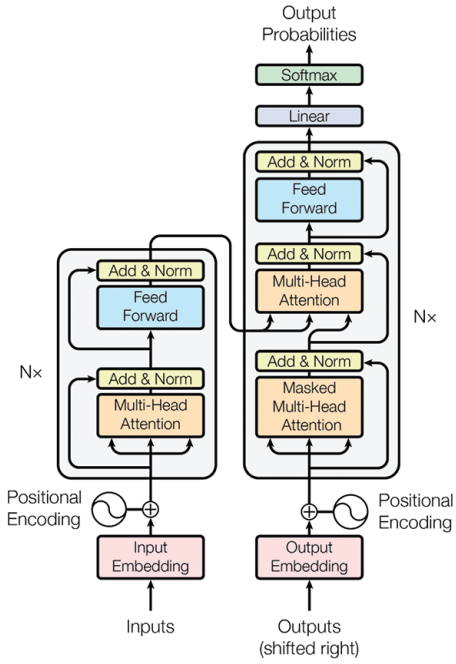
1. **Prolaz unaprijed**: Mreža uzima ulazne podatke i, koristeći trenutne (nasumične) težine, izračunava izlaz. Recimo da je ciljani izlaz 0.6, a mreža je izračunala 0.9. Ovo je situacija prikazana na slici.
2. **Izračun pogreške**: Greška je razlika između dobivenog i očekivanog rezultata: $0.6 - 0.9 = -0.3$.
3. **Prolaz unatrag**: Ovdje se događa "učenje". Kad bismo mogli izravno promijeniti rezultat izlaznog neurona u 0.6 (trenutno je 0.9), naša bi mreža savršeno računala trenutni testni primjer. Ne možemo izravno promijeniti rezultat, ali možemo promijeniti težine koje utječu na rezultat. Od dva neurona C i D spojena na neuron E, onaj čiji je rezultat veći smatramo odgovornijim za grešku: težine kojima množimo njegov rezultat bit će više prilagođene. Što je greška veća, više mijenjamo težine. Ovako bi mogao izgledati pojednostavljen račun. Greška u izlaznom sloju iznosi -0.3, neuron C u prolazu unaprijed imao je vrijednost 0.8, pa težinu w_1 u izlaznom sloju mijenjamo za -0.24: $2 - 0.24 = 1.76$. Neuron D imao je vrijednost 0.65, što je manje od 0.8, pa je manja i promjena: -0.19. Grešku potom provlačimo kroz ostatak mreže (u ovom primjeru preostao je samo još jedan sloj), tj. sada mijenjamo i težine u neuronima C i D, a iznos promjene ovisit će o rezultatima njihovih ulaza A i B.

Opisani algoritam možemo brzo izvršiti. Čak i potpun *backpropagation* algoritam, bez našeg pojednostavljenja, nije bitno kompliciraniji od našeg opisa (zahtijeva samo jednostavne računske operacije za svaki neuron). To je jedan razlog zbog kojeg je opisani algoritam učenja moguće efikasno izvršiti. Drugi bitan razlog jest što ga je lako paralelizirati. No, ono što je ključno nije samo brzina izvršavanja algoritma, već brzina kojom se model prilagođava podacima tijekom izvršavanja algoritma (petlje). Primjerice, najveći današnji modeli imaju velik broj neurona u sebi, koji zajedno imaju do oko 2 bilijuna težina (broj konstanti w_i i b svih neurona zajedno). Pretpostavimo li da te težine mogu biti samo brojevi od 0 do 10, to znači da *backpropagation* izabire jednu mrežu između čak $10^{2\,000\,000\,000\,000}$ mogućih konfiguracija u nekoliko mjeseci učenja (naravno, na bitno snažnijem hardveru od običnih računala). To je iznenađujuće unatoč činjenici da u prostoru mogućih konfiguracija postoji više zadovoljavajućih odabira.

Transformeri

Istraživači iz Googlea 2017. objavili su rad *Attention Is All You Need* [9], predstavljajući **transformer**. Kontekst rada bio je strojno prevođenje prirodnih jezika (primjerice *Google Translate*). Transformer se sastoji od dvije komponente: enkodera i dekodera. Ideja enkodera jest da čita tekst u odabranom jeziku i proizvodi posebno kodiran ulaz za dekoder. Dekoder iz do sada prevedenog dijela teksta i izlaza enkodera generira tekst na drugom jeziku. Obje komponente, enkoder i dekoder, temelje se na neuronskim mrežama.

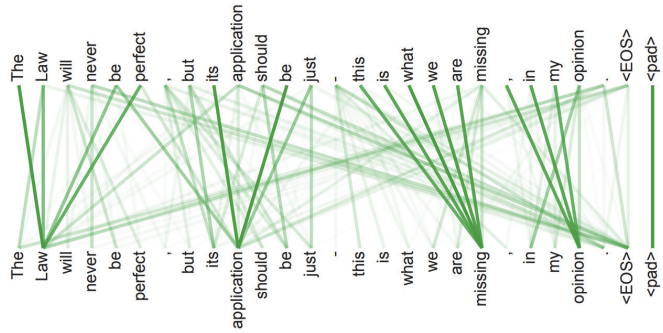
Generativni veliki jezični modeli poput Geminija i GPT-a zapravo pojednostavljuju model transformera. Umjesto da se tekst generira na temelju izlaza enkodera i dosadašnjeg prijevoda, uklanja se enkoder, a dekoder se prepušta sam sebi. Na taj način dekoder stvara tekst 'iz ničega', tj. koristeći samo vlastito znanje o jeziku (a time i znanje o svijetu) i do sada generirani izlaz. Posebno je korisno inicijalizirati "do sada generirani izlaz" unaprijed definiranim tekstom poput nekog pitanja čiji nas odgovor zanima, kako bismo usmjerili rad dekodera u nekom korisnom smjeru.



Slika 5. Struktura transformera. Izvor: [9].

Glavna novost u članku *Attention Is All You Need* u odnosu na neuronske mreže i druge ranije poznate koncepte jest **mehanizam pažnje** (engl. *attention mechanism*). Umjesto sekvencijalnog čitanja teksta, transformer promatra velike podskupove teksta odjednom i za svaku riječ računa koliko je ona važna (koliko 'pažnje' treba obratiti na nju) u kontekstu svake druge riječi. Rezultati obraćanja pažnje koriste se kao ulazi za dublje slojeve neuronske mreže. Važno je spomenuti da niti razlozi obraćanja pažnje, niti kriterij važnosti,

nisu unaprijed određeni. Radije se tijekom učenja modela određeni slojevi modela 'slučajno' specijaliziraju za različite oblike pažnje. Obično se u gotovim modelima mogu pronaći slojevi koji podsjećaju na gramatičku analizu. Primjerice, subjekt rečenice obraća pažnju na objekt, pridjev obraća pažnju na imenicu na koju se odnosi, ili zamjenica obraća pažnju na imenicu koju mijenja. Osim toga, česti su slojevi koji povezuju semantički bliske riječi (primjerice, "kralj" i "car"). Struktura transformera determinira da slojevi pažnje postoje, ali ne i kako izgledaju, tj. kakve će se izvedene informacije proizvoditi u tim slojevima i kako će se te informacije koristiti dalje u modelu.



Slika 6. Mehanizam pažnje: jedan mogući sloj pažnje za dani ulaz. Izvor: [9].

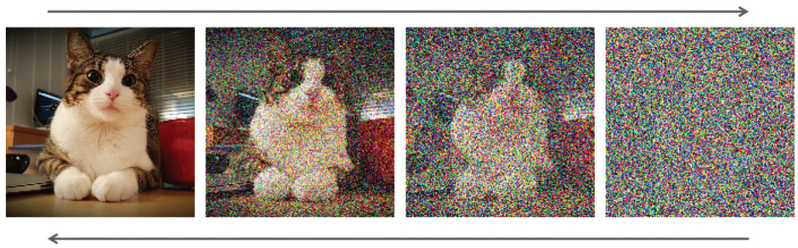
Proces korištenja transformera uključuje i tzv. **tokenizaciju**, tj. razbijanje teksta u manje dijelove koji se nazivaju tokeni te koji se kodiraju kao vektori brojeva. Token je jedinica predviđanja u transformerima. U tekstu smo do sada govorili o predviđanju slova, n-grama i riječi. No, transformeri zapravo predviđaju *tokene*, odnosno kratke nizove simbola koji okvirno odgovaraju slogovima. Dakle, tokeni nisu fiksne duljine poput n-grama, ali ne odgovaraju niti čitavim riječima. Za ovaj okvirni pregled rada transformera ništa se bitno ne mijenja govorimo li o riječima ili tokenima, pa ovo nismo ranije spomenuli. Spomenimo ipak još da je pojedini token u transformeru predstavljen vektorom brojeva. Vektori su odabrani tako da su semantički bliski tokeni (poput tokena "kralj" i "car") bliski u euklidskom smislu. Za optimalan odabir tokena i pridruživanje vektora tokenima također se koriste alati umjetne inteligencije. Transformeri zapravo ni u jednom trenutku ne rade izravno s riječima ili (tekstualnim) tokenima, već samo s vektorskim reprezentacijama tokena. Izlaz iz transformera (predviđeni tokeni) također su vektorske reprezentacije tokena, koje potom možemo prevesti natrag u uobičajen zapis tokena (tekst).

Primjene izvan jezika

Podsjetimo se da su dekoderi preuzeti iz strukture transformera, te su sastavni dio velikih jezičnih modela. No, dekoderi, kao i enkoderi i čitavi transformeri, imaju i druge primjene.

Difuzijski modeli koriste se za generiranje slika (primjerice, Midjourney). Proces učenja obično uključuje postupno *dodavanje* šuma u isprva čistu sliku. Na taj način završavamo s primjerice stotinu slika označenih indeksima od 1 (slika bez šuma) do 100 (šum bez slike). Model potom za svaki indeks uči odrediti što je šum na slici. Indeks je pritom bitan: model će drugačije određivati što je šum na indeksu 80 i na indeksu 20. Kad pokrećemo difuzijski model, krećemo od čistog šuma (indeks 100), i u svakoj iteraciji odbacujemo

dio šuma (neki postotak modelom predviđenog šuma za tu iteraciju), sve dok ne završimo s potpuno čistom slikom.

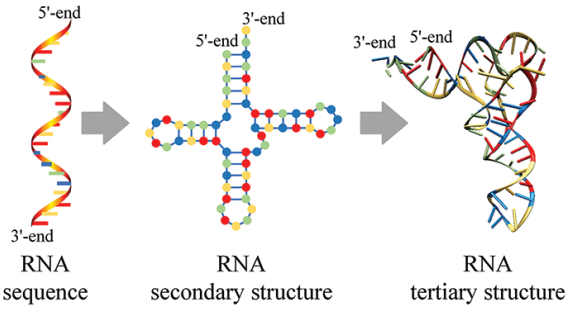


Slika 7. Princip rada difuzijskih modela. Slijeva nadesno: dodavanje šuma u procesu učenja. Zdesna nalijevo: iterativno uklanjanje šuma prilikom korištenja naučenog modela. Izvor: [7].

GameNGen [8] pokazuje kako difuzijski modeli mogu generirati videoigru (Doom) u stvarnom vremenu, predviđajući svaki *frame* odnosno simulirajući *game engine*. Ovi modeli predviđaju sljedeći *frame* koristeći trenutni *frame* i stanje tipkovnice (koje su tipke pritisnute, a koje otpuštene). Na ovaj je način prilično uspješno rekreirana igra *Doom*.

Slika 8. GameNGen: Generiranje igre Doom u stvarnom vremenu. Izvor: [8].

Googleov model **AlphaFold 2** [3] predviđa 3D strukturu proteina iz sekvence aminokiselina s višestruko većom točnošću od dotadašnjih modela (u međuvremenu je izašao i **AlphaFold 3**). Problem određivanja (trodimenzionalne) strukture proteina smatra se iznimno teškim problemom.



Slika 9. Predviđanje strukture proteina/RNA. Izvor: Zhao et al. (2021) [10].

Demis Hassabis i John Michael Jumper, obojica zaposlenici u Googleovom DeepMind laboratoriju, dobili su 2024. godine Nobelovu nagradu za kemiju za svoj doprinos izradi modela AlphaFold 2.

Zaključak

Sadržaju generiranom umjetnom inteligencijom općenito se ne može vjerovati. Zbog toga, idealan problem za umjetnu inteligenciju jest onaj koji se teško *rješava*, a lako *provjerava*. Tada nije potrebna vjera: ako umjetna inteligencija ispravno riješi problem, tu činjenicu možemo lako provjeriti. Ako ne uspije, možemo pokušati sami riješiti problem. U oba slučaja ne postoji rizik oslanjanja na pogrešnu informaciju. Dokazi matematičkih teorema idealan su primjer. U trenutku pisanja ovog teksta, veliki jezični modeli jako zaostaju za ljudskim sposobnostima stvaranja matematičkih dokaza; vjerojatno ne postoji intelektualan zadatak u kojem su veliki jezični modeli (zasad) toliko loši kao što je matematika. To nije iznenađujuće jer je omjer kreativnosti i šablonskog posla puno veći pri stvaranju matematičkih dokaza u odnosu na, primjerice, razvoj softvera.

Veliki jezični modeli međutim mogu se koristiti i za lakši problem: problem formalizacije matematičkih problema u oblik pogodan za automatiziranu provjeru drugim metodama. Ovo je pristup koji zagovara primjerice Marijn Heule ([4]). On predlaže korištenje automatizirane verifikacije temeljene na problemu *SAT* (engl. *satisfiability*, odnosi se na ispunjivost logičkih formula). Više o problemu SAT možete pročitati u članku [6] (Primjer 6.1). Preduvjet ovom postupku jest formalizacija (prevođenje) problema u oblik pogodan za *SAT*. To je dio koji nije zahtijevan koliko i samo dokazivanje, a velikoj je većini matematičara ipak vrlo naporan jer nisu upoznati s potrebnim formalizmima.

Slika 10. Izvor: [1].

Douglas Hofstadter u svojoj je knjizi *Gödel, Escher, Bach* [2] iz 1979. raspravljao o budućnosti simulacije ljudskog razmišljanja. Smatrao je da je za mnoge probleme važan konstrukt *I* ("ja"), jer taj konstrukt povezuje inače odvojene razine stvarnosti u ljudskom razmišljanju. Primjerice, u igri šaha ljudi razmišljaju o formalnim pravilima šaha, o svom iskustvu i strategijama, o fizici pomicanja figurica, o kontekstu igre (poput poretka na ljestvici u slučaju pobjede/ poraza), itd. Taj je konstrukt (*I*) često važan u pronalaženju *out-of-the-box* rješenja problema, gdje je potrebno zaključivati o istom problemu na više razina kako bismo došli do rješenja. Hofstadter je smatrao da je *I* ne samo koristan prečac već i nužan za ljudsku razinu kreativnosti i inteligencije, primjerice za postizanje ljudske razine uspješnosti igranja šaha (u vrijeme pisanja knjige [2] šah se smatrao teškim problemom). Hofstadter je dakle smatrao da bi računalo koje će pobijediti u igri šaha moralo biti računalo koje će imati *I*, odnosno metaforički, računalo koje će moći reći

I'm bored with chess, let's talk about poetry.

kao nuspojavu vlastitih sposobnosti i kreativnosti. Teško je procijeniti je li bio u pravu ili ne, no zanimljiva je situacija opisana u objavi firme Anthropic u nastavku. Radi se o firmi koja je autor jednog od trenutno najsnažnijih velikih jezičnih modela (*Claude*).

Napomena. Zahvaljujem prof. dr. sc. Mladenu Vukoviću na čitanju radnih verzija ovoga teksta i brojnim korisnim sugestijama. Ovaj je tekst nastao na temelju predavanja o umjetnoj inteligenciji održanog 12. prosinca 2024. u XV. gimnaziji u Zagrebu, u sklopu "Dana PMF-a u MIOC-u".

Literatura

[1] Anthropic, *Even while recording these demos, we encountered some amusing moments*, Twitter, 2024., https://twitter.com/anthropicai/status/1848742761278611504

[2] D. R. Hofstadter, *Gödel, Escher, Bach: an Eternal Golden Braid*, Basic Books, 1979.

[3] J. Jumper et al., *Highly accurate protein structure prediction with AlphaFold*, Nature, 596 (7873), 583–589, 2021.

[4] J. Pavlus, *To have machines make math proofs, turn them into a puzzle*, Quanta Magazine, 2025., https://www.quantamagazine.org/to-have-machines-make-math-proofs-turn-them-into-a-puzzle-20251110/

[5] C. E. Shannon, *Prediction and entropy of printed English*, Bell System Technical Journal, 30 (1), 50–64, 1951.

[6] V. Šego, $P = NP$?, Matematičko-fizički list, 70 (Izvanredni broj J), 26–35, 2020., https://hrcak.srce.hr/243630

[7] A. Vahdat & K. Kreis, *Improving diffusion models as an alternative to GANs, part 2*, NVIDIA Technical Blog, 2022., https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-2

[8] D. Valevski et al., *GameNGen: Diffusion Models Are Real-Time Game Engines*, arXiv:2408.14837, 2024.

[9] A. Vaswani et al., *Attention is all you need*, Advances in neural information processing systems, 30, 2017.

[10] Q. Zhao et al., *Review of machine learning methods for RNA secondary structure prediction*, PLOS Computational Biology, 17 (8), 2021.