UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Jovan Marković

FINO PODEŠAVANJE JEZIČKOG MODELA BERT ZA ANALIZU SENTIMENTA

master rad

Beograd, 2025.

Mentor:

prof Aleksandar KARTELJ, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

prof Mladen NIKOLIĆ, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

prof Vladimir FILIPOVIĆ, redovan profesor
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Svim ljudima koje sam upoznao tokom studiranja na fakultetu

Naslov master rada: Fino podešavanje jezičkog modela BERT za analizu sentimenta

Rezime: Ovaj rad demonstrira različite pristupe za fino podešavanje jezičkog modela BERT sa ciljem rešavanja problema analize sentimenta. U okviru rada analiziraju se ključne karakteristike velikih jezičkih modela, njihova struktura i mogućnost prilagođavanja generativne prirode specifičnim zadacima. Posebna pažnja posvećena je upoređivanju metoda potpunog i parcijalnog treniranja modela, kroz primenu parametarski efikasnih tehnika finog podešavanja kao što su pristup zasnovan na treniranju samo poslednjeg klasifikacionog sloja, pristup zasnovan na dodavanju i optimizovanju adaptera, kao i pristup zasnovan na korišćenju matrica niskog ranga. U radu se analiziraju i upoređuju različiti aspekti finog podešavanja, koji uključuju potrebne resurse i dobijene rezultate. Cilj rada je ultimativno shvatanje različitih pristupa jednom istom problemu, uz implicitno predstavljanje daljih pravaca u razvoju finog podešavanja.

Ključne reči: veliki jezički modeli, BERT, analiza sentimenta, fino podešavanje, adapteri, LoRA

Sadržaj

Glava 1

Uvod

Veštačka inteligencija je pojam koji je učestalo krenuo da se koristi u poslednjoj deceniji, najčešće u kontekstu označavanja prelaska u najnoviji period razvoja nauke i načina svakodnevnog života. Ovaj pojam je, sa druge strane,već dugo poznat u svetu računarskih nauka i njegove tehnike razvijane su još od sredine prošlog veka [1]. Iako su dotadašnje primene imale širok spektar različitih problema koji su mogli biti rešavani njima, pažnju svetske javnosti privukao je razvoj jezičkih modela i njihova integracija u svakodnevnom životu. Ovi modeli omogućavaju jednostavnu komunikaciju koja se može odnositi na najrazličitije teme i oblasti. Kada osoba želi da napravi ručak, odradi domaći zadatak, napiše sastav ili pravi plan za teretanu, najčešća putanja vodi do generalizovanih robota koji odgovaraju u sličnom stručnom, prijateljskom ili prilagođenom tonu [2].

Ono što omogućava jezićkim modelima da budu efikasni u svom radu je dostupnost velikog broja informacija u svetu. Jezički modeli svoje znanje crpe trenirajući se na milijardama i milijardama najrazličitijih dostupnih podataka iz oblasti života, literature, nauke i kao takvi omogućavaju komunikaciju na gotovo svaku temu koja je korisniku potrebna. Na internetu postoje milioni informacija o receptima za zdrave obroke, načine popravljanja kućnih aparata ili analizama ljudske psihologije, tako da ne čudi što je moguća komunikacija na temu analize osećanja likova u Šekspirovim delima ili istraživanja različitih mogućnosti nastanka univerzuma [3].

Međutim, primene ovih modela ne završavaju se samo na rešavanju svakodnevnih zadataka. Njih mogu koristiti i ljudi u obavljanju posebnih specifičnih zadataka, kao što je rad u laboratoriji, pisanje koda ili naučnih radova, u kojima oni postaju eksperti i savetnici iz određenih oblasti. Ova primena razlikuje se od uobičajene zbog svoje uske specijalizovanosti ili manjka dostupnih informacija iz konkretnih oblasti.

Takođe, ukoliko želimo da naši modeli ne budu samo mašine u našim očima, već da imitiraju govor pravih ljudi, odgovaraju uvek u nekom specifičnom maniru ili na neki drugi način budu prilagođeni našim potrebama, naophodna je primena posebnih tehnika kojima će se oformljen generalizovan model pretvoriti u specijalizovanog asistenta [4].

Tehnike koje omogućavaju menjanje ponašanja modela za specifične potrebe nazivaju se tehnikama finog podešavanja (eng. fine-tuning). One obuhvataju spektar različitih metoda za menjanje načina funkcionisanja već istreniranih modela. Mogu se ostvariti na različite načine zahtevajući različite nivoe shvatanja funkcionisanja, kao i strukture pojedinačnih modela.

Fine-tuning tehnike značajne su zbog toga što cena pravljenja sopstvenih jezičkih modela može biti velika zahtevajući dragocene resurse [5]. Zbog toga je često praktično izabrati i iskoristiti već postojeće, istrenirane modele i prilagoditi ih sopstvenim potrebama. Na ovaj način zadržavaju se dragocena svojstva odabranih modela, kao što je shvatanje konteksta i mogućnost generalizacije, dok se svojstva modela prilagođavaju specifičnim zadacima.

Popularnost jezičkih modela ogleda se i u njihovoj širokoj dostupnosti na internetu, kao i konstantnim novim modelima koji se mogu koristiti. Razvijaju se i posebne platforme koje omogućavaju preuzimanje modela, kao što je Hugging Face [6], koji korisnicima nude postojeća rešenja na koja mogu nadograditi svoje željene funkcionalnosti.

Zbog svoje kompleksne i memorijski zahtevne strukture, promene na modelima mogu oduzeti dragoceno vreme i potrebne resurse ukoliko ih želimo prilagoditi vlastitim potrebama. Ovo je razlog zašto je potrebno istražiti i uporediti različite metode finog prilagođavanja, kako bismo znali koji metod je potrebno primeniti u zavisnosti od dostupnih resursa i potrebnih performansi.

Moguće je prepoznati različite načine za podelu fine-tuning tehnika, ali ona na kojoj će biti fokus je podela prema broju parametara za treniranje. U ovom slučaju razlikujemo:

1. Potpuno fino podešavanje (eng. full fine-tuning) - Svi parametri datog modela mogu se obučavati i menjati svoja unutrašnja stanja.

2. Parametarski efikasno fino podešavanje (eng. parameter efficient fine-tuning - PEFT)[7] - U ovom slučaju odabira se poskup svih dostupnih parametara modela i na njima su moguće izmene, dok ostali ne menjaju svoja stanja.

U ovom radu biće prikazane sledeće metode finog podešavanja modela:

1. potpuno fino podešavanje (eng. full fine-tuning)
2. fino podešavanje glave za klasifikaciju (eng. classification head fine-tuning) [8]
3. fino podešavanje zasnovano na adapterima (eng. Adapter-based fine-tuning) [9]
4. fino podešavanje zasnovano na matricama niskog ranga (eng. Low Rank Adaptation - LoRA) [10]

Prvi pristup utiče na celokupnu strukturu modela, dok su ostali pristupi vrste pristupa zasnovanog na parametarski efikasnom finom podešavanju, koji za cilj ima brže i manje zahtevno prilagođavanje modela, koje ne utiče na celokupnu strukturu, već samo na odabrani podskup parametara.

Fino podešavanje glave za klasifikaciju predstavlja jednostavan odabir podskupa parametara za treniranje koji je ograničen samo na poslednji sloj parametara modela. Ideja je da se zadrži većina svojstava već istreniranog modela, kako se ne bi radikalno uticalo na kvalitet modela [11]. Ovaj pristup je koristan u slučaju da su komputacioni i vremenski resursi ograničeni jer se trenira samo mali podskup parametara. Takođe je korisno u slučajevima da se trenira na malom skupu podataka jer su promene izraženije [12].

Pristup fine-tuning-a zasnovanog na adapterima podrazumeva umetanje posebnih modula između postojećih slojeva modela. U ovom pristupu zamrzavaju se svi parametri originalnog modela, dok se samo adapteri ostavljaju kao parametri za treniranje. Ovaj pristup donosi sa sobom benefite kao što su:

1. Komputaciona efikasnost: Mi biramo koliko parametara će biti određeno za treniranje, a najčešće predstavlja oko 1-5% ukupnog broja parametara modela.
2. Omogućuje brzu promenu konteksta: Ukoliko postojeće adaptere zamenimo drugim pretreniranim adapterima, omogućuje se brza zamena podešavanja istog modela.
3. Niska cena isporučivanja: Umesto zamene celog modela, dovoljno je isporučiti mali podskup adaptera.

Na ovaj način, adapteri donose mnoge pogodnosti koje su nabrojane uz prilagođavanje samo glave za klasifikaciju, donoseći i benefite u pogledu fleksibilnosti mogučnosti modela.

Nastalo objavljivanjem rada iz 2021. godine, fino prilagođavanje zasnovano na matricama niskog ranga predstavlja jednu od najnovijih metoda fine-tuning-a. U ovom slučaju ubacuju se matrice niskog ranga unutar specifičnih težinskih matrica i tako se samo one podešavaju. Radi po sličnom principu kao i adapteri, tako da sa sobom donosi i sve njihove benefite, ali pažljivo proračunatim izračunavanjima doprinosi još manjem broju parametara za treniranje (manje od 1% od ukupnog broja parametara modela), kao i bržem treniranju.

Iako se tehnike finog podešavanja koje će biti izložene u ovom radu mogu primeniti na sve vrste modela fokus će biti prebačen na Google-ov model BERT, specijalizovan za razumevanje ulaznih podataka, primenjen na poznat problem analize sentimenta [13]. Kao konkretan primer, biće iskorišćen skup recenzija za sajt IMDB [14] koji se često koristi za potrebe testiranja performansi različitih jezičkih modela.

Glava 2

Osnovni koncepti

2.1 Obrada prirodnih jezika

Obrada prirodnih jezika (eng. Natural Language Processing - NLP) je široka oblast koja je počela da se razvija davno pre pojave velikih jezičkih modela, već oko 30-ih godina prošlog veka [15]. U svojoj suštini odnosi se na mašinsko razumevanje teksta napisanog na prirodnom jeziku. Neke od oblasti rada koje ovaj pojam obuhvata su prepoznavanje govora, prevođenje tekstova, sumarizacija teksta itd. Primeri popularnih aplikacija čija suština leži u obradi prirodnih jezika su virtualni asistenti (Alexa, Cortana, Siri...), internet pretraživači, prevodioci, onlajn jezički modeli itd.

Tekst koji se obrađuje može biti struktuiran i nestruktuiran. Primeri nestruktuiranog teksta:

1. „Milica je pošla u školu."
2. „Kupi jaja i mleko."

Primeri struktuiranog teksta:

subjekat: Milica,
objekat: škola,
radnja: poći

ili

<namirnice>
<predmet>jaja</predmet>
<predmet>mleko</predmet>
</namirnice>

Struktuiran tekst je lakši za obradu i razumevanje mašinama, međutim ljudima nije prirodno da komuniciraju na ovaj način. Sa druge strane, nestruktuiran tekst je način na koji ljudi komuniciraju, ali je mašinama teško da shvate njegov kontekst i strukturu.

Proces prevođenja nestruktuiranog u struktuiran tekst naziva se shvatanje prirodnih jezika (eng. Natural Language Understanding - NLU) [16], dok se u suptornom smeru primenjuje generisanje prirodnih jezika (eng. Natural Language Generation) [17].

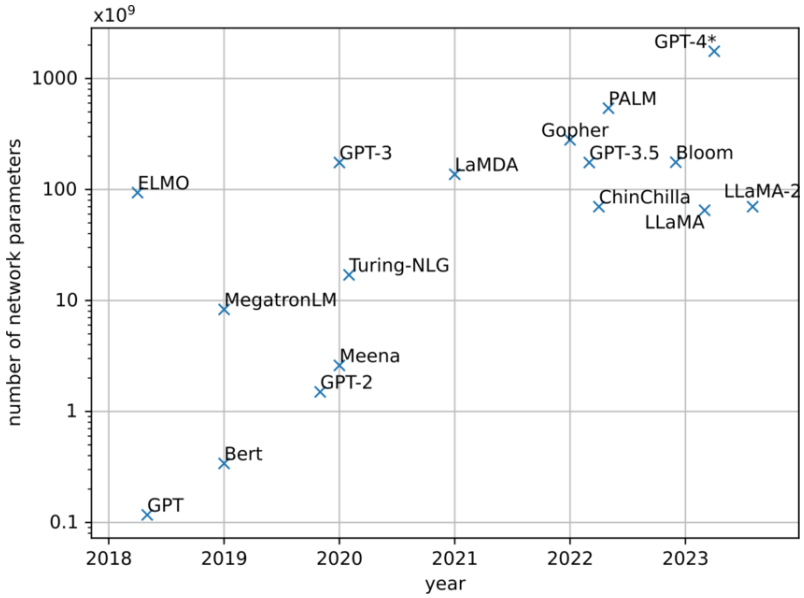
2.2 Veliki Jezički Modeli - LLM

Unutar široke oblasti obrade prirodnih jezika, koja se tradicionalno fokusira na skup osmišljenih pravila, statističke modele i algoritme fokusirane za izvršavanje pojedinačnih zadataka, razvoj velikih jezičkih modela (eng. Large Language Models - LLMs) predstavljaju značajnu prekretnicu. Za razliku od klasičnih metoda obrade jezika, koje poseduju zasebne načine za procesiranje raznorodnih zadataka, kao što su analiza sentimenta, mašinsko prevođenje itd. jedan dobro istreniran veliki jezički model poseduje mogućnost, kroz generalizaciju i prenos znanja za uspešno korišćenje u sve te svrhe.

LLM-ovi su duboke neuronske mreže, istrenirane na ogromnim količinama tekstualnih podataka (ponekad sadrže i po milijardu tokena). Njihova sadržajnost nije izražena samo na količini podataka iskorišćenih za treniranje, već i u količini parametara koje poseduju , krećuću se od nekoliko stotina miliona („GPT-2 Small" model poseduje 117 miliona parametara [18]), sve do preko stotina milijardi (Guglov „PaLM" model sadrži oko 540 milijardi parametara [19]). Slika 2.1 pokazuje kako se sa razvojem novijih modela eksponencijalno povećava broj parametara za njihovo obučavanje [20]. Dalji razvoj modela ide u smeru sve većih i složenijih modela, dok informacije za neke od najnovijih, kao što su GPT-5 [21] i ne otkrivaju njihovu složenost, koja se procenjuje da ima i nekoliko biliona parametara.

2.3 Tokenizatori

Iako se čini da jezički modeli rade sa sirovim tekstovima podataka, ovo nije tačno. Reči predstavljaju niske slova, odnosno simbola, sa kojima je mašinama teže da rade



Slika 2.1: Odnos godine objavljivanja i količine parametara modela, izražen u stotinama milionima. *Broj parametara za GPT-4 nije zvanično objavljen, ali se po procurelim podacima pretpostavlja da je sadrži ok 1760 milijardi parametara.

nego sa brojevima. Zbog toga se unutar modela vrše transformacije rečenica u liste korespodentnih numeričkih vektora kako bi mašinama bio olakšan rad.

Transformacija ulaznih podataka u vektore koji se prosleđuju modelima naziva se tokenizacija [22], a obavlja se pomoću posebnih komponenti koje se nazivaju tokenizatori. Tokenizatori preuzimaju ulazne podatke, vrše njihovu obradu, a na izlaz dostavljaju određen skup tokena. Tokeni mogu biti reči ili delovi reči koji su kodirani na određen način, a zatim su prosleđeni odgovarajućim jezičkim modelima za potrebe treniranja.

Različiti tokenizatori kodiraju skup reči na različite načine. Prilikom početnog treniranja modela nad ulaznim podacima, način rada tokenizatora može biti značajan za krajnje mogućnosti modela [23] [24]. Zbog različitog načina kodiranja, prilikom rada sa nekim unapred istreniranim jezičkim modelima, bitno je primeniti isti tokenizator, jer u suprotnom može doći do značajnog umanjenja moći modela.

Jedan od najpoznatijih načina za transformisanje reči je kodiranje pomoću parova bajtova (eng. Byte Pair Encoding - BPE) [25]. Prvi put je opisan u radu 1994. godine kao jednostavan algoritam kompresovanja podataka [26]. Prepoznajući njegov značaj, 2016. godine on nalazi svoju primenu pri razvoju jezičkih modela, a zbog svoje jednostavnosti i efikasnosti u mnogima od njih koristi se u originalnoj ili izme-

njenoj verziji. Rad ovog algoritma ogleda se u iterativnom pronalaženju učestalih parova bajtova u sekvenci i njihovom objedinjavanju u specijalizovane grupe koje se nazivaju tokenima.

Slika 2.2: Primer rada tokenizatora modela BERTić. Tokeni su označeni različitim bojama, dok je ispod njih njihova brojna vrednost.

Na slici 2.2 nalazi se primer rada tokenizatora za model BERTić koji je model posebno napravljen da razume jezik sa naših prostora i sadrži 8 milijardi različitih tokena [27].

Kako su tokenizatori programi koji prepoznaju obrasce u tekstu i rečima i dodeljuju im jedinstvene vrednosti na osnovu skupa na kojem su trenirani, moguće je da se desi i da za prosleđen simbol ili reč ne postoji odgovarajući token koji se može generisati. U ovom slučaju tokenizatori vraćaju specijalan token koji se naziva nepoznat token (eng. unknown token). Sa svakim neprepoznatim tokenom, gubi se deo informacija, ali ukoliko se ne javlja u velikoj meri, nije potrebno posebno rukovanje u ovim slučajevima.

Pored transformisanja rečenica u niz tokena, tokenizatori takođe imaju i ulogu konvertovanja skupa tokena nazad u rečenice čitljive krajnjim korisnicima.

2.4 Transformeri

Od objavljivanja dokumenta Attention Is All You Need [28], koji je predstavio istraživački tim iz Google-a 2017. godine, došlo je do značajnog pomaka u razvoju jezičkih modela. Dotadašnji pristupi obrade prirodnog jezika zasnivali su se na rekurentnim neuronskim mrežama (RNN) i modelima duge kratkoročne memorije (LSTM) [29]. Ovi pristupi, iako efikasni u hvatanju vremenskih zavisnosti, bili su ograničeni u pogledu paralelizacije i skalabilnosti pri radu sa velikim količinama podataka. Uvođenjem Transformer arhitekture, ovi problemi su u velikoj meri prevaziđeni, a dalji razvoj velikih jezičkih modela (eng. Large Language Models) postao je eksponencijalno brži.

Kako se u dokumentu navodi, Transformeri su sastavljeni od tri vrste gradivnih slojeva koji se nazivaju matrice pažnje (eng. Attention Matrix), neuronske mreže sa propagacijom unapred (eng. Feedforward Neural Network - FNN) kao i normalizacioni sloj (eng. Layer Normalization). Njihovim naizmeničnim kombinovanjem omogućava se efikasno učenje složenih reprezentacija jezičkih sekvenci. Ključna prednost Transformer arhitekture predstavlja istovremeno obraćanje pažnje na različite delove ulazne sekvence, čime se poboljšava razumevanje konteksta i odnosa između reči [30]. Pored toga, zahvaljujući potpunoj eliminaciji rekurentnih i konvolutivnih struktura, Transformer omogućava visoku paralelizaciju procesa učenja, što ga čini posebno pogodnim za treniranje na savremenim hardverskim arhitekturama [31].

Transformeri nisu ostali ograničeni samo na obradu jezika. Njihova primena se proširila i na druge oblasti mašinskog učenja, poput obrade slika, signala i biomedicinskih podataka, gde su takođe pokazali značajna poboljšanja u performansama u poređenju sa ranijim arhitekturama. Upravo ta univerzalnost i fleksibilnost čine transformatore jednom od najvažnijih inovacija u oblasti veštačke inteligencije poslednje decenije [32].

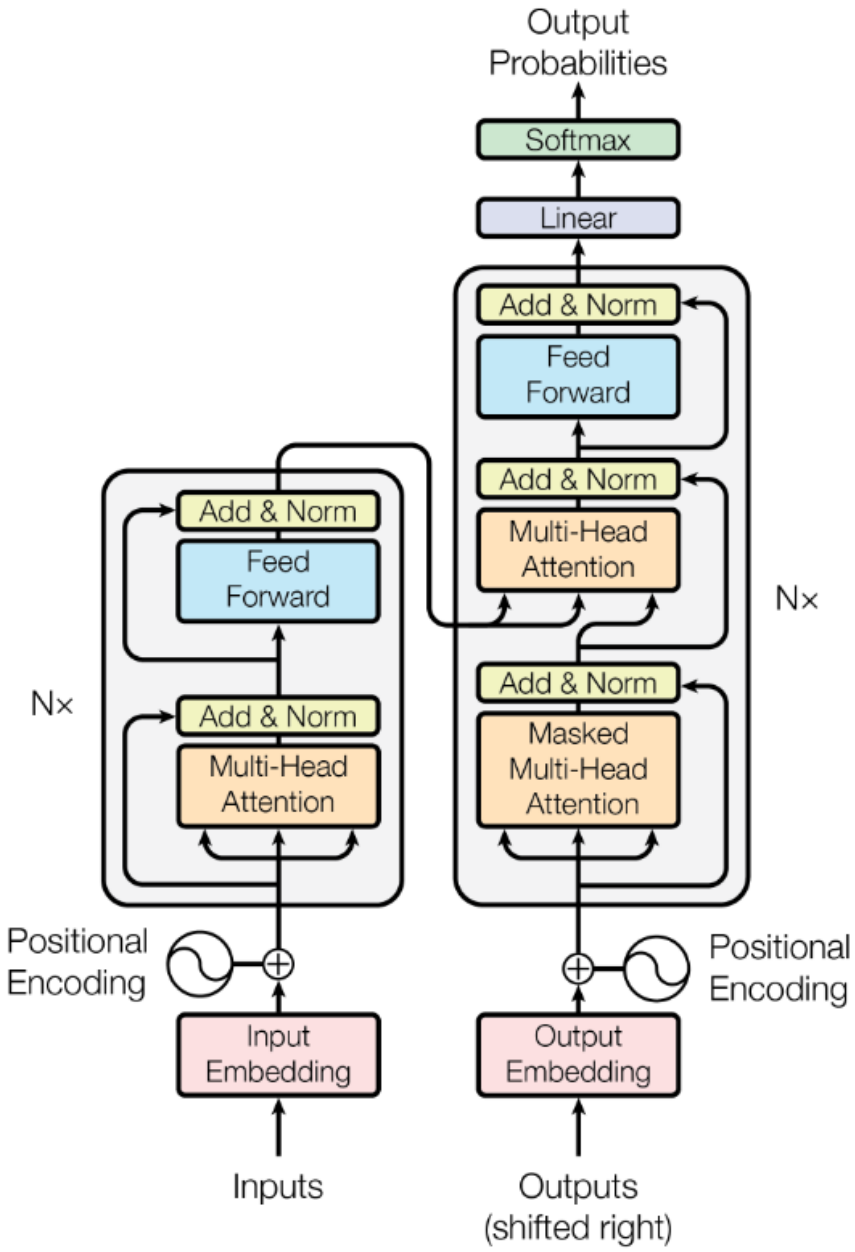
Matrice pažnje

Osnovna uloga ovog sloja je prilagođavanje svake reči određenom kontekstu u kojem se javlja. Iako tokenizator dodeljuje predefinisanu vrednost svakoj reči, ona se dalje transformiše u sloju pažnje u zavisnosti od okruženja u kojem se ta reč nalazi.

Mehanizam pažnje omogućava modelu da za svaku reč identifikuje relevantne informacije iz drugih reči u sekvenci. Intuitivno, to znači da značenje reči ne zavisi samo od njene vrednosti, već i od njenog odnosa prema ostalim rečima u ulaznom tekstu. Veličina dela sekvence koji model može istovremeno da obradi naziva se veličina konteksta (eng. Context size) i ona ima ključan uticaj na performanse modela. Veći kontekst omogućava bogatije reprezentacije, ali istovremeno dovodi do kvadratnog rasta dimenzija matrice pažnje, što značajno povećava računsku složenost.

U zavisnosti od okoline koja se posmatra, razlikujemo:

1. Maskirane jezičke modele (eng. Masked Language Models - MLMs), koji posmatraju tokene i sa leve i sa desne strane u posmatranog tokena. Na ovaj način postižu dobre rezultate u zadacima razumevanja jezika i analize teksta. Modeli koji koriste ovu paradigmu su BERT, RoBERTa, ALBERT.



Slika 2.3: Osnovna struktura transformera predstavljena u originalnom radu (Vaswani i drugi). Model se sastoji od enkoder-dekoder arhitekture u kojoj se svaka komponenta gradi od slojeva pažnje (self-attention) i FFN mreža uz rezidualne konekcije i normalizaciju između njih. Dekoder dodatno uključuje cross-attention prema izlazima enkodera.

2. Kauzalne jezičke modele (eng. Causal Language Models - CLMs), koji posmatraju isključivo tokene koji prethode trenutnom. Ovi modeli specijalizovani su za zadatke generisanja teksta, a primeri su GPT, LLaMA i Phi modeli.

3. Hibridne modele - koji kombinuju ove dve paradigme, gde ulazna komponenta koristi maskirani pristup, dok izlazna generiše tekst na kauzalan način. Ovaj pristup dovodi do boljih performansi u balansiranim zadacima gde su podjednako bitni i razumevanje i generisanje teksta. Predstavnici su T5 i BART.

Mehanizam pažnje definiše se sledećom formulom:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Matrica pažnje (eng. Attention Matrix) omogućava modelu da dinamički proceni značaj svake reči u sekvenci u odnosu na ostale reči u kontekstu. Ulazni vektori Q (Query), K (Key) i V (Value) prolaze kroz linearne transformacije koje određuju koliko svaka reč doprinosi reprezentaciji drugih reči. Skalarni proizvod QK^T meri sličnost između reči u sekvenci. Što je rezultat veći, to je jača međusobna zavisnost između odgovarajućih reči. Radi numeričke stabilnosti, ubacuje se faktor $\sqrt{d_K}$a primenom softmax funkcije dobijaju se verovatnoće koje ponderišu vrednosti vektora V, čime se dobija nova reprezentacija reči koja objedinjuje informacije iz celokupnog konteksta[1] .

Ova transformacija predstavlja samo jedan od mnogo slojeva koji se paralelno računaju kako bi predložili izmenu u vektoru koji određuje svaku reč. Kada se svi oni sumiraju i dodaju na originalan vektor, dobija se konačna nova vrednost tog vektora. Zbog toga se koristi i pojam višeglave pažnje (eng. Multi Head Attention - MHA)

[1]Formulom je predstavljena matrica veličine |veličina konteksta| x |veličina konteksta| gde su po kolonama smeštene vrednosti Q (eng. Query), koje predstavljaju rezultat matričnog množenja posebnih podesivih matrica M_Q(eng. Query Matrix) sa svakom reči u kontekstu, dok su vrednosti K (eng. Key) po redovima nastali množenjem sa drugom matricom M_K (eng. Key Matrix). Vrednosti u matrici pažnje su skalarni proizvodi koji su blizu 1 ukoliko postoji korespodencija između dve reči. Po svakoj koloni se primenjuje softmax funkcija kako bi se dobile verovatnoće koje konfigurišu u tekstu. Promenljiva T predstavlja temperaturu i obično je u opsegu od 1 do 2, $\sqrt{d_K}$ pomaže za potrebe stabilnosti prilikom računanja, a predstavlja koren od broja dimenzija vektora K. Svaki od vektora V dobija se kao proizvod posebne prilagodljive matrice M_V sa vektorima koji predstavljaju reprezentaciju određene reči i u apstraktnom smislu predstavlja vektor koji treba dodati na ostale reči ukoliko želimo da objedinimo njegova svojstva da budu primetna. Nakon primene softmax funkcije, po kolonama se radi još jedan skalarni proizvod sa vektorima V, kako bi se primenio proporcionalni uticaj svake od prethodnih reči. U literaturi, matrica M_V često je prikazana kao proizvod dve matrice, prva koja preslikava vektor u prostor koji odgovara veličini vektora K i Q, i drugu koja to preslikavanje vraća u originalnu dimenziju. Ova transformacija naziva se i transformacija niskog nivoa (eng. low-rank transformation)

koji prestavlja paralelno izvršavanje više matrica pažnje koje na kraju linearnom transformacijom kombinuje rezultat u konačnu matricu izlaznih vektora, gde svaki red predstavlja konačni vektor određenog tokena.

Neuronske mreže sa propagacijom unapred

Neuronske mreže sa propagacijom unapred predstavljaju jednu od osnovnih arhitektura u oblasti mašinskog učenja i obrade prirodnih jezika [33]. Njihova suštinska osobina jeste da se informacije prenose u jednom smeru, od ulaza ka izlazu, bez povratnih veza. U kontekstu transformera, ovaj sloj se može posmatrati kao mesto na kojem se model uči, rezonuje i ubacuje znanje stečeno tokom obuke.

U okviru transformera, feed-forward slojevi primenjuju se nezavisno nad svakim tokenom, što omogućava visoku paralelizaciju procesa. Tokom transformacije tokena, podaci prolaze kroz više faza: najpre linearno preslikavanje u prostor veće dimenzionalnosti, zatim primenu nelinearne aktivacione funkcije (najčešće ReLU), i konačno vraćanje u izvorni dimenzioni prostor. Prebacivanje u višu dimenzionalnost omogućava modelu da raspolaže većim brojem parametara, čime se otvara prostor za složenije reprezentacije i preciznije zaključivanje.

Formalno, ovaj postupak opisan je sledećom jednačinom:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.2}$$

U ovoj formuli W_1 i W_2 označavaju matrice težina, b_1 i b_2 vektore slobodnih članova tih matrica, dok funkcija max() predstavlja ReLU[34] aktivaciona funkcija.

U praksi se, pored ReLU funkcije, koriste i varijacije poput GeLU[35] ili Swish[36] koje omogućavaju stabilnije gradijente i bolje performanse u određenim zadacima. Takođe, dimenzije matrica W_1 i W_2 određuju kapacitet modela - veće dimenzije omogućavaju veću moć rezonovanja, ali po cenu veće računske složenosti i veće mogućnosti preprilagođavanja. Ove karakteristike čine FFN sloj veoma fleksibilnim, omogućavajuću da se njegovo ponašanje adaptira pažljivim izborom parametara i aktivacione funkcije.

Normalizacioni sloj

Normalizacioni sloj je komponenta u transformeru čija je uloga stabilizacija i ubrzavanje procesa treniranja. Oni se postavljaju između slojeva pažnje i neuronskih mreža i primenjuje se po pojedinačnom tokenu normalizujući njegov vektor. Na

ovaj način se ublažava problem nestajućih i eksplodirajućih gradijenata i doprinosi stabilnijem treniranju modela.

U originalnom radu, izlaz ovog sloja se opisuje formulom:

$$LayerNorm(x + Sublayer(x)) \tag{2.3}$$

Ovde Sublayer() predstavlja funkciju koja vrši transformacije nad bilo kojem od transformacionih slojeva (matrica pažnje ili nauronske mreže sa propagacijom unapred).

Ovaj pristup je poznat kao *Post-LayerNorm* i u njemu se proces normalizacije izvršava nakon sabiranja rezultata sa početnim vektorom. Nasuprot ovome, noviji modeli često koriste *Pre-LayerNorm* varijantu u kojoj se normalizacija vrši pre početka obrade podataka.

$$X + Sublayer(LayerNorm(X)) \tag{2.4}$$

Ovaj pristup se pokazao stabilnijim kod veoma dubokih transformera i omogućava bolju propagaciju gradijenata [37].

Svi predloženi slojevi međusobno se smenjuju i u svakoj iteraciji približavaju cilju. Matrice pažnje služe za međusobno shvatanje odnosa između povezanih struktura i time njihovo prilagođavanje kontekstu u kojem se javljaju. Nasuprot tome, prolazak tokena kroz neuronsku mrežu obavlja se samostalno i nezavisno od okoline, gde se primenjuje stečeno znanje modela. Normalizacioni slojevi su postavljeni između njih kao konfiguracioni slojevi koji osiguravaju stabilna i brza izračunavanja. U svakom transformeru ova tri sloja se međusobno smenjuju i ponavljaju nekoliko desetina ili stotina puta, čime se obezbeđuje mogućnost učenja složenih jezičkih i semantičkih obrazaca.

2.5 BERT model

Model BERT[38] nastao je 2018. godine od strane stručnjaka iz Guglovog tima. Motivacija za razvoj ovog modela bilo je zapažanje da su dotadašnji modeli koristili isključivo čitanje konteksta sa leve ili sa desne strane. Ovaj model, čije puno ime stavlja akcenat upravo na njegovu karakteristiku da pri analizi rečenica koristi kon-

tekst i sa leve i sa desne strane (eng BERT - Bidirectional Encoder Representations from Transformer), pomerio je granice mogućnosti rezonovanja modela.

Unidirekcioni modeli (oni koji koriste samo kontekst sa leve ili sa desne strane svake reči) pokazuju se dobro na zadacima generisanja teksta, međutim često se javljaju sub-optimalna rešenja usled toga što se ne uzima u obzir kontekst iz cele okoline, već samo sa jedne strane teksta. Sa druge strane, za potrebe razumevanja celokupnog konteksta, bidirekcioni modeli se pokazuju kao bolji izbor. U ove probleme spadaju klasifikacija teksta, odgovaranje na pitanja ili analiza sentimenta.

Bidirekciona priroda BERT modela omogućava jedinstven način treniranja modela, optimizovanjem MLM (Masked Language Model) funkcije cilja. Ovo se postiže tako što MLM nasumično maskira tokene sa ulaza, a cilj je da se pomoću konteksta uspešno izvrši predikcija maskiranih tokena. Istovremenim spajanjem konteksta sa obe strane, ovaj model naziva se i dubokim bidirekcionim modelom, nasuprot plitkih bidirekcionih modela, nalik na ELMo[39] model koji koristi konkatenaciju dva odvojena unidirekciona sloja.

Drugi zadatak na kojem se BERT model trenirao je predviđanje sledeće rečenice (eng. Next Sentence Prediction - NSP). Ovo je zadatak binarne klasifikacije u kojoj model uči da prepozna da li druga rečenica logički sledi nakon prve. Ovaj zadatak od suštinskog je značaja za razumevanje odnosa među rečenicama u širem tekstu.

Što se tiče unutrašnje strukture, osnovni BERT model sastoji se od 12 slojeva transformera (svaki uključuje mehanizam višeglave pažnje, feed-forward mrežu i normalizaciju), pri čemu se svaki token reprezentuje kao vektor dimenzije 768. Vokabular modela obuhvata oko 30 000 tokena, a ukupno broj parametara iznosi približno 110 miliona[2].

Model je treniran na ogromnim tekstualnim korpusima sa ciljem da obuhvati što širi spektar jezičkih stilova i domena. Konkretno, korišćen je BookCorpus skup podataka koji sadrži oko 800 miliona reči iz više od 11 000 knjiga različitih žanrova, kao i engleska verzija Vikipedije koja obuhvata oko 2.5 milijarde reči iz više od 2.5 miliona članaka.

[2]Od ukupno 110 miliona parametara u modelu, oko 23 miliona koristi se u embedding slojevima, dok MHA i FFN koriste respektivno približno 2.3 i 4.7 miliona parametara po svakom od 12 slojeva. Ovde posttoje i positional embedding slojevi, kao i slojevi normalizacije, ali njihov doprinos je zanemarljiv u ovoj računici

2.6 Problem analize sentimenta

Problem analize sentimenta (eng. Sentiment analysis) predstavlja zadatak klasifikacije teksta u kojima je cilj odrediti da li odražava pozitivno ili negativno mišljenje o nekoj temi. Zbog svoje jednostavnosti za shvatanje, velike količine skupova podataka koja se može pronaći iz različitih oblasti, kao i različitih mogućnosti za generisanje cilja, ovaj problem predstavlja jedan od najosnovnijih koji se koriste u svrhu testiranja rezonovanja jezičkih modela. Skupovi tekstova mogu poticati iz različitih izvora, kao što su internet forumi, ocene proizvoda poručenih preko interneta, komentari na društvenim mrežava, ocene filmova... Kao cilj klasifikacije česte su podele na dve (pozitivno ili negativno) ili tri (pozitivno, negativno ili neutralno) klase, a moguće je i na izlazu generisati brojčanu vrednost (na primer ocena od 1 do 10, gde 1 predstavlja najlošiju ocenu, a 10 najbolju).

Modeli specijalizovani za rešavanje problema analize sentimenta svoju primenu mogu takođe naći na različitim mestima. Značajni mogu biti u sistemima preporuke gde se koriste informacije o utiscima proizvoda, ali takođe i na društvenim mrežama gde se prati koliku reputaciju imaju određeni brendovi ili organizacije. Omogućavaju praćenje uspešnosti filmova/serija, ali i političkih kampanja ili događaja u svetu. Takođe bitna stavka je i korišćenje za potrebe ispitivanja finansijskih tržišta, gde je bitno prepoznati mogući uspeh nekog proizvoda u ranoj dobi. Pored nabrojanih, postoje i razne druge oblasti [40].

Zbog svoje raznovrsnosti, kako granulacije na kojoj se analiza može vršiti (analiza dokumenata, pasusa, rečenica, komentara...)[41], tako i kompleksnosti samih tekstova (analiza emoji-ja, lokalizama, formalan ili neformalan govor, reči u kojima su sva slova velika, sarkazma u tekstu...) u ovim problemima mogu se javljati razni izazovi koje modeli treba uspešno da prevaziđu [42].

Analiza sentimenta jedna je od starijih problema u oblasti klasifikacije, i za njeno rešavanje mogu se koristiti mnoge metode, kao što su klasične metode nalik na Naivni Bajes[43] ili sistem potpornih vektora[44], metode dubokog učenja nalik na konvolutivne[45] ili rekurentne neuronske mreže[46], LSTM (Long-short term memory)[47]... a moguće je i koristiti jezičke modele zasnovane na transformerima, kao što su BERT, DistilBERT, RoBERTa...

Trenutno je dostupan veliki broj različitih skupova za probleme analize sentimenta, a neki od najpoznatijih su SST (Stanford Sentiment Treebank)[48], Amazon products review [49], Sentiment140 [50], IMDB Large Movie Review Dataset [14] i

drugi.

Large Movie Review Dataset

Large Movie Review Dataset je skup podataka nastao sastavljen od recenzija filmova sa sajta IMDB (Internet Movie Database). Nastao je 2011. godine, a kreirao ga je Andrew L. Maas [14]. On sadrži ukupno 50 000 recenzija, ravnomerno podeljenih na pozitivne i negativne, što ga čini pogodnim za zadatke binarne klasifikacije sentimenta.

Svaka recenzija je označena kao pozitivna ili negativna na osnovu ocene filma dodeljene od strane korisnika. Recenzije sa ocenom 7 ili više smatraju se pozitivnim, dok se recenzije sa ocenom 4 ili manje smatraju negativnim. Recenzije sa osrednjim ocenama (5 i 6) su izostavljene kako bi klasifikacija bila jasna i binarna.

Ovako transformisane, recenzije su ravnomerno organizovane u trening i test skupove bez preklapanja, kako bi se smanjila pristrasnost prema nekoj od klasa, dok veličina skupa omogućava treniranje savremenih jezičkih modela.

Glava 3

Fino podešavanje velikih jezičkih modela

Fino podešavanje nastaje prilagođavanjem pre-treniranog modela kao osnove na manjem domenski-određenom skupu. Ono što razlikuje ovu metodu od klasičnog treniranja modela, jeste to što se kao osnova koristi model koji već ima opsežno znanje i percepciju lingvističkih pojmova i njihovih odnosa. Zbog toga, fino podešavanje predstavlja dodatni korak koji može značajno doprineti rezultatima izvršavanja modela u oblastima za koje je dodatno treniran.

3.1 Značaj i uloga finog podešavanja

Fino podešavanje nadograđuje postojeće znanje modela, prilagođava ga oblasti u kojoj se očekuje njegovo korišćenje, poboljšavajući performanse dok istovremeno koristi manje kompjuterskih resursa ili podataka pri njihovoj obradi. Zbog svega ovoga, primena finog podešavanja postala je popularna u NLP zadacima, pogotovo u oblasti klasifikacije teksta, analize sentimenta i generisanja odgovora.

Pogodnosti korišćenja finog podešavanja, ukoliko je ono moguće, su brojne, a neke od njih su date u nastavku:

- **Prenos znanja (eng. Transfer Learning)** Fino podešavanje omogućava modelu da zadrži svoje stečeno generalno znanje i dalje ga prilagodi specifičnim zadacima. Ovaj proces značajno smanjuje vreme i resurse u odnosu na treniranje modela od starta, kako model već poseduje snažne osnove u shvatanju prirodnih jezika [51].

- **Smanjena količina potrebnih podataka** Kako pre-trenirani modeli već poseduju izvesnu količinu znanja, fino podešavanje je usmereno većinom ka shvatanju suptilnijih veza između relevantnih pojmova u definisanom domenu, nije potrebna velika količina podataka za njegovo prilagođavanje. Ovo je naročito korisno u specijalizovanim oblastima gde ne postoje veliki označeni skupovi ili u primeni na jezike za koje ne postoje ekstenzivni resursi[52].

- **Poboljšana generalizacija** Kako fino podešavanje obično obohvata prilagođavanje samo pojedinih elemenata modela, umesto svih elemenata što je slučaj kod inicijalnog treniranja, na ovaj način može se zadržati visok stepen generalizacije početnog modela, smanjujući mogućnost za preprilagođavanjem modela [53].

- **Efikasna isporuka modela** Fino podešeni modeli računski su efikasniji za primenu u aplikacijama u stvarnom svetu, kako ne zahtevaju pre-treniranje u punom obimu. Ovo ih čini podesnim za isporuku u produkcionim okruženjima gde resursi mogu biti ograničeni. Dodatno, fino podešavanje omogućava prilagođavanje modela bez potrebe za potpuno novom arhitekturom.

- **Prilagođavanje na različite zadatke** Fino podešeni modeli omogućavaju jednom pre-treniranom modelu da se prilagodi na širok spektar NLP zadataka. Umesto da postoji poseban model za svaki zadatak, fino podešenim modeli može se promeniti svrha bez ekstenzivnih modifikacija.

- **Performanse u domenskim zadacima** Fino podešavanje omogućava modelima visoke performanse u specijalizovanim oblastima, nalik na biomedicinu, pravo ili finansije, kao i na nove i nedovoljno dokumentovane oblasti. Ukoliko pre-treniran model posveti pažnju domenski specifičnim podacima, on može naučiti posebne žargone i pojmove i njihove odnose i kasnije ih primeniti u praksi[54].

- **Brža konvergencija** Kako je pre-treniran model već prilagođen generalnom shvatanju jezika, dodatne izmene konvergiraju brže nego kod modela koji sve moraju da preračunaju ispočetka. Ovo rezultuje manjem vremenu treniranja kao i manjim računskim zahtevima, što omogućava brže eksperimentisanje sa različitim tehnikama finog podešavanja i čini ga preferiranim metodom za mnoge primene NLP-a u stvarnom svetu.

3.2 Vrste finog podešavanja

Kako fino podešavanje predstavlja opšti način za usmeravanje modela ka specifičnim oblastima za koje će biti korišćen, njegovo izvršavanje može se ostvariti na različite načine. U opštem smislu, metode korišćene u svrhu finog podešavanje mogu se implementirati imajući u vidu 3 različita pristupa:

- **Po podacima koji se koriste**
- **Po broju datih primera**
- **Po strategiji odabira parametara koji se modifikuju**

1. Po podacima koji se koriste

U ovoj podeli vrši se fino podešavanje modela kroz različite pružene podatke a najčešće zavisi od problema koji se rešavaju.

- **Nadgledano fino podešavanje** – Model je treniran nad labelisanim skupom, kako bi dodatno prilagodio svoje parametre. Korisno za rešavanje problema klasifikacije ili mašinske prevođenja teksta.
- **Nenadgledano fino podešavanje** – Model se trenira na neoznačenim podacima u svrhu daljeg shvatanja veza između elemenata teksta. Korisno ukoliko model želimo da upoznamo sa usko specifičnim oblastima ili oblastima za koje nema puno podataka.
- **Fino podešavanje kroz prompt instrukcije** – Umesto modifikacije parametara modela, modelu se nagoveštava način na koji treba da generiše odgovor. Korisno ukoliko nam je struktura ili veličina generisanog odgovora bitna.

2. Po broju datih primera

U ovom slučaju kao značajna komponenta figuriše količina pruženih podataka. Ukoliko imamo ograničen skup podataka, i treniranje na svega nekoliko pruženih instanci može poboljšati performanse modela.

1. **Učenje bez primera (eng. Zero-shot learning)** – U ovom slučaju model se izvršava bez ikakvog predznanja o specifičnostima za zadatu oblast, oslanjajući se na svoje pre-trenirano znanje.

2. **Učenje sa malo primera (eng. Few-shot learning)** – U ovom slučaju modelu ja pružen mali skup označenih podataka (često svega 2-3 primera), najčešće u svrhu uspešne generalizacije problema.

3. **Puno fino podešavanje** – Model se trenira na velikom skupu kako bi se potpuno prilagodio postavljenim zadacima.

3. Po strategiji odabira parametara koji se modifikuju

U zavisnosti od odabrane strategije, kao i količine parametara kojima je omogućeno treniranje može se balansirati između prilagodljivosti modela specifičnim zadacima kao i zadržavanju generalizovanog znanja koje model poseduje. Ovo je oblast finog podešavanja koja poseduje najveće mogućnosti i mnogo različitih načina za ostvarivanje. Zbog toga će se u ovom radu najviše pažnje posvetiti upravo ovim metodama finog podešavanja i ispitati njihov uticaj na model.

- **Potpuno fino podešavanje (eng. Full Fine-Tuning)** – Svi parametri modela označavaju se kao mogući za treniranje. Ovaj pristup zahteva najviše računskog vremena kako utiče na celokupan model.

- **Parametarski efikasno fino podešavanje (eng. PEFT - Parameter-Efficient Fine-Tuning)** – Širok spektar različitih strategija za fino podešavanje, u opšem smislu odnose se na zamrzavanje velikog broja parametara modela, dok se za treniranje koristi manji broj postojećih ili dodatih parametara kako bi se ubrzao proces treniranja i prilagodio model bez gubitka moći generalizacije. U ovu kategoriju potpadaju mnogi pristupi, od kojih su najpoznatiji:

  - **Fino podešavanje zasnovano na adapterima (eng. Adapter-based fine-tuning)** – Modelu se svi parametri označavaju kao netrenirajući, a ubacuje se dodatni sloj adaptera koji se prilagođava. Na ovaj način smanjuju se računski zahtevi, a dosta prostora se ostavlja eksperimentisanju sa slojevima adaptera i njihovog uticaja na specifične probleme.
  - **Fino podešavanje zasnovano na matricama niskog ranga (eng. Low-Rank Adaptation)** – Popularan efikasan metod koji smanjuje korišćenje memorije trenirajući podskup težinskih matrica.

Neki pristupi finom podešavanju kombinuju različite nabrojane tehnike kako bi maksimalno iskoristili mogućnosti svojih modela. Odabir korišćene tehnike pre svega zavisi od problema koji se rešava, dostupnih kompjuterskih resursa i količine dostupnih podataka, kao i poznavanja unutrašnje strukture različitih modela.

3.3 Potpuno fino podešavanje

Potpuno fino podešavanje (eng. Full Fine Tuning) predstavlja najklasičniji i najdirektniji pristup prilagođavanju velikih jezičkih modela specifičnom zadatku. Suština ove metode je da se svi parametri modela treniraju ponovo na ciljanom skupu podataka, vršeći potpunu optimizaciju težina mreže, nasuprot dodavanju ili modifikaciji specifičnih slojeva. Na ovaj način, model se u potpunosti prilagođava karakteristikama novog zadatka, jer svaka njegova sastavna komponenta prolazi kroz dodatni proces učenja[55].

Jedna od najvećih prednosti full fine-tuninga je ostvarivanje visokih performansi. Pošto se svi parametri optimizuju, model ima potencijal da postigne vrhunske rezultate u ciljanom zadatku, naročito kada ima dovoljno veliki i reprezentativan skup podataka na raspolaganju. Ukoliko je potrebno da se model prilagodi specifičnoj oblasti rada i terminologiji, kao što je medicinska ili pravna dokumentacija, gde je takođe dostupan veliki broj informacija, ovo je često preporučen pristup [56].

Glavni izazov kod ovog pristupa je računska složenost. Prilagođavanje milijardi parametara zahteva ogromne resurse, kako u smislu GPU memorije, tako i u vremenu treniranja. Pored toga, koristi se i značajna memorija, jer svaka verzija prilagođenog modela zauzima jednak prostor kao i originalni model. Zbog sve većeg broj parametara koje savremeni modeli poseduju, ovaj pristup postaje sve teži za praktično ostvarivanje i često nije održiv za modele koji imaju preko nekoliko milijardi parametara.

Drugi značajni problemi prilikom potpunog finog podešavanja predstavljaju mogućnost overfittinga i nestabilnog treniranja. Kada se trenira ceo model, ukoliko je količina podataka u ciljnim domenima mala, model može da nauči napamet trening skup umesto da generalizuje znanje. Sa druge strane, količina i vrsta podataka koja se pruža modelu takođe može značajno doprineti različitim performansama modela. Jedna studija istraživača sa Saarland i EPFL univerziteta ukazala je da prilikom full fine-tuninga ponašanje modela može značajno varirati, čak i pri istim hiperparametrima zbog problema u optimizaciji (npr. nestajući gradijenti) i nepredvidivih

razlika u generalizaciji iz istih gubitaka. Ovo znači da čak i redosled i izbor primera mogu direktno uticati na stabilnost i konačne performanse modela [57].

Obzirom na to da se prilikom treniranja na ovakav način menja celokupna struktura modela, posebno je potrebno obratiti pažnju na parametar stope učenja koji se koristi. Jedan od najkarakterističnijih problema koji se mogu javiti u procesu potpunog prilagođavanja je katastrofalno zaboravljanje. Ovo je pojava u kojoj inicijalne težine modela gube svoja mogućnost generalizacije. Da bi se ovo sprečilo, za stopu učenja se postavlja vrednost daleko manja od one korišćene u inicijalnom procesu treniranja modela [58].

U poređenju sa alternativnim metodama, full fine-tuning se posmatra kao zlatni standard kada resursi nisu ograničeni. Iako je njegova praktična primena ograničena na istraživačke institucije ili kompanije sa snažnim hardverskim infrastrukturama i obimnom količinom podataka, ovaj pristup često se koristi kao referentna tačka u odnosu na nove metode koje teže da postignu slične rezultate uz značajno manju potrošnju resursa.

3.4 Fino podešavanje klasifikacione glave

Nasuprot potpunom finom podešavanju, fino podešavanje klasifikacione glame zamrzava veliku većinu slojeva ostavljajući samo mali podskup parametara koji se nalaze u poslednjem sloju podložnim za treniranje. Ideja iza ovog pristupa je da se očuvaju bogate postojeće reprezentacije jezika koje model već poseduje [38].

Ovaj pristup rešava većinu glavnih problema potpunog finog podešavanja, jer drastično smanjenje broja parametara za optimizaciju vodi značajnim uštedama. Ubrzava proces treniranja i smanjuje potrošnja energije. Potrošnja memorije za pravljenje različitih verzija modela primetno je manja jer je samo potrebno preneti modifikovan poslednji sloj modela. Pošto se jezgro modela ne menja, značajno se i smanjuje rizik od problema katastrofalnog zaboravljanja. Prednost se ogleda i u stabilnosti treniranja, jer je treniranje malih slojeva numerički stabilnije i manje podložno problemima optimizacije[57]. Potreban broj primera da bi se izbeglo preprilagođavanje takođe je manji zbog malog skupa parametara koji se menjaju.

Glavna ograničenja fine-tuninga klasifikacione glave je njegova ograničena primena. Ukoliko je potrebno sveobuhvatno prilagođavanje specifičnom zadatku, čiji domen podataka značajno odstupa od onoga na čemu je model treniran, performanse mogu biti suboptimalne. Odavde se zaključuje da je u ovom pristupu značajan

i izbor modela nad kojim će se prilagođavanje ostvarivati, jer osnovno shvatanje konteksta ostaje nepromenjeno.

U praksi ovaj pristup se često koristi za klasifikacione zadatke poput analize sentimenta, kategorizacije teksta ili detekcije namere. Takođe je koristan u scenarijima kada su skup za treniranje, računski, vremenski ili memorijski resursi ograničeni. U ovakvim specifičnim situacijama, izborom ove tehnike prilagođavanja postiže solidan nivo performansi [59].

3.5 Fino podešavanje zasnovano na adapterima

Adapteri predstavljaju jednu od najznačajnijih tehnika parametarski efikasnog prilagođavanja jezičkih modela. Oni uvode male dodatne neuralne module - adapterske slojeve (eng. Adapter layers) unutar već postojećih (npr. između feed-forward slojeva u transformeru), pri čemu se samo oni treniraju, dok originalna mreža ostaje nepromenjena. Ovime se ostvaruje kompromis između potpunog fine-tuninga i prilagođavanja samo klasifikacione glave [9].

Tipična arhitektura adaptera uključuje takozvanu „usko grlo" arhitekturu (eng. bottleneck architecture), pri kojoj se ulazni vektor projektuje u prostor manje dimenzionalnosti, prolazi kroz nelinearnu transformaciju i vraća nazad u originalan prostor. Formula koja ovo opisuje je:

$$f_{adapter}(x) = W_{up}g(W_{down}x + b_{down}) + b_{up}, \tag{3.1}$$

gde su

$W_{down} \in R^{r \times d}$ - matrica koja preslikava originalnu dimenzionalnost u manju (r « d), praveći „usko grlo"

$W_{up} \in R^{d \times r}$ - matrica koja vraća vektor iz niže dimenzionalnost u originalan prostor

$g(\dot{})$ - nelinearna funkcija (najčešće ReLU)

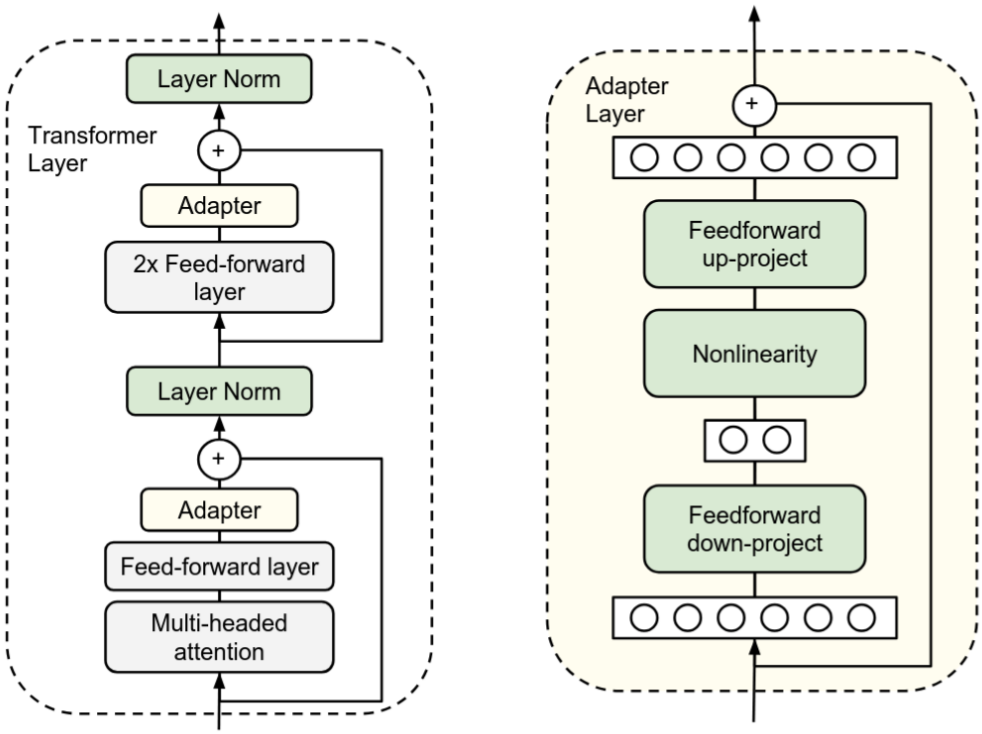
b_{down} - vektor pomeraja (bias) unutrašnje dimenzije r

b_{up} - vektor pomeraja originalne dimenzije d

Ovako definisan modul ubacuje se u model uz rezidualnu konekciju

$$h_{out} = x + f_{adapter}x \tag{3.2}$$

Ova redukcija smanjuje broj parametara koje je potrebno trenirati na samo $W_{up}, W_{down}, b_{up}ib_{down}$, što predstavlja $O(d\dot{r})$ parametara za treniranje, omogućavajući efikasno učenje specifičnih karakteristika zadatka. Primer arhitekture dat je na slici 3.1, a preuzet je iz rada Parameter-Efficient Transfer Learning for NLP[9].



Slika 3.1: Primer arhitekture zasnovanog na adapterima. Feedforward up/down-project predstavljaju funkcije koje vektor preslikavaju u prostor više/niže dimenzionalnosti.

Implementacija adaptera u modelima najčešće je ostvarena kroz ubacivanje modula u dve tačke unutar svakog BERT transformer sloja: nakon self-attention izlaza i nakon feedforward izlaza. Ovo predstavlja implementaciju originalne Houlsby [9] arhitekture. Alternativni pristup je predložen od strane Pfeiffer-a [60] koji sugeriše dodavanje adaptera samo nakon attention sloja, kako bi se napravila dodatna ušteda parametara za treniranje, dok su eksperimenti pokazali da se pri tome ne gubi značajno na preciznosti.

Osnovna prednost ovog pristupa je parametarska i memorijska efikasnost. Kod adaptera dovoljno je čuvanje samo parametara dodatnih slojeva, čime se troškovi skladištenja i isporučivanja drastično smanjuju. Takođe, dodavanje malih modula,

specifičnih za određene zadatke čine ih pogodnim za scenarije sa više zadataka ili domena, čineći ih fleksibilnijim u odnosu na samo prilagođavanje klasifikacione glave.

Modularnost adaptera posebno ih čini podobnim za korišćenje za rešavanje različitih raznorodnih familija problema, pri čemu je dovoljno zameniti adaptere specijalizovane za svaki zadatak. Ovime se omogućava ponovno korišćenje istog osnovnog modela, nasuprot pravljenju i dostavljanju pojedinačnih za svaki zadatak [61].

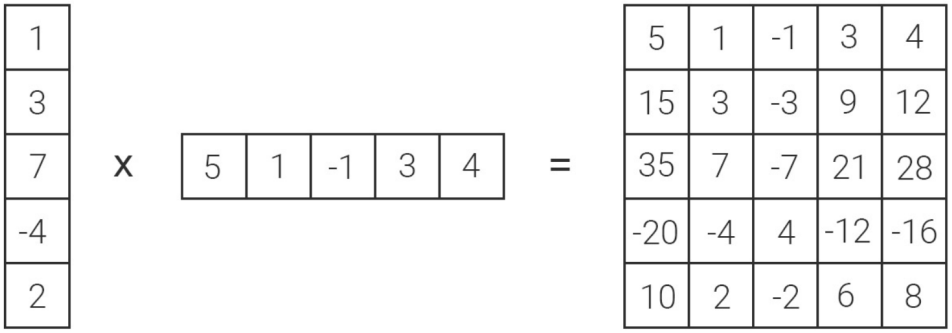
Kao i kod prilagođavanja samo klasifikacione glave, i ovde na kvalitet utiče sam osnovni model koji se koristi. Sposobnost adaptera da se prilagode zadacima imaju svoja ograničenja i mogu zaostajati za full fine-tuningom kada je reč o prilagođavanju za ekstremno specifične zadatke. Takođe, sama struktura adaptera i njihovo pozicioniranje u osnovnom modelu može predstavljati iscrpan proces u kojem rezultati mogu značajno varirati [62].

Primena adaptera naročito dolazi do izražaja u industrijskim aplikacijama gde je potrebno održavati veliki broj modela za različite klijente ili jezike. Ovaj pristup u tim situacijama vodi velikom smanjenju troškova održavanja, nasuprot održavanju velikog broja različitih modela. Kao relativno nov pristup učenju modela, ostavlja prostora za dodatna istraživanja u cilju daljih poboljšanja performansi modela [60].

3.6 Fino podešavanje zasnovano na matricama niskog ranga

Nastalo radom objavljenom 2021. godine od strane Edward Hu i Yelong Shen-a, fino podešavanje zasnovano na matricama niskog ranga (eng. Low-Rank Adaptation - LoRA) predstavlja osnovu novog state-of-the-art načina za fino prilagođavanje modela [10]. U ovom pristupu polazi se od zapažanja da se umesto direktnog menjanja težina modela zapravo prate promene koje želimo da primenimo na matričnom nivou. Dakle, uz svaku matricu koju želimo da treniramo možemo dodati još jednu matricu iste dimenzionalnosti koja će se trenirati i čije će vrednosti najzad biti sabrane sa već postojećim težinama modela koje se neće menjati. Uštede na memorijskom i računskom nivou ogledaju se u tome što se matrica težina koje se obučavaju predstavlja kao proizvod dve manje matrice. Proces predstavljanja matrice kao proizvod dve manje matrice naziva se dekompozicija matrice (eng. matrix decomposition) i na slici 3.2 ilustrovan je jedan primer uštede broja parametara koje je potrebno čuvati za proizvoljnu matricu dimenzija 5x5. Ovaj proces je moguće ostvariti u oba

smera - od manjih matrica u veće, kao i razlaganje velike matrice u proizvod dve manje.



Slika 3.2: Matrica dimenzija 5x5 može se predstaviti kao proizvod matrica 5x1 i 1x5. Na ovaj način, umesto čuvanja 25 težina, dovoljno je čuvati njih 10. Ušteda je još izraženija što se više povećava dimenzionalnost osnovne matrice.

Parametar na koji je potrebno obratiti pažnju prilikom treniranja pomoću LoRA tehnike je njen rank - **r**, a označava unutrašnju dimenzionalnost matrica koje grade osnovnu matricu. Sa povećanjem parametra **r** rastu memorijski i računski resursi potrebni za treniranje, dok se zauzvrat dobija na većoj preciznosti modela. Zavisnost veličine ranka u odnosu na ukupan broj parametara dat je u tabeli 3.3. Iako se za rank često uzimaju vrednosti između 8 i 64, ovde se vidi da je ušteda značajna čak i za mnogo veće vrednosti.

Još jedan hiperparametar koji se oved uvodi je α koji se koristi da označi koeficijent sa koji će se pomnožiti dekomponovana matrica pre dodavanja. Na ovaj način utiče se na stopu učenja modela.

Ovime dobijamo i formulu za računanje nove matrice težina, kao

$$W' = W + \frac{\alpha}{r} BA \tag{3.3}$$

U originalnom radu, iako je moguće iskoristiti LoRA princip nad bilo kojom matricom, dekompozicija samo matrica pažnje dovela je do rezultata koji su komparativni sa potpunim finim podešavanjem. Ovime se pokazao veliki potencijal nove tehnike, kao i izneo dokaz da dimenzionalnosti današnjih modela nisu neophodne i da znanje i mogućnosti modela često žive u manjem podprostoru.

Najveći problem kod LoRA pristupa je odabir hiperparametara, kao i slojevi nad kojima se koristi. Na Majkrosoftovom sajtu korišćeni su primeri gde hiperparametar

Rank	7 mlrd.	13 mlrd.	70 mlrd.	180 mlrd.
1	167,332 *(0.002%)*	228,035 *(0.002%)*	529,150 *(0.001%)*	848,528 *(0.000%)*
2	334,664 *(0.005%)*	456,070 *(0.004%)*	1,058,301 *(0.002%)*	1,697,056 *(0.001%)*
4	669,328 *(0.010%)*	456,070 *(0.007%)*	1,058,301 *(0.003%)*	1,697,056 *(0.002%)*
8	1,338,656 *(0.019%)*	1,824,281 *(0.014%)*	4,233,202 *(0.006%)*	6,788,225 *(0.004%)*
16	2,677,312 *(0.038%)*	3,648,561 *(0.028%)*	8,466,404 *(0.012%)*	13,576,450 *(0.008%)*
512	85,673,987 *(1.224%)*	116,753,964 *(0.898%)*	270,924,934 *(0.387%)*	434,446,406 *(0.241%)*
1024	171,347,973 *(2.448%)*	233,507,927 *(1.796%)*	541,849,869 *(0.774%)*	868,892,813 *(0.483%)*

Tabela 3.1: Upoređivanje broja (i procenata) trenabilnih parametara u zavisnosti od veličine modela i ranka matrica

r uzima vrednosti 8 i 16, a parametar α je njegova duplirana vrednost[63], dok rad Hu i Shen-a koristi 32, dok se za α uzima dvostruko ili četvorostruko manja vrednost. Rad koji predstavlja QLora pokazuje da je razlika kada se koristi r između 8 i 256 zanemarljiva [64]. Na Internetu postoje i druge studije koje se bave istraživanjem i upoređivanjem ovih parametara [65].

Kako je ovaj pristup rešavanju problema prilagođavanja nov pojam, očekuje se još značajnih rezultata koji će dovesti do boljeg razumevanja i optimizacija, pogotovo kada je reč o korišćenju hiperparametara. Bez obzira, na to on je već ostvario značajne rezultate koji su doveli do razvoja daljih unapređenja u vidu novih pristupa finom podešavanju, kao što su QLora [64], AdaLoRA [66], Alora [67] i drugi.

3.7 Izazovi prilikom finog podešavanja

Nasuprot brojnim pogodnostima koje fino podešavanje donosi, uvek se treba paziti i na određene probleme koji mogu da se pojave. Neki od njih dati su u nastavku, dok metode za njihovo sprečavanje prevazilaze okvire ovog rada:

- **Katastrofalno zaboravljanje (eng. Catastrophic Forgetting)** – Ovaj problem najčešće se javlja ukoliko koristimo potpuno fino podešavanje u kojem se menja celokupna konfiguracija modela. U ovom slučaju moguće je da se model toliko fokusira na zadate probleme da izgubi moć generalizacije. Rešenje: Elastična konsolidacija težina i progresivno učenje, kao i korišćenje metoda učenja bez ili sa malo primera.

- **Preprilagođavanje na malim skupovima** – Ukoliko je skup podataka premali, model može da zapamti tačan obrazac umesto da se fokusira na efikasnu generalizaciju. Rešenje: Povećanje skupa podataka, različite tehnike regularizacije [68] i rano zaustavljanje.

- **Računska i memorijska zahtevnost** – Kao i u inicijalnom treniranju, i prilikom finog podešavanja ukoliko prosleđujemo modelu veliki skup podataka ili ukoliko se trenira veći deo modela može doći do visoke računske i memorijske zahtevnosti.

  Rešenje: Korišćenje metoda parametarski efikasnog finog podešavanja.

- **Osetljivost na hiperparametre** – Prilikom odabira količine pruženih podataka, kao i količine parametara za treniranje kao i slojeva na kojima će se fino podešavanje ostvariti velika pažnja se mora posvetiti efikasnom odabiru svih ponuđenih parametara.

  Rešenje: Korišćenje Grid Search-a i Bajesovih optimizacija [69]

- **Pristrasnost prema podacima** – Fino podešavanje nad pristrasnim podacima takođe produbljuje pristrasnost.

  Rešenje: Korišćenje tehnika za razbijanje pristrasnosti [70]

- **Negativan transfer** – Ako se domen podataka za fino podešavanje suviše razlikuje od domena pretreniranja, model može da pogorša performanse umesto da ih poboljša. Ovo se u literaturi označava kao negativan transfer (eng. Negative Transfer) [71].

Rešenje: Korišćenje treninga na domenima koji su između izvornog i ciljnog, kao i domen-adaptivnog pretreniranja (eng. Domain-Adaptive Pre-Training - DAPT) [72]

- **Nestabilnost pri učenju** – Na osnovu radova kao što je *On the Stability of Fine-tuning BERT* pokazalo se da fino podešavanje može biti vrlo nestabilno, pri čemu izbor ili čak redosled primera koji se modelu prosleđuju za iste parametre mogu dati značajno različite rezultate

  Rešenje: Pokretanje više treninga, balansirani batch sampling, tehnike kalibracije [73]

Još jedan od problema je i međusoban uticaj različitih metoda finog podešavanja na model. Ukoliko neke metode pokazuju dobre rezultate kada se primene pojedinačno, to ne znači da će model dobro da se ponaša i kada se one međusobno iskombinuju. U radu AdapterFusion: Non-Destructive Task Composition for Transfer Learning autori navode da kombinovanje razli;itih adapter tehnika (ili dodavanje LoRA slojeva preko adaptera) dovodi do redudantnih parametara i smanjenja efikasnosti, bez značajnog dobitka na tačnosti [74].

Glava 4

Implementacija rešenja

Praktična implementacija rešenja zasniva se na korišćenju unapred treniranog jezičkog modela DistilBERT dostupnog preko platforme Hugging Face Transformers [6]. Ovaj model moguće je trenirati za potrebe rešavanja različitih zadataka, ali u ovom radu biće korišćen za rešavanje problema analize sentimenta, što predstavlja problem binarne klasifikacije teksta. Da bi model mogao uspešno da se koristi u ovu svrhu, neophodno je dodati na njega izlazni sloj, u ovom slučaju za potrebe binarne klasifikacije.

Preuzimanje baznog BERT modela vrši se pomoću metode

```
AutoModel.from_pretrained("bert-base-uncased")
```

AutoModel je deo *transformers* biblioteke i često je korišćen za dobavljanje različitih modela sa HF platforme. Parametar „distilbert-base-uncased" predstavlja DistilBERT model koji uvozimo, base predstavlja njegovu osnovnu verziju, dok „uncased" označava da model ne uzima u obzir razliku između velikih i malih slova (za ove potrebe koristi se „cased" verzija modela). Moguće je koristiti i biblioteku ıAutoModelForClassificationHead sa istom metodom i argumentima koja vraća istreniran model posebno namenjen za primenu nad klasifikacionim zadacima, međutim, ovo predstavlja sistematičniji način za upoznavanje sa strukturom modela.

U svakoj od implementacija modela, na ovako dobijen model dodaje se klasifikaciona glava kako bi se model prilagodio za rešavanje konkretnog problema. Ovo je ostvareno dodavanjem linearnog modula koji uzima vektor dužine 768, što predstavlja unutrašnju dimenziju poslednjeg sloja DistilBERT modela i transformiše u dimenziju veličine 2 za potrebe binarne klasifikacije.

Još jedna zajednička stvar koju svaka verzija fine-tuninga koristi je preopterećivanje funkcije forward(), koja se nalazi unutar svakog modula koji dodajemo korišćenjem torch.nn.Module klase [75]. Ova funkcija je ključna komponenta svake klase izvezene iz `torch.nn.Module` i odgovorna je za prolazak informacija kroz mrežu jer se u njoj definiše na koji način se ulazni podaci obrađuju i dalje prosleđuju sledećem sloju u obradi.

Svi modeli trenirani su na tri različite dimenzije Large Movie Review seta podataka. Oni sadrže 100, 1 000 i 22 500 instanci za treniranje i redom 10, 100 i 2 500 instanci na skupu za validaciju. Trening skup čini 25 000 instanci sa kojima model nije upoznat.

4.1 Implementacija potpunog finog podešavanja

Implementacija potpunog finog podešavanja je direktno odrađena jednostavnim dodavanjem poslednjeg sloja na postojeći pretrenirani BERT model. U taj poslednji sloj dodaje se Dropout sloj sa stopom od 0.1 radi regularizacije i smanjenja rizika od prenaučenosti. Linearni sloj mapira skriveni sloj - dimenzije 768 u slučaju BERT-a u prostor dimenzije 2, što odgovara binarnoj klasifikaciji (pozitivno/negativno).

Ulaz u klasifikacionu glavu predstavlja vektor CLS tokena (eng. classification token). Ovaj token je specijalni simbol koji se na početku dodaje svakoj sekvenci prilikom obrade u BERT arhitekturi. Njegova skrivena reprezentacija nakon prolaska kroz poslednji sloj mreže koristi se kao agregatna reprezentacija čitave sekvence, pa je samim tim pogodna za zadatke klasifikacije na nivou rečenice ili dokumenta.

Ovako implementiran model sadrži oko 110 miliona parametara, od kojih su svi postavljeni za treniranje.

4.2 Implementacija finog podešavanja klasifikacione glave

Implementacija ovog metoda fine-tuninga zasniva se na sličnoj ideji kao i za potpuno fino podešavanje, sa razlikom u tome što se svi unutrašnji slojevi modela zavrzavaju i nisu podložni za treniranje. Za treniranje se koriste samo parametri koji se nalaze u poslednjem sloju. U slučaju BERT modela, u pitanju je 1538 parametara

(768 parametara poslednjeg izlaznog sloja * 2 - dimenzija izlaznog sloja u binarnoj klasifikaciji), što predstavlja 0.000014% ukupnog proja parametara.

4.3 Implementacija finog podešavanja zasnovanog na adapterima

Pristup zasnovan na adapterima pri treniranju BERT modela zasniva se na ubacivanju instanca klase AdapterModule koja implementiraju jednostavnu kompoziciju funkcija za projekciju ulaznog vektora dimenzije 768 u vektor niže dimenzionalnosti, primenu ReLU funkcije i ponovnu projekciju nazad u originalnu dimenzionalnost. U ovoj implementaciji, primenjuje se Houlsby arhitektura u kojoj se adapteri stavljaju i posle attention i posle feedforward sloja u mreži.

Normalizacija (LayerNorm) koristi se nakon dodavanja adaptera na rezidualnu putanju. Njena osnovna uloga je stabilizacija treniranja, jer normalizacijom izlaza smanjuje varijansu vrednosti koje prolaze kroz mrežu i time smanjuje šansu od eksplodirajućih ili nestajućih gradijenata. U kontekstu adaptera, normalizacija obezbeđuje da doavanje novih slojeva za treniranja ne naruši balans reprezentacija koje su već naučene u pretreniranom modelu. Ovo uz inicijalno postavljanje težina u up-projekciji obezbeđuje kontrolisanu integraciju novih parametara u postojeću arhitekturu.

Implementacijom adaptera broj parametara za treniranje je 2 380 802, što čini 2.13% od ukupnog broja parametara modela. Model je obučavan u zavisnosti od unutrašnje dimenzionalnosti korišćene u adapterima, gde uzimaju vrednosti 64 i 128.

4.4 Implementacija finog podešavanja zasnovanog na matricama niskog ranga

Implementacija ove vrste finog podešavanja izvedena je kroz dve klase: LoRALayer i LoRALinear.

Klasa LoraLayer definiše niskorangiranu aproksimaciju linearne transformacije pomoću dve matrice A - dimenzija $in\_features \times r$ i B - dimenzija $r \times out\_features$, gde r predstavlja rang, a druge dve promenljive originalne dimenzije osnovne matrice. Na izlazu se LoRA doprinos računa prema formuli:

$$\Delta h(x) = \frac{\alpha}{r}(xA)B, \tag{4.1}$$

gde je α faktor skaliranja doprinosa.

U inicijalizaciji, matrica A se inicijalizuje pomoću Kaiming[**?**] uniformne raspodele koja omogućava stabilian tok gradijenta kroz ReLU aktivacije, dok je matrica B inicijalizovana nulama. Na ovaj način obezbeđuje se zadržavanje osnovnim svojstava pretreniranog modela uz mogućnost naknadnog doprinosa LoRA slojeva tokom treniranja.

Klasa LoRALinear obmotava postojeći sloj tako da se originalne težine zamrzavaju, dok se ostatak koristi kao doprinos u učenju. Izlaz iz ovog sloja formira se kao suma originalnog linearnog sloja i LoRA komponente. Na ovaj način model uči samo matrice A i B čime se postiže značajno smanjenje broja trenirajućih parametara, na ukupno 296 450, što predstavlja 0.26% ukupnog broja parametara.

Glava 5

Poređenje rezultata

5.1 Poređenje procesa treniranja

Vreme potrebno za obuku, kao i hardverski resursi potrebni za taj proces, predstavljaju bitne kriterijume za odabir pristupa kada je fino podešavanje u pitanju. Ova dva aspekta određuju održivost i isplativost metode u realnim uslovima, posebno u slučajevima kada se radi sa ograničenim resursima ili kada je potrebno brzo isporučivanje.

Analiza vremenske složenosti

Tabela 5.1 prikazuje vreme potrebno za treniranje razlličitih varijanti fine-tuning pristupa nad tri skupa različite veličine. Već na prvi pogled jasno je da postoji eksponencijalan rast vremena treniranja u odnosu na beoj prosleđenih instanci za treniranje. Na malom skupu svi modeli završavaju trening u roku od nekoliko sekundi do nekoliko minuta, pri čemu je i najsporiji model (full fine-tuning) treniran za manje od 9 minuta. Ukoliko se gleda prosečno vreme izvršavanja jedne epohe, razlika je još manje varijabilna. Sa porastom podataka za treniranje, razlike u odabiru modela postaju sve izraženije, pa tako ukupno vreme treniranja pristupa zasnovanog na adapterima dostiže 25 minuta, podeljenih u 3 epohe.

Ukoliko se posmatra vreme potrebno za završetak jedne epohe, primećuje se da za male skupove adapteri predstavljaju najneoptimalnije rešenje, dok za velike skupove tu ulogu preuzima full fine-tuning. Head fine-tuning ovde se postavlja kao najoptimalniji izbor. LoRA pristup pokazuje bolje performanse u odnosu na adaptere, međutim daleko zaostaje za prilagođavanjem samo poslednjeg klasifikacionog

Set/Model	Small	Medium	Large
Full fine-tuning	8.81 (3 x 2.94)	43.09 (2 x 21.54)	520.97 (1 x 520.97)
Head fine-tuning	0.98 (1 x 0.98)	39.96 (5 x 7.99)	761.07 (4 x 190.27)
Adapters (dim=48) fine-tuning	7.21 (2 x 3.61)	20.49 (1 x 20.49)	997.67 (2 x 498.84)
Adapters (dim=96) fine-tuning	6.93 (2 x 3.46)	41.69 (2 x 20.84)	1505.78 (3 x 501.93)
LoRA (r=32, α=64) fine-tuning	2.21 (1 x 2.21)	67.51 (4 x 16.88)	397.64 (1 x 397.64)
LoRA (r=64, α=32) fine-tuning	2.58 (1 x 2.58)	87.09 (5 x 17.42)	821.64 (2 x 410.82)

Tabela 5.1: Ukupno vreme za obuku modela (broj epoha x prosečno vreme za jednu epohu)

sloja.

Iz rezultata treniranja vidi se da izbor konfiguracije u slučaju adaptera i LoRA zasnovanog pristupa ne utiče značajno na vreme potrebno za prolazak kroz jednu iteraciju, ali može imati uticaj na brzinu konvergencije modela, pa samim tim i ukupnog trajanja treniranja. Ove razlike su, međutim, primetne samo za velike skupove podataka, a izraženije su kod LoRA pristupa.

Posmatrajući broj izvršenih epoha, skoro svaki pristup doveo je do primene early-stopping mehanizma koji je podešen da prestaje sa treningom ukoliko dođe do pogoršanja performansi na metrici koja meri gubitak pri validaciji. Gledano po ovome, head fine-tuning pokazuje stabilnost u učenju, dok su LoRA i full fine-tuning veoma podložni preprilagođavanju na većim (a LoRA i na malim) skupovima.

Analiza računske složenosti

Što se razlike u računskim složenostima, ovde je razlika najizraženija u zavisnosti od odabrane tehnike. Tabela 5.2 prikazuje poređenje modela u odnosu na zauzetost GPU memorije i broja treniranih parametara. Ovde full fine-tuning očekivano koristi nekoliko puta više memorije od ostalih pristupa i zahteva ažuriranje 100% parametara. Sa druge strane, head fine-tuning obučava svega 0.000014% od ukupnog broja parametara i zahteva najmanje memorije za izvršavanje.

Adapteri i LoRA pristup postavljaju se između ova dva pristupa, pri čemu su daleko bliži head nego full fine-tuning-u. Koriste gotovo identične količine GPU memoruje, međutim LoRA obučava 3 puta manje parametara od pristupa sa adapterima. Vrednosti prikazane u tabeli odnose se na konfiguraciju adaptera sa unutrašnjom dimenzijom 96 i LoRA pristup sa parametrima r=32 i α=64. U drugim konfiguracijama dobijaju se druge vrednosti, ali razlika je neznatna ukoliko se samo posmatra ušteda broja parametara za treniranje u odnosu na početnih 109 miliona.

Model	Full	Head	Adapters	LoRA
GPU Usage (MB)	2114	855	899	888
Num Trainable Params	109.483.778 *100.00%*	1.538 *0.00%*	3.561.218 *3.15%*	1.181.186 *1.04%*

Tabela 5.2: Prosečan GPU usage i broj parametara za treniranje u zavisnosti od korišćenog modela. Vrednosti za Adaptere su u verziji sa unutrašnjom dimenzijom 96, dok su vrednosti za LoRA prikazane sa konfiguracijom r=32, α=64

Uporedna analiza vremenske i računske složenosti

Na osnovu kombinovanih rezultata vremenske i resursne analize dolazi se do sledećih zaključaka. Full fine-tuning pokazuje se kao metod za čije treniranje je potrebno izdvojiti primetno najveće resurse po svakom osnovu. To čini ovaj metod pogodan jedino u scenarijima gde resursi nisu ograničeni. Sa druge strane, head fine-tuning ima najmanju računsku složenost i vreme izvršavanja. Ova metoda pokazuje se korisnom u eksperimentalnim uslovima kada je potrebno brzo validirati ideju ili kada je dostupna količina podataka mala.

Adapteri predstavljaju srednje rešenje, pri čemu zahtevaju umereno povećanje memorije i broja parametara i pružaju dobar kompromis, ali ne efikasan kao LoRA pristup koji je primetno efikasniji po broju parametara za treniranje, kao i vremena obuke. Variranje parametara ove dve metode ne odražava se značajno ni na vreme, ni na računske zahteve potrebne za treniranje.

5.2 Poređenje kvaliteta modela

Poređenje kvaliteta različitih pristupa finom podešavanju podeljena je po modelima koji su trenirani nad različitom veličinom skupa za treniranje. U cilju evaluacije korišćene su četiri standardne metrike: tačnost (Accuracy), preciznost (Precision), F1 mera (F1 score) i površina ispod ROC krive (ROC AUC). Tačnost meri udeo ispravno klasifikovanih primera u ukupnom broju primera. Ovde je posebno značajna sa obzirom da su instance za testiranje potpuno balansirane - polovina je pozitivna, polovina negativna. Preciznost ukazuje na pouzdanost pozitivnih predikcija, odnosno koliki deo primera koji su označeni kao pozitivni zaista pripada toj klasi. F1 mera predstavlja harmonijsku sredinu tačnosti i odziva (recall). U balansiranim skkupovima, kao što je ovaj, često će vrednost biti bliska tačnosti, ukoliko model sa jednakom učestalošću pogrešno klasifikuje pozitivne i negativne instance. Roc AUC vrednost pokazuje sposobnost modela da razlikuje pozitivnu od negativne klase u različitim pragovima odlučivanja i često se smatra robusnijom metrikom u poređenju sa tačnošću

Analiza modela treniranih na malom skupu podataka

Analiza rezultata na malom skupu od 100 instanci (Tabela 5.3) pokazuje značajne razlike u zavisnosti od odabranog pristupa finom podešavanju.

Najbolje performanse postiže full fine-tuning, sa primetno najboljim vrednostima za sve merene metrike, dostižući vrednost za tačnost od čak 0.81. Ovo potvrđuje da čak i na malim skupovima podataka, kada se svi parametri modela optimizuju, može doći do relativno dobrog prilagođavanja zadatku.

Fino podešavanje klasifikacione glave pokazuje izrazito loše rezultate, sa tačnošću od 0.5 i F1 merom 0.37. Ovo ukazuje da samo prilagođavanje poslednjeg sloja u modelu nije dovoljno da se izvuku sve značajne informacije na malom pruženom skupu podataka. Ovo implicira da model nema kapacitet da iskoristi semantičku strukturu koju BERT već poseduje jer nema mogućnost za dubljom adaptacijom.

Adapteri i LoRA daju rezultate koji variraju između vrednosti prethodna dva pristupa. Bitno je napomenuti da LoRA daje prosečno bolje rezultate od adaptera, kao i da performanse variraju u zavisnosti od konfiguracije rešenja. Tako Adapteri sa unutrašnjom dimenzijom 96 daju bolje rezultate od verzije sa unutrašnjom dimenzijom 48, što znači da drugi pristup ipak dovodi do gubitka značajnih informacija. Varijacije između LoRA modela su manje, u korist konfiguracije u kojoj je α parametar duplo veći od ranga.

Small	**Accuracy**	**Precision**	**F1 Score**	**ROC AUC**
Full fine-tuning	0.81	0.81	0.81	0.88
Head fine-tuning	0.50	0.51	0.37	0.52
Adapters (dim=48)	0.55	0.56	0.53	0.58
Adapters (dim=96)	0.59	0.60	0.58	0.63
LoRA (r=32, α)=64)	0.63	0.64	0.62	0.71
LoRA (r=64, α)=32)	0.61	0.64	0.59	0.68

Tabela 5.3: Metrike u zavisnosti od modela na skupu za treniranje od 100 instanci

Analiza modela treniranih na srednjem skupu podataka

Svi posmatrani modeli trenirani na skupu podataka od 1000 instanci (Tabela 5.4) pokazuju poboljšanje u performansama. I u ovom slučaju, full fine-tuning pokazuje najbolje vrednosti, dostižući tačnost od 0.9.

Head fine-tuning beleži značajan skok performansi u poređenju sa modelom treniranom na malom skupu, gde F1 vrednost dostiže 0.74, u odnosu na pređašnjih 0.37. Dobijeni rezultati pokazuju rezultate bolje i od pristupa zasnovanog na adapterima, čime se ukazuje na mogućnost korišćenja ovog pristupa ukoliko je na raspolaganju dovoljno bogat skup podataka.

Rezultati podešavanja zasnovanog na adapterima pokazuju namanji napredak u odnosu na modele trenirane na malom skupu. Iako su razlike u dve konfiguracije manje izražene, čime se ukazuje na manje variranje u zavisnosti od unutrašnjih podešavanja, ovaj pristup pokazao je najlošije rezultate, dostižući tačnost od 0.64, odnosno 0.62, ovog puta, u korist modela sa manjim unutrašnjim dimenzijama.

LoRA pristup ovde pokazuje izuzetne rezultate, pri čemu konfiguracija r=32, α=64 postiže identične vrednosti kao i full fine-tuning, dok druga konfiguracija poka-

zuje minimalno lošije rezultate. Ovi rezultati ukazuju da LoRA predstavlja izuzetno efikasnu alternativu full fine-tuningu, pri čemu koristi znatno manji broj treniranih parametara.

Medium	Accuracy	Precision	F1 Score	ROC AUC
Full fine-tuning	0.90	0.90	0.90	0.96
Head fine-tuning	0.74	0.74	0.74	0.82
Adapters (dim=48)	0.64	0.65	0.64	0.69
Adapters (dim=96)	0.62	0.65	0.59	0.71
LoRA (r=32, α)=64)	0.90	0.90	0.90	0.97
LoRA (r=64, α)=32)	0.89	0.89	0.89	0.96

Tabela 5.4: Metrike u zavisnosti od modela na skupu za treniranje od 1000 instanci

Analiza modela treniranih na velikom skupu podataka

Rezultati modela treniranih na velikom skupu od 25.000 instanci (Tabela 5.5) nastavljaju trendove, pri čemu full fine-tuning zadržava najviše performanse (F1=0.93, ROC AUC=0.98), međutim uz značajno manje razlike u odnosu na ostale modele iz prethodna dva scenarija.

Head fine-tuning beleži tačnost od 0.84 što je primetno poboljšanje u odnosu na manje skupove, međutim, i u ovom slučaju pokazuje najlošije rezultate od svih posmatranih metoda. Pristup sa adapterima pokazuje za nijansu bolje rezultate od ovog pristupa, uz minimalnu razliku od odabrane konfiguracije.

LoRA pristup i dalje pokazuje značajne rezultate, komparabilne sa potpunim finim prilagođavanjem, pokazujući moguću primenu i na velikim skupovima podataka.

Uporedna analiza različitih veličina skupova za treniranje

Sveukupno posmatrano, potpuno fino podešavanje pokazuje se kao najuspešniji metod za postizanje najboljih performansi, posebno se ističući u situacijama kada je na raspolaganju mali skup za treniranje, gde adaptivni pristupi nisu dovoljno jaki. Uprkos primetnim razlikama na malom skupu za treniranje, LoRA se izdvaja kao metoda koja gotovo u potpunosti dostiže ovaj pristup već na skupovima koji sadrže

Large	Accuracy	Precision	F1 Score	ROC AUC
Full fine-tuning	0.94	0.94	0.94	0.98
Head fine-tuning	0.84	0.84	0.84	0.92
Adapters (dim=48)	0.86	0.86	0.86	0.94
Adapters (dim=96)	0.87	0.87	0.87	0.94
LoRA (r=32, α)=64)	0.93	0.93	0.93	0.98
LoRA (r=64, α)=32)	0.93	0.93	0.93	0.98

Tabela 5.5: Metrike u zavisnosti od modela na skupu za treniranje od 25000 instanci

preko 1000 instanci. LoRA takođe pokazuje konstantno najbolje rezultate u odnosu na sve druge adaptivne pristupe.

Ponašanje adaptera pokazuje se kao najneujadnečenije, pokazujući se kao dobar izbor ukoliko je skup za treniranje mali ili dovoljno veliki, dok u nekim trenucima pokazuju zaostajanje za head fine-tuning pristupom. Ovo sugeriše da izbor dimenzije adaptera i strategija optimizacije igraju ključne uloge, koje je potrebno pažljivo odabrati, posebno na manjim skupovima.

Head fine-tuning pokazao se kao pristup koji je preporučljiv isključivo ukoliko je dostupan skup dovoljno veliki. Pokazao je najlošije rezultate na velikom skupu za treniranje, međutim, sa vrednostima uporednim sa pristupom zasnovanom na primeni adaptera, pri čemu je ostvario čak i bolje performanse na skupu srednje veličine.

Glava 6

Zaključak

Prikazani eksperimenti u ovom radu jasno pokazuju da izbor tehnike za fino podešavanje modela ima presudan uticaj na odnos između potrebnih resursa za rad sa modelom, kao i postignut kvalitet rešenja. Dok je tradicionalni pristup potpunog fine-tuninga u stanju da postigne odlične rezultate, njegova primena u realnim okruženjima često nailazi na barijeru visokih hardverskih i vremenskih troškova. Alternativne tehnike poput treniranja samo izlazne glave, korišćenja adaptera ili primena LoRA modula demonstriraju značajan potencijal da se zadrže visok nivo performansi uz značajno rasterećenje vremenskih i računskih resursa. Time se omogućava mnogo pristupačnije eksperimentisanje i brže pravljenje prototipa, što može biti značajno u eksperimentalnim, kao i dinamičnim okruženjima. Ove metode demonstriraju veću isplativost sa povećanjem broja dostupnih podataka za treniranje.

Jedan od krajnjih zaključaka rada jeste traženje kompromisa između cene i kvaliteta modela. Ovaj rad jasno ilustruje kako zavisnost od veličine skupa za treniranje i različite primenjene tehnike finog podešavanja, tako i aspekte cene koje svaki od modela zahteva za njegovo korišćenje. U različitim situacijama potrebno je pristupiti na drugačiji način, kao i balansirati između dostupnih resursa. Ovo nije situacija u kojoj one-size-fits-all, već je prikazano da se pri radu sa modelima moraju razumeti i analizirati razni aspekti pre pristupanja finom podešavanju, kako bi se zadati cilj dostigao uz minimalne druge troškove.

Posebno je važno istaći da su rezultati ovog rada potvrda šireg trenda u oblasti obrade prirodnog jezika i dubokog učenja. Modeli su dostigli takav nivo razvijenosti da se istraživačka pažnja sve više preusmerava sa kreiranja novih, većih i složenijih arhitektura na pronalaženje efikasnijih i praktičnijih načina njihovog prilagođavanja tazličitim zadacima. Na ovaj način, postojeće arhitekture dobiće mogućnost efika-

snog „recikliranja" zarad dobijanja na diverzifikaciji kao i produbljivanju znanja i prilagođavanja modela najrazličitijim potrebama.

Upravo u ovome leži značaj adaptera i LoRA pristupa. Ove tehnike omogućavaju modularnost i fleksibilnost, jer dodatne komponente mogu biti nezavisno trenirane i lako uklopljene u već postojeću arhitekturu. Ovi pristupi odgovorni su i za poboljšanje skalabilnosti, kada je potrebno održavati čitave kolekcije specijalizovanih modela. Ovi scenariji koji su vrlo učestali u realnim okruženjima omogućavaju veliku memorijsku uštedu kao i vreme potrebno za modifikovanje i distribuciju novih rešenja.

Rezultati izvedeni u ovom radu pokazuju i da postoji prostor za dalje unapređenje samih tehnika. Iako adapteri i LoRA već sada daju impresivne rezultate, dalja istraživanja u smeru boljeg odabira dostupnih parametara, regularizacije i optimizacije ostavljaju dosta prostora za dalje uštede i kao i dobijanje kvalitetnijih rešenja. Posebno obećavajuća je mogućnost kombinovanja različitih tehnika, koje iako nisu obrađene ovde, predstavljaju sledeći logičan korak u ispitivanju poboljšanja modela specifičnom zadatku.

Treba napomenuti i da priloženi efikasni pristupi prilagođavanju modela otvaraju vrata širokom spektru korisnika. Dok je pun fine-tuning rezervisan za institucije sa značajnim hardverskim kapacitetima, korišćenje ovih pretreniranih modela uz parametarski efikasne metode omogućuju istraživačima, manjim timovima, kao i pojedinačnim korisnicima da učestvuju u naprednom radu sa velikim jezičkim modelima, čime se podstiče dostupnost i demokratizacija tehnologije. Gledano u perspektivi, čini se da u budućnosti pri radu sa jezičkim modelima najbolje rešenje neće pripasti nužno onima koji imaju resurse za pravljenje najvećeg modela, već onome ko na najefikasniji način može iskoristiti postojeća rešenja. Time se otvara i prostor za kreativnije i inovativnije pristupe rešavanju konkretnih problema.

Važno je naglasiti i aspekt održivosti. S obzirom na to da treniranje velikih modela zahteva ogromne količine energije, preusmeravanje na parametarski efikasne metode može značajno doprinetu smanjenju prirodnih resursa potrebnih za njihovo održavanje, kao i smanjiti količinu zagađenja u prirodi koje oni prave. Brojni autori ističu značajan ugljenični otisak (carbon footprint) koji nastaje prilikom rada sa velikim modelima kao i potrošnju energije koja je potrebna za svakodnevni rad [76]. Zbog ovoga je maksimalna optimizacija svih dobijenih modela prioritet kojij će doprineti boljoj održivosti i većoj ekološkoj odgovornosti.

Sve navedeno ukazuje na to da buduća istraživanja u ovoj oblasti treba da budu

usmerena na dalju optimizaciju metoda finog podešavanja, kao i na razvoj kriterijuma za odabir najpogodnijeg pristupa u zavisnosti od zadatka, dostupnih resursa i željenog nivoa performansi. Drugim rečima, umesto jedinstvenog rešenja, biće sve važnije razvijati fleksibilne metodologije koje se mogu prilagođavati specifičnim potrebama. Iako su obrađena rešenja u ovom radu relativno novi pristupi u poznatom problemu, svakodnevno se javljaju i nova. Neke od najskorijih tehnika su QLoRA (eng. Quantized Low-Rank Adaptation)[64] koja kombinuje kvantizaciju sa poznatom LoRA adaptacijom kako bi omogućila efikasno treniranje veoma velikih modela na ograničenim hardverskim resursima. Takođe, postoje i pristupi poput QDyLoRA i dinamičkih metoda prilagođavanja rangova [77] koje se istražuju zarad dodatnog poboljšanja odnosa dobijenih performansi i potrošnje resursa.

Na kraju, ovaj rad pokazuje da je razmevanje i poređenje različitih tehnika finog podešavanja ključno za pravilno korišćenje velikih jezičkih modela. Dobijeni rezultati pružaju osnovu za donošenje informisanih odluka u praksi dok istovremeno ukazuju na pravce daljeg napretka u istraživanju. Na taj način, rad doprinosi širem razumevanju ne samo tehničkih aspekata problema, već i njegove strateške važnosti u budućem razvoju veštačke inteligencije.

Bibliografija

[1] Roberto Cordeschi. Ai turns fifty: Revisiting its origins. *Applied Artificial Intelligence*, 21(4-5):259–279, 2007.

[2] Elon University News Bureau. Survey: 52% of u.s. adults now use ai large language models like chatgpt. `https://www.elon.edu/u/news/2025/03/12/survey-52-of-u-s-adults-now-use-ai-large-language-models-like-chatgpt/`, Mar 2025.

[3] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. *Data-centric AI: Perspectives and Challenges*, pages 945–948.

[4] Joel Eapen and VS Adhithyan. Personalization and customization of llm responses. *International Journal of Research Publication and Reviews*, 4(12):2617–2627, 2023.

[5] Abi Aryan, Aakash Kumar Nain, Andy McMahon, Lucas Augusto Meyer, Harpeet Singh Sahota. The Costly Dilemma: Are Large Language Models the Pay-Day Loans of MAchine Leaning? 2023. `https://abiaryan.com/assets/EMNLP%20Submission_Non-Anon.pdf/`.

[6] Hugging Face. Hugging Face: The AI Community Building the Future, 2023. `https://huggingface.co/`.

[7] He, Ruibin and Anastasopoulos, Antonios and Neubig, Graham. A Survey on Parameter Efficient Transfer Learning for NLP. *arXiv preprint arXiv:2203.06904*, 2022.

[8] Wenxuan Wang, Jifan Yu, and Xiang Ren. Parameter-efficient tuning makes a good classification head. *arXiv preprint arXiv:2210.16771*, 2022.

[9] Parameter-Efficient Transfer Learning for NLP, author=Houlsby, Neil and Giurgiu, Andrei and Jastrzebski, Stanislaw and Morrone, Brianna and de Laroussilhe, Quentin and Gesmundo, Andrea and Attariyan, Mona and Gelly, Sylvain, booktitle=Proceedings of the 36th International Conference on Machine Learning (ICML), pages=2790–2799, year=2019, url=https://arxiv.org/abs/1902.00751.

[10] Edward J. Hu and Yelong Shen and Phillip Wallis and Zeyuan Allen-Zhu and Yuanzhi Li and Shean Wang and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[11] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022.

[12] Mingze Gao, Qilong Wang, Zhenyi Lin, Pengfei Zhu, Qinghua Hu, and Jingbo Zhou. Tuning pre-trained model via moment probing, 2023.

[13] Bing Liu. *Sentiment Analysis and Opinion Mining.* 2012.

[14] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[15] Prashant Johri, Sunil Kumar Khatri, Ahmad Al-Taani, Munish Sabharwal, Shakhzod Suvanov, and Avneesh Chauhan. *Natural Language Processing: History, Evolution, Application, and Future Work*, pages 365–375. 01 2021.

[16] Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198. Association for Computational Linguistics, July 2020.

[17] Paul Semaan. Natural language generation: an overview. *J Comput Sci Res*, 1(3):50–57, 2012.

[18] Hugging Face. Gpt-2 model card. `https://huggingface.co/transformers/v2.2.0/pretrained_models.html`, 2019. Accessed: 2025-08-25.

[19] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. Palm: Scaling language modeling with pathways. `https://research.google/blog/pathways-language-model-palm-scaling-to-540-billion-parameters-for-breakthrough` April 2022. Google Research Blog.

[20] Johannes Gerstmayr, Peter Manzl, and Michael Pieber. Multibody models generated from natural language. *Multibody System Dynamics*, 62:249–271, 01 2024.

[21] OpenAI. Gpt-5: Advancing the capabilities of large language models, 2025. Accessed: 2025-08-24.

[22] Jin Guo. Critical tokenization and its properties. *Computational Linguistics*, 23(4):569–596, 1997.

[23] Hugging Face. Tokenizers — hugging face nlp course. `https://huggingface.co/learn/nlp-course/en/chapter2/4`, 2023. Accessed: 2025-01-28.

[24] Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Schulze Buschhoff, et al. Tokenizer choice for llm training: Negligible or crucial? *arXiv preprint arXiv:2310.08754*, 2023.

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725, 2016.

[26] Philip Gage. A new algorithm for data compression. In *C Users Journal*, volume 12, pages 23–38, 1994.

[27] Dimitar Dimitrov, Nikola Ljubešić, Ranka Stanković, Marko Tadić, and Jelena Mitrović. BERTić - the transformer language model for bosnian, croatian, montenegrin and serbian. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2102–2112, Marseille, France, 2022. European Language Resources Association.

[28] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N. and Kaiser, Łukasz and Polosukhin, Illia. Attention is All You Need. *Advances in Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[29] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[30] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[31] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[32] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[33] Daniel Svozil, Vladimír Kvasnicka, and Jiri Pospichal. Introduction to multilayer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, 1997.

[34] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[35] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. Introduces the GELU activation function: $x\,\Phi(x)$, which offers smoother transitions and improved performance across CV, NLP, and speech tasks.

[36] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. Introduces Swish: $f(x) = x{\cdot}\sigma(x)$, a smooth self-gated activation that outperforms ReLU in deep networks.

[37] Ruibin Xiong, Yi Yang, Di He, Kai Zheng, Shuo Zheng, Chao Xing, Hang Zhang, Yelong Lan, Liwei Wang, Tie-Yan Liu, and Maosong Sun. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, 2020.

[38] Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[39] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

[40] Ronen Feldman. Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82–89, April 2013.

[41] Benaissa Azzeddine Rachid, Harbaoui Azza, and Ben Ghezala Henda. Sentiment analysis approaches based on granularity levels. In *Proceedings of the 14th International Conference on Web Information Systems and Technologies*, volume 1, pages 324–331, 2018.

[42] Doaa Mohey El-Din Mohamed Hussein. A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences*, 30(4):330–338, 2018.

[43] Prabha PM Surya and B Subbulakshmi. Sentimental analysis using naive bayes classifier. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–5, 2019.

[44] Munir Ahmad, Shabib Aftab, and Iftikhar Ali. Sentiment analysis of tweets using svm. *Int. J. Comput. Appl*, 177(5):25–29, 2017.

[45] Xi Ouyang, Pan Zhou, Cheng Hua Li, and Lijun Liu. Sentiment analysis using convolutional neural network. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2359–2364, 2015.

[46] Lilis Kurniasari and Arif Setyanto. Sentiment analysis using recurrent neural network. *Journal of Physics: Conference Series*, 1471(1):012018, feb 2020.

[47] GSN Murthy, Shanmukha Rao Allu, Bhargavi Andhavarapu, Mounika Bagadi, and Mounika Belusonti. Text based sentiment analysis using lstm. *Int. J. Eng. Res. Tech. Res*, 9(05):299–303, 2020.

[48] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[49] Tanjim Ul Haque, Nudrat Nawal Saber, and Faisal Muhammad Shah. Sentiment analysis on large scale amazon product reviews. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, pages 1–6, 2018.

[50] Himanshu Pal and Bharat Bhushan. Sentiment analysis on twitter dataset using voting classifier. In *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, volume 1, pages 1–6, 2024.

[51] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[52] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[53] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott

Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[54] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019.

[55] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[56] Clément Christophe, Praveen K Kanithi, Prateek Munjal, Tathagata Raha, Nasir Hayat, Ronnie Rajan, Ahmed Al-Mahrooqi, Avani Gupta, Muhammad Umar Salman, Gurpreet Gosal, Bhargav Kanakiya, Charles Chen, Natalia Vassilieva, Boulbaba Ben Amor, Marco AF Pimentel, and Shadab Khan. Med42 – evaluating fine-tuning strategies for medical llms: Full-parameter vs. parameter-efficient approaches, 2024.

[57] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines, 2021.

[58] Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. Investigating the catastrophic forgetting in multimodal large language models. *arXiv preprint arXiv:2309.10313*, 2023.

[59] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022.

[60] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers, 2020.

[61] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Simple, scalable adaptation for neural machine translation, 2019.

[62] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. On the effectiveness of adapter-based tuning

for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*, 2021.

[63] Microsoft. Lora: Low-rank adaptation of large language models. `https://github.com/microsoft/LoRA`, 2021. Accessed: 2025-08-30.

[64] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc., 2023.

[65] Unsloth Documentation. Lora hyperparameters guide, 2025. `https://docs.unsloth.ai/get-started/fine-tuning-llms-guide/lora-hyperparameters-guide`.

[66] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

[67] Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. Alora: Allocating low-rank adaptation for fine-tuning large language models, 2024.

[68] Xuansheng Wu, Wenhao Yu, Xiaoming Zhai, and Ninghao Liu. Self-regularization with sparse autoencoders for controllable llm-based classification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, page 3250–3260, New York, NY, USA, 2025. Association for Computing Machinery.

[69] Nupur Kalele and Vijay Kumar Kalyani. Grid search and llm assisted hybrid approach for hyperparameter optimization. In *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, volume 3, pages 1–5, 2025.

[70] Manish Tripathi and Er Agarwal. Bias mitigation in nlp: Automated detection and correction. *International Journal of Research in Modern Engineering & Emerging Technology*, 13, 05 2025.

[71] Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer. *IEEE/CAA Journal of Automatica Sinica*, 10(2):305–329, 2023.

[72] Junqi Ding, Bo Li, Chang Xu, Yan Qiao, and Lingxian Zhang. Diagnosing crop diseases based on domain-adaptive pre-training bert of electronic medical records. *Applied Intelligence*, 53(12):15979–15992, 2023.

[73] Yuxi Xia, Pedro Henrique Luz de Araujo, Klim Zaporojets, and Benjamin Roth. Influences on llm calibration: A study of response agreement, loss functions, and prompt styles, 2025.

[74] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021.

[75] torch.nn — pytorch documentation, 2025. Accessed: 2025-08-30.

[76] Peng Jiang, Christian Sonne, Wangliang Li, Fengqi You, and Siming You. Preventing the immense increase in the life-cycle energy and carbon footprints of llm-powered intelligent chatbots. *Engineering*, 40:202–210, 2024.

[77] Hossein Rajabzadeh, Mojtaba Valipour, Tianshu Zhu, Marzieh Tahaei, Hyock Ju Kwon, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. Qdylora: Quantized dynamic low-rank adaptation for efficient large language model tuning, 2024.