Univerzitet u Beogradu - Elektrotehnički fakultet

Katedra za signale i sisteme

DIPLOMSKI RAD

Parametarski efikasno podešavanje jezičkog modela za detekciju pretnji nasiljem u tekstu

Kandidat
Janko Mitrović, 2019/0123

Mentor
Prof. Dr Goran Kvaščev

Beograd, *Septembar* 2024. godine

REZIME RADA

U ovom radu se analiziraju tehnike parametarski efikasnog finog podešavanja (eng. parameter-efficient fine-tuning) pretreniranog jezičkog modela BERT (eng. Bidirectional encoder representation from transformers) u svrhu detektovanja pretnje fizičkim nasiljem, nagovaranja na fizičko nasilje, nagovaranja na samubistvo/samopovređivanje u tekstu na srpskom jeziku na latinici.

Trening skup je napravljen ručno, pa je augmentisan pomoću openai api i dodavanjem šuma.

Za ovaj zadatak klasifikacije korišćene su tehnike LoRA (eng. Low-Rank Adaptation) i QloRA (eng. Quantized Low-Rank Adaptation). Ove tehnike su memorijski štedljive, omogućavaju fino podešavanje modela na specifične zadatke bez potrebe za treniranjem celog modela, tako da prilagođavanja jezičkih modela za neki specifičan zadatak postaju izvodljivi bez velikih računarskih resursa.

Korišćena je 4-bitna kvantizacija, za dodatnu uštedu računarskih resursa, takođe se pokazuje da se kvantizacija može posmatrati i kao metoda regularizacije.

U radu se objašnjavaju pojmovi poput mehanizma pažnje (eng. attention mechanism), i transformer modela, i parametarski efikasnih metoda finog podešavanja.

Korišćenjem jako malog broja parametara – manje od 1 procenta originalnog modela, takođe dodatnom uštedom zbog kvantizacije postiže se pristojna tačnost na testirajućem skupu od približno 90 procenata.

Glavni nedostatak ovog rada je premali trenirajući skup – generalno za zadatke treniranja jezičkih modela. Takođe nedostatak računarsih resursa je onemogućio dovoljno povećavanje broja trenirajućih parametara i postizanje poželjnih rezultata, ali je demonstrirano da sa povećanjem broja trenirajućih parametara eksperimentisanjem sa veličinom LoRA matrica može postići željeni rezultat.

SADRŽAJ

1 UVOD

Pretnje fizičkim nasiljem i nagovaranje na samoubistvo predstavljaju ozbiljan društveni problem zato što su česte u digitalnim prostorima i mogu se brzo širiti i podsticati fizičko nasilje i samodestruktivno ponašanje.

Ovaj problem je prepoznat i u krivičnom zakoniku Srbije, gde su pretnje ubistvom i povređivanjem (član 138. Ugrožavanje sigurnosti), kao i nagovaranje na samoubistvo (član 119. Navođenje na samoubistvo i pomaganje u samoubistvo) prepoznati kao krivična dela.

1.1 Definisanje zadatka i cilj rada

Cilj ovog rada je da istraži kako modeli obrade prirodnog jezika (NLP, eng. Natural Language Processing), zasnovani na savremenim metodama poput **transformera** i **mehanizma pažnje** (eng. attention), mogu pomoći u prepoznavanju ''nasilnog govora''. Specifično u kombinaciji sa parametarski efikasnim finim podešavanjem tehnikama LoRA (eng. Low-Rank Adaptation of Large Language Models) i QLoRA (eng. Quantized Low-Rank Adaptation).

U ono što model treba da detektuje spadaju:

- Pretnje fizičkim nasiljem
- Pretnje ubistvom
- Nagovaranje na fizičko nasilje
- Nagovaranje na samoubistvo
- Nagovaranje na samo povređivanje

A izuzetci koje model treba da klasifikuje kao ''normalan govor'':

- Pretnje samoubistvom – ideja je da ovo više poziv za pomoć nego bilo kakva pretnja nasiljem, takođe ovde spadaju molbe za pomoć pri samoubistvo i slično
- Pretnje samopovređivanjem

Teži primeri za klasifikaciju uključuju primere vezane za:

- ''Geming contex'', Ovo je dodato da se zadatak oteža, model treba da prepozna da se pretnje upućuju igračima u igrici.
- Ostali specifični kontekti poput religijskog, pravnog i slično...

1.2 Zašto baš parametarsi efikasno fino podešavanje

Kada razvijamo model za zadatak klasifikacije teksta, ili generalno bilo koji problem iz oblasti prirodne obrade jezika (eng. Natural Language Processing) možemo trenirati model od nule ili fino podesiti već pretreniran model.

Treniranje modela od nule, kao što je model sa BERT arhikteturom (eng. Bidirectional encder representation form Transformers), zahteva ogromne resurse, ne samo u smislu hardvera nego i u vidu količine podataka. Ova strategija se koristi u slučaju zadatka na nekom jeziku za koje ne postoje već pretrenirani modeli, ili ako je domen zadatka naučno usko specifičan na primer ako je zadatak analiza DNK sekvence u formi teksta.

Sa druge strane, fino podešavanje (eng. Fine Tuning) već pretreniranih modela, kao što je pretrenirani BERT-base, koji je već obučen na velikom skupu podataka (eng. dataset) nam omogućava da iskoristimo bogato znanje jezika koje je model već stekao i da ga prilagodimo specifičnom zadatku koristeći mnogo manji skup podataka.

Fino podešavanje može biti potpuno (eng. Full Fine-Tuning) gde se sve težine modela treniraju i može biti parametarski efikasno (PEFT, eng. Parameter-Efficient Fine-Tuning) gde se trenira samo podskup od svih težina u modelu. Parametarski efikasno fino podešavanje dodatno smanjuje vreme treniranja, količinu potrebne memorije za treniranje, efikasno je za rad sa malim skupovima podataka i kompatibilno je sa kvantizacijom.

2 METODOLOGIJA RADA

U ovoj sekciji ću predstaviti na koji način su prikupljeni podaci za obučavanje jezičkog modela, i koje tehnike finog podešavanja koristim. Pored toga biće predstavljen teoretski uvod u BERT arhikteturu, proces pretreniranja i tokenizacije.

2.1 Metodologija izrade skupa podataka za obučavanje

Finalni skup podataka je napravljen iz više koraka:

1. Pravljenje malog skupa podataka (eng. dataset) ručno u excelu

   Ovaj korak treba da predstavlja osnovu za dalje uvećanje skupa podataka, zato je jako važno da ovaj skup podataka bude izuzetnog kvaliteta – dakle bitno je onemogućiti tačnu klasifikaciju samo na osnovu prostih obeležja poput: broja reči u rečenici, postojanja specifičnih reči, dužine rečenice, postajanja specifičnih znakova interpunkcije, postojanja brojeva, velikih slova, forme rečenice itd. Na primer: ako su u jednoj klasi sve rečenice dugačke oko 50 karaktera, a u drugoj oko 20 karaktera – za model je dovoljno da nauči broj tokena da bi doneo odluku o klasifikaciji – takav model će davati sto postotne rezultate na ovom skupu podataka/tekstova/rečenica ali zbog toga što smo model obučavali na lošem skupu podataka takav model će biti neupotrebljiv za realan slučaj kada rečenice iz obe klase mogu imati raznovrsnu dužinu.

2. Augmentacija ručno napravljenog skupa podataka parafraziranjem korišćenjem openai api:

   Model koji se koristi za ovakvu augmentaciju je: gpt-4o.

   Konačni upit (eng. prompt) za stvaranje dodatnih rečenica zavisi od klase rečenice i sastoji se iz više delova:

   - Uvodni deo upita koji je isti za obe klase:

   "Parafraziraj sledeću rečenicu tako da može zadržati ili promeniti značenje, ali je važno da priroda rečenice "

   - Sredina upita koja zavisi od klase:

"(rečenica ne predstavlja pretnju fizičkim nasiljem niti nagovaranje na isto) ostane ista. ", za klasu “0”,

"(rečenica predstavlja pretnju fizičkim nasiljem ili nagovaranje na isto) ostane ista. ", za klasu “1”.

- Dodatno pojašnjenje koje je isto za obe klase:

"Molim te da koristiš sinonime, promeniš imena, brojeve, redosled reči, i preformulišeš delove rečenice.\n".

- Deo upita koji predstavlja primer koji zavisi od klase:

"Na primer ako ti dam rečenicu: Idemo na piknik sutra u 6 Stefane. Tvoj odgovor može biti: Nikola, planiramo izlet za sutra uveče u 8!.", za klasu “0”,

"Na primer ako ti dam rečenicu: Prebiću te večeras Marija. Tvoj odgovor može biti: Jovane razbuću ti nos kasnije.", za klasu “1”.

- Krajnji deo upita koji sadrži rečenicu koju treba parafrazirati:

"\nRečenica: ” + originalna rečenica koju treba parafrazirati

Za svaku rečenicu u originalnom ručno napravljenom skupu podataka se dodaju još 6 parafraziranih rečenica (na 3 različite temperature po 2 rečenice)

3. Ručno filtriranje

Kako jezički model za augmentaciju ima specifična etička pravila, ponekat neće želeti da parafrazira rečenicu, takođe ako odgovor bude sadržao dodatan uvod/pojašnjenja umesto samog parafraziranog teksta – takvi odgovori se moraju ručno odbaciti iz skupa podataka.

4. Zašumljavanje skupa podataka

Ideja je model što više prilagoditi realističnim uslovima gde su greške u kucanju na tastaturi normalna pojava – zato se uvodi šum koji treba da imitira greške pri kucanju na tastaturi. Za ovu svrhu sam mapirao svaki karakter sa karakterima koji su mu najbliži na tastaturi, takođe sa karakterima koji nastaju promenom tastature sa srpske latinične na englesku ako se posle promene karakteri razlikuju, i česte greške tipa: (ć,c), (š,s), (ž, z), (z, y), itd.

U skup podataka je dodato dodatnih 3 procenata zašumljenih rečenica. Tako što je uzete 5% postojećih rečenica, iz tog podskupa za svaku rečenicu je za svaki karakter postojala verovatnoća od 3% da će se karakter zameniti sa nasumičnim karakterom iz liste karaktera sa kojom je mapiran. Samo ako dođe do jedne ili više ovakvih zamena onda se ovako zašumljena rečenica dodaje u postojeći skup podataka.

5. Priprema skupa podataka za treniranje

Odnos veličina između trenirajućeg skupa, skupa za validaciju i skupa za testiranje je odabran da bude 80/10/10, dodatno je obezbeđena balansiranost klasa u sva 3 skupa. Da bi se ovi skupovi podataka pripremili za treniranje neophodno je napraviti prilagođenu klasu koja nasleđuje PyTorch-ovu Dataset klasu, koja uzima skup podataka i tokenizer i podatak o maksimalnoj dozvoljenoj veličini ulazne sekvence.

```python
class SentimentDataset(Dataset):
    def __init__(self, dataset, tokenizer, max_length):
        self.texts = [item['text'] for item in dataset]
        self.labels = [item['label'] for item in dataset]
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        label = self.labels[idx]
        encoding = self.tokenizer(
            text,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
        )
        input_ids = encoding['input_ids'].squeeze()
        attention_mask = encoding['attention_mask'].squeeze()
        return {
            'input_ids': input_ids,
            'attention_mask': attention_mask,
            'labels': torch.tensor(label, dtype=torch.long)
        }
```

Slika 1. Priprema skupa podataka za treniranje u google colab-u

2.2 Izbor modela

Jezički pretreniran model korišćen za ovaj rad se naziva: **BERTić** (tačnije checkpoint classla/bcms-bertic), koji je pretreniran MLM (Masked Language Modeling) metodom na 8 milijardi tokena Srpskog, Hrvatskog, Bosanskog i Crnogorskog jezika. Ova metoda omogućava modelu da nauči kontekst reči u rečenici tako što se nasumično maskira određen broj reči u rečenici a zatim model pokušava da pogodi koje su reči bile maskirane na osnovu ostatka rečenice.

Zadaci koje BERT generalno može dobro da obavlja:

- Klasifikacija teksta
- Klasifikacija na nivou tokena (eng. Token Classification): ovde spadaju zadaci poput odgovaranja na pitanja (QA, eng. Question Answering), prepoznavanje entiteta (NER, eng. Named Entity Recognition) i prepoznavanje maskiranog tokena (eng. Masked Token Prediction)
- Klasifikacije para rečenica (eng. Text-Pair Classification) gde spadaju zadaci poput predikcije naredne rečenice (NSP, eng. Next Sentence Prediction) i zaključivanja u prirodnom jeziku (NLI, eng. Natural Language Inference)

Zadaci koje BERT ne može dobro da obavlja:

- Generacija teksta (eng. Text Generation), gde spadaju zadaci poput translacije (eng. Text Translation) i generisanja odgovora (eng. Dialogue Generation)

Dakle BERT je dizajniran za razumevanje teksta ali ne i za generisanje teksta.

Kako je ovaj model pretreniran na korpusu tekstova na srpskom imaće u svom vokabularu tokene koji odgovaraju bitnim delovima reči u srpskom jeziku – na primer najpoznatiji BERT model zasnovan na BERT-base arhikteturi ''bert-base-uncased'', koji je prilagođen engleskom jeziku, uopšte ne uzima u obzir kukice, kvačice kod slova č, ć, š, ž, đ nego će ovi karakteri biti promenjeni pre tokenizacije u c, s i z (Tabele 1-2).

Tako da je BERTić odličan izbor za fino podešavanje (eng. fine tuning) za zadatak klasifikacije na srpskom jeziku.

Bertić je dobijen prilagođavanjem BERT-base arhikteture (110M parametara, 12 enkoderskih blokova, 12 glava pažnje po enkoderskom bloku...), ova arhiktetura će biti u nastavku detaljno objašnjena.

2.3 BERT tokenizacija

Na ulazu u model se nalazi tokenizovani tekst, tokenizator nam pomaže da napravimo vokabular, a vokabular je rečnik koji kao ključeve ima reči/delove reči mapirane sa indeksima tokena (eng. token id), dakle model kao ulaz prima listu indeksa tokena. Tokenizator se pravi pre pretreniranja modela samo na osnovu skupa podataka, kod BERT-a se koristi **Word-Piece tokenizacija**.

Word-Piece algoritam tokenizacije – pravljenje vokabulara:

- Inicijalni rečnik: prvo se prolazi kroz sve reči u skupu podataka i svaka reč se deli na najmanje moguće delove – pojedinačne karaktere, ali uz dodatak posebnog prefiksa (##) za sve karaktera koji se nalaze unutar reči. Na primer ako je prva reč u nekom tekstu u skupu podataka za pravljenje vokabulara ’mačka’ naš vokabular će posle obrađivanje ove reči sadržati 5 tokena: [’m’, ’##a’, ’##č’, ’##k’, ’##a’].
- Spajanje tokena u veće tokene: kada se oformi inicijalni vokabular sa svim karakterima sa i bez prefiksa (##), za svaka dva tokena u vokabularu se računa rezultat kao:

$$rezultat = učestalost\ para/(učestalost\ prvog * učestalost\ drugog\ tokena)$$

  Na osnovu ovog rezultata se određuje koja će se dva tokena spojiti i takvi spojeni dodati u vokabular, dakle u trenutnoj iteraciji spajanja spojiće se samo 2 tokena sa najvećim međusobnim rezultatom, u sledećoj iteraciji se ovaj rezultat ponovo računa za sve varijacije od 2 tokena na izmenjenom vokabularu.
- Prekid algoritma: algoritam se prekida kada se dostigne predefinisani maksimalni broj reči u vokabularu, primera radi kod ‘bert-base-uncased’ modela to je 30522 tokena dok kod ‘classla/bcms-bertic’ modela imamo 32000 tokena u vokabularu.
- Na kraju se u vokabular dodaju specijalni tokeni – koji imaju posebne uloge vezane za procesiranje teksta i treniranje BERT modela. Ovi tokeni su [PAD], [UNK], [CLS], [SEP] i [MASK] (Tabela 3).

Glavne uloge algoritma za tokenizaciju su:

- Efikasno kodiranje teksta: Word-Piece tokenizacija smanjuje veličinu vokabulara u poređenju sa tokenizacijom na nivou celih reči, zadržavaći ključne delove reči, kao što su koren reči, sufiksi i prefiksi.
- Kodiranjem reči van vokabulara (eng. out-of-vokabulary) deljenjem reči na tokene delova reči (eng. subword tokens) omogućava se tokenizacija retkih i pogrešno napisanih reči.

Tabela 1, isečak iz različitih vokabulara (vokabular preslikava token u token_ID)

Token ID	bert-base-uncased token	classla/bcms-bertic token
2139	de	##đa
2140	##l	##sno
2141	born	##zna
2142	united	##liko
2143	film	stra
2144	since	Sa
2145	still	##tni
2146	long	##skog
2147	work	##bu

Tabela 2, upoređivanje tokenizacije pomoću drugačijih vokabulara

reč	bert-base-uncased	classla/bcms-bertic
šumarstvo	['sum', '##ars', '##tv', '##o'], [7680, 11650, 9189, 2080]	['šuma', '##rstvo'], [7434, 11446]
uzvratiti	['u', '##z', '##vr', '##ati', '##ti'], [1057,2480,19716,10450,3775]	['uzvrati', '##ti'], [21020, 1916]
mačka	['mack', '##a'], [11349, 2050]	['mačka'], [22314]

Tabela 3, specijani tokeni za BERT model

Specijalni token	Uloga
Token: [PAD], ID: 0	Token za popunjavanje (eng. padding). Svaka ulazna lista indexa tokena će biti produžena na predefinisanu dužinu dodavanjem nula, ove nule odgovaraju [PAD] tokenu.
Token: [UNK], ID: 1	Nepoznat token. U slučaju da se neka reč ili deo reči ne može tokenizovati iz postejećeg vokabulara tada se ta reč ili deo reči tokenizuje tokenom [UNK].
Token: [CLS], ID: 2	Klasifikacioni token. Zadnji skriveni vektor [CLS] tokena se koristi kao sažeta reprezentacija cele ulazne sekvence za zadatak klasifikacije. Prvi token svakog ulaza je specijalni token [CLS].
Token: [SEP], ID: 3	Separcioni token. Koristi se za razdvajanje dve sekvence u zadacima poput predikcije sledeće rečenice (NSP, eng. Next Sentence Prediction).
Token: [MASK], ID: 4	Maskirani token. Koristi se u maskiranom modelovanju reči (MLM), kada se model trenira da predviđa maskirane reči u rečenici.

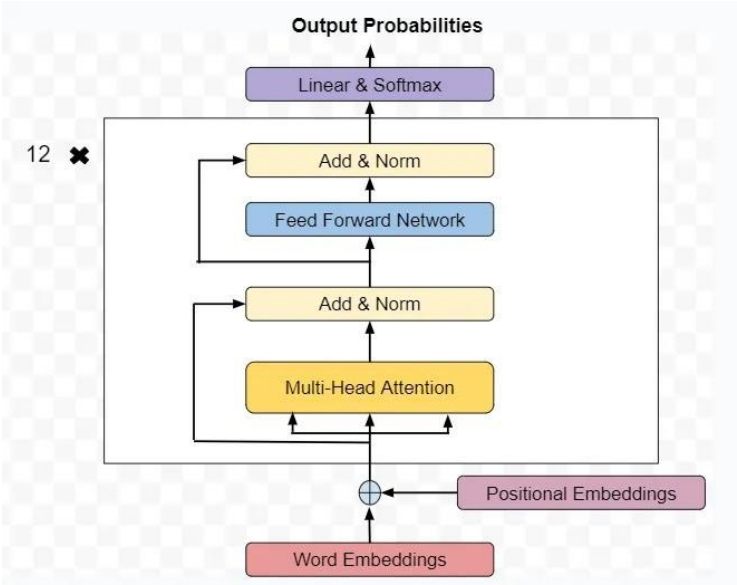
Važno je naglasiti da su model i njemu odgovarajući tokenizator nerazdvojni, pošto se model pretrenirava koristeći tokene iz odredjenog vokabulara (Slika 1).

```
1 from transformers import AutoTokenizer, BertForSequenceClassification
2 import torch
3
4 bertic_model_checkpoint = "classla/bcms-bertic"
5 bertic_model = BertForSequenceClassification.from_pretrained(bertic_model_checkpoint, num_labels=2)
6 bertic_tokenizer = AutoTokenizer.from_pretrained(bertic_model_checkpoint)
```

Slika 1. Očitavanje modela i tokenizatora u google colab-u, dakle tokenizator i model će biti učitani za isti checkpoint

2.4 BERT arhiktetura

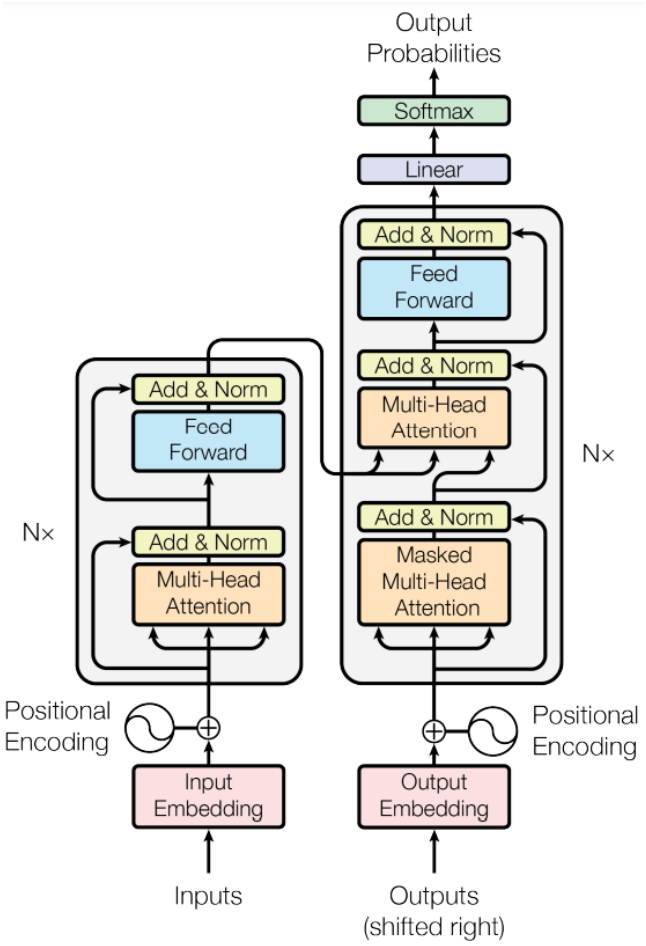
Kada govorimo o BERT modelu, obično se misli na pretrenirani BERT model - dakle sve težine modela su već podešene za vreme treniraranja kroz zadatke maskiranog modelovanja i predikcije sledeće rečenice na velikom skupu podataka. Kada govorimo o BERT arhikteturi misli se na specifičnu strukturu modela zasnovanu na transformer arhikteturi, pri čemu BERT koristi samo enkoderski deo transformera [1] (Slike 2-3). Najpoznatije BERT varijante su:



Slika 2. Arhiktetura BERT-base modela, vidimo da je broj enkoderskih blokova naslaganih jedan na drugi 12 [3].

- BERT-large: arhiktetura sa 24 enkoderskih slojeva, oko 340 miliona parametara,
- BERT-base: arhiktetura sa 12 enkoderskih slojeva, oko 110 miliona parametara,
- DistilBERT: arhiktetura sa 6 enkoderskih slojeva, oko 66 miliona parametara.

U ovom radu će biti razmatrana BERT-base arhiktetura na primeru pretreniranog modela classla/bcms-bertic. Pre enkoderskih blokova se nalaze slojevi ulazne reprezentacije (eng. Input Embedding), tu spadaju: sloj ulazne reprezentacije reči (eng. Word Embedding), sloj pozicionog kodiranja (eng. Possitional Embeding) i sloj segmentnog kodiranja (eng. Segment Embedding), kao i normalizacija sloja (eng. Layer Normalization).



Slika 3. Osnovna transformer arhiktetura, enkoderski deo je prikazan na levoj strani, dekoderski deo transformera na desnoj, Nx pretstavlja broj ponavljanja celog bloka enkodera ili dekodera [1].

2.5 BERT arhiktetura – Embedding slojevi

Vektorska reprezentacija reči (eng. Word Embedding) je matrica koja ima redova onoliko koliko ima tokena u vokabularu, dakle svaki token iz teksta model vidi kao vektor sa određenim brojem karakteristika (obično stotinama dimenzija, u našem primeru sa BERT-base arhikteturom to je 768 dimenzionalni vektor) gde svaka dimenzija tog vektora nosi neku informaciju o značenju tokena i sličnosti sa drugim tokenima. U slučaju classla/bcms-bertic modela imamo vokabular sa 32000 tokena što daje sloj Word Embedding-a od približno 24.5 miliona parameta, što je oko 22% od ukupnog broja parametara modela.

```
Sloj: bert.embeddings.word_embeddings.weight | Veličina: torch.Size([32000, 768])
Sloj: bert.embeddings.position_embeddings.weight | Veličina: torch.Size([512, 768])
Sloj: bert.embeddings.token_type_embeddings.weight | Veličina: torch.Size([2, 768])
Sloj: bert.embeddings.LayerNorm.weight | Veličina: torch.Size([768])
Sloj: bert.embeddings.LayerNorm.bias | Veličina: torch.Size([768])
```

Slika 4. Arhiktetura BERT-base, checkpoint: classla/bcms-bertic, Embedding slojevi i sloj za normalizaciju

Poziciono kodiranje (eng. Possitional embedding) pruža modelu informaciju o redosledu tokena u ulaznaj sekvenci, ovo je izuzetno važno jer BERT (kao i ostali modeli bazirani na transformer arhikteturi) nemaju ugrađen mehanizam za razumevanja pozicije tokena, dok na primer rekurentne neuralne mreže obrađuju ulaznu sekvencu token po token, samim tim stiču razumevanje koja reč dolazi pre koje.

U originalnom radu „Attention is all you need“ iz 2017 [1]. poziciono kodiranje transformer modela se definiše kao:

$$PE(pos, 2i) = \sin\left(\frac{pos}{100000^{\frac{2i}{d_{model}}}}\right)$$

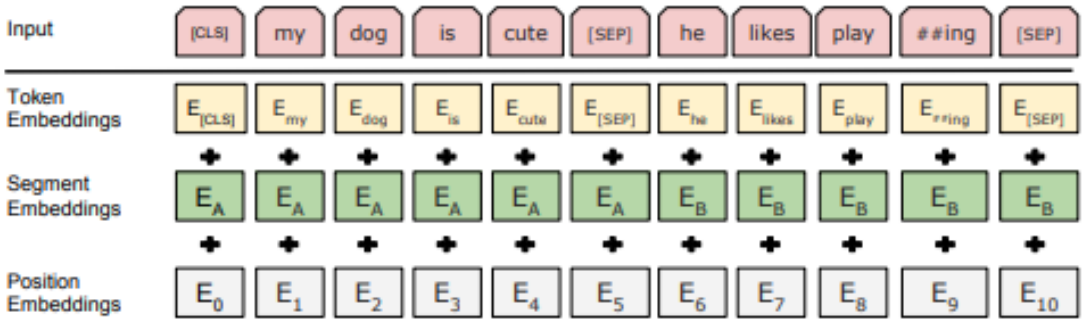
$$PE(pos, 2i+1) = \cos\left(\frac{pos}{100000^{\frac{2i}{d_{model}}}}\right)$$

Gde su promenljive:

- pos - pozicija tokena u sekvenci
- i - indeks u vektoru Word Embedding-a
- d_{model} - veličina vektora Word Embedding-a

Kod BERT-a (kao i kod ostalih modela zasnovanih na originalnom transformer modelu poput GPT modela) sloj za poziciono kodiranje se uči za vreme pretreniranja, zajedno sa slojem za reprezentaciju reči (eng. Word Embedding) i slojem za segmentaciono kodiranje. Svi ovi slojevi se mogu dodatno menjati za vreme finog podešavanja (eng. Fine Tuning) ako to dozvolimo. U ovom radu su svi Embedding slojevi zamrznuti tokom finog podešavanja.

Segmentaciono kodiranje (eng. Segment Embedding) služi kako bi smo naznačili modelu koji token pripada prvoj sekvenci a koji token drugoj u zadacima poput predviđanje sledeće rečenice, odgovaranja na pitanja i zaključivanja u prirodnom jeziku. Dakle sloj segmentacionog kodiranja je matrica koja ima dve vrste i broj kolona koliko ima dimenzija vektor reprezentacije reči. Ako se trenutni token nalazi pre prvog tokena za separaciju [SEP], njegovoj vektorskoj reprezentaciji će biti dodat prvi vektor/prva vrsta te matrice, ako se nalazi posle tokena za separaciju biće mu dodat drugi vektor/druga vrsta te matrice (Slika 5).



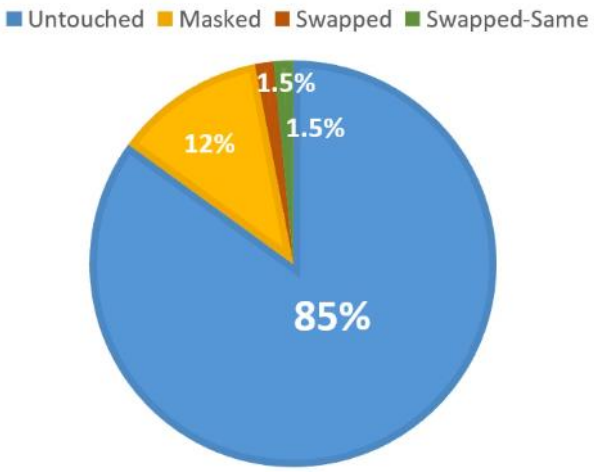
Slika 5. Bertova ulazna reprezentacija [4].

2.6 BERT proces pretreniranja

Da bi se trenirala duboka bidirekciona reprezentacija, neki procenat ulaznih tokena se nasumično maskira, a zatim se predviđaju ti maskirani tokeni. Ovaj postupak nazivamo **maskirano jezičko modelovanje** (eng. Masked Language Modeling). Može se maskirati 15% svih WordPiece tokena u svakoj sekvenci nasimično. Nedostatak ovogo je što postoji neslaganje između pretreniranja i finog podešavanja (token [Mask] se ne pojavljuje tokom finog podešavanja), tako da se zapravo izabere 15% nasumčnih tokena u sekvenci i svaki izabrani token se:

- Zamenjuje [MASK] tokenom u 80% slučajeva nasumično
- Zamenjuje se nasumičnim tokenom iz celokupnog vokabulara u 10% slučaja
- Ostaje nepromenjen

Dakle u zadatku maskiranog jezičkog modelovanja model treba da predvidi koji token je bio maskiran (označen) od svih tokena u vokabularu. Na primer model na izlazu vraća verovatnoće za svih 30522 tokena za bert-base-uncased ili 32000 verovatnoća za classla/bcms-bertic. Ovo se omogućava dodavanjem klasifikacionog sloja na sve skrivene vektore osih za token [CLS]. Ovaj klasifikator je jedna potpuno povezana neuralna mreza sa brojem neurona u izlaznom sloju jednakom broju tokena u vokabularu.



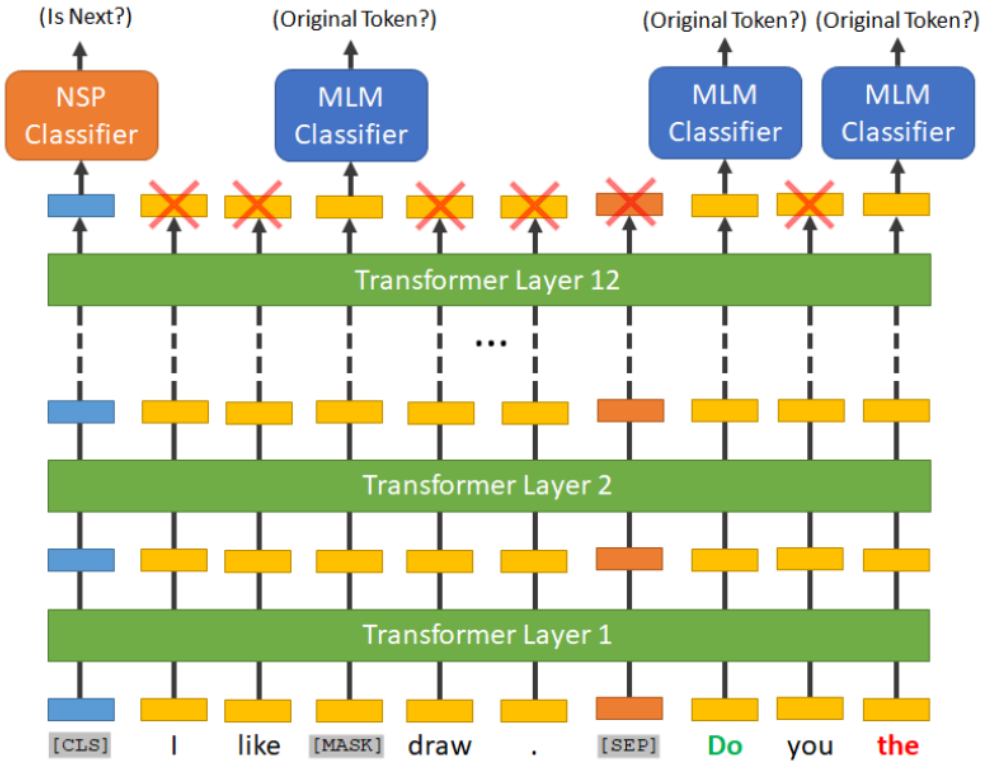
Slika 6. Odnos označenih tokena koji su zamenjeni tokenom [MASK], zamenjeni ili nepromenjeni, dakle maskirani token neće uvek biti zamenjen tokenom [MASK] [2].

Osim maskiranog jezičkog modelovanja model se trenira na zadatku **predikcije naredne rečenice** (NSP, eng. Next Sentence Prediction), ovo je značajno za zadatke poput Odgovaranja na pitanja (QA, eng. Question Answering) i Zaključivanja u prirodnom jeziku (NLI, eng. Natural Language Inference), pošto se ovakvi zadaci zasnivaju na razumevanju odnosa izmedju dve rečenice što nije direktno obuhvaćeno maskiranim jezičkim modelovanjem. Dakle ovaj zadatak je zadatak binarne klasifikacije gde model treba da za date dve rečenice proceni da li je druga rečenica nastavak prve ili ne.

Posle svakog enkoderskog bloka dobijamo skrivene vektore za svaki token. Sve prikazane transformacije na Slici 5 su na nivou jednog tokena, svi tokeni se procesuiraju paralelno, dakle stvarni izlaz iz enkoderskog bloka je matrica dimenzije 512*768 za BERT-base arhikteturu, odnosno za svih 512 tokena dobijamo po skriveni vektor dimenzije 768. Za zadatak klasifikacije se koristi isključivo zadnji skriveni vektor za specijalni token [CLS], kako ovaj vektor ima dimenziju 768 mora mu se dodati još jedan potpuno povezan sloj sa brojem neurona koliko klasa imamo u zadatku, kako u zadatku predikcije naredne rečenice imamo 2 klase to se moraju dodati 2 neurona prilikom pretreniranja.

Maskirano jezičko modelovanje i predikcija naredne rečenice se vrše paralelno (Slika 7). Ukupan gubitak (eng. Loss), se računa kao zbir gubitka za predikciju naredne rečenice i ukupnog zbira gubitaka za sve maskirane tokene:

$$Total\ Loss = MLM\ Loss + \ NSP\ Loss$$



Slika 7. Proces pretreniranja, za ulaznu sekvenu: "I like to draw. [SEP] Do you?" [2], ''NSP Classifier'' je binarni klasifikator, ''MLM Classifier'' radi sa onoliko klasa koliko ima tokena u vokabularu.

2.7 Enkoderski blok

Enkoderski blok se sastoji od dva ključna dela: višeglavog mehanizma pažnje (eng. multi-head self attention) i potpuno povezane neuralne mreže (FFNN, eng. Fully Connected Neural Network). Svaki enkoderski blok koristi rezidualne veze (eng. residual connections) oko mehanizma pažnje i oko potpuno povezane neuralne mreže, kako bi očuvao informacije iz prethodnih slojeva u dubljim slojevima i kako bi treniranje bilo stabilnije (Slika 5).

Mehanizam pažnje (eng. attention) je ključni koncept u modernim modelima za obradu prirodnog jezika, posebno velikim jezičkim modelima (LLM, eng. Large Language models). Najveći značaj mehanizma pažnje ogleda se u tome što omogućava paralelizaciju svih operacija za svaki token u sekvenci posebno, zajedno sa razumevanjem konteksta između tokena.

```
Sloj: bert.encoder.layer.0.attention.self.query.weight | Veličina: torch.Size([768, 768])
Sloj: bert.encoder.layer.0.attention.self.query.bias | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.attention.self.key.weight | Veličina: torch.Size([768, 768])
Sloj: bert.encoder.layer.0.attention.self.key.bias | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.attention.self.value.weight | Veličina: torch.Size([768, 768])
Sloj: bert.encoder.layer.0.attention.self.value.bias | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.attention.output.dense.weight | Veličina: torch.Size([768, 768])
Sloj: bert.encoder.layer.0.attention.output.dense.bias | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.attention.output.LayerNorm.weight | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.attention.output.LayerNorm.bias | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.intermediate.dense.weight | Veličina: torch.Size([3072, 768])
Sloj: bert.encoder.layer.0.intermediate.dense.bias | Veličina: torch.Size([3072])
Sloj: bert.encoder.layer.0.output.dense.weight | Veličina: torch.Size([768, 3072])
Sloj: bert.encoder.layer.0.output.dense.bias | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.output.LayerNorm.weight | Veličina: torch.Size([768])
Sloj: bert.encoder.layer.0.output.LayerNorm.bias | Veličina: torch.Size([768])
```

Slika 8. Jedan encoderski blok, BERT-base arhiktetura

U kontekstu BERT modela podrazumevano govorimo o mehanizmu “self-attention” kod koga svaki token ulazne sekvence “obraća pažnju” (Slika 7) na sve ostale tokene iz iste ulazne sekvence kao i na samog sebe (ovo se razlikuje od mehanizma ’’encoder-decoder attention’’ koji se koristi u modelima koji imaju i enkoder i dekoder kao što je slučaj sa modelima za prevođenje).

Da bi se razumeo mehanizam višeglave (eng. multi-head self attention) pažnje neophodno je razumeti mehanizam pažnje koji se u radu “Attention is all you need” navodi kao “scaled dot-product attention”, kako je mehanizam višeglave pažnje proširenje ovog mehanizma.

Skalirani mehanizam pažnje sa skalarnim proizvodom (eng. scaled dot-product attention):

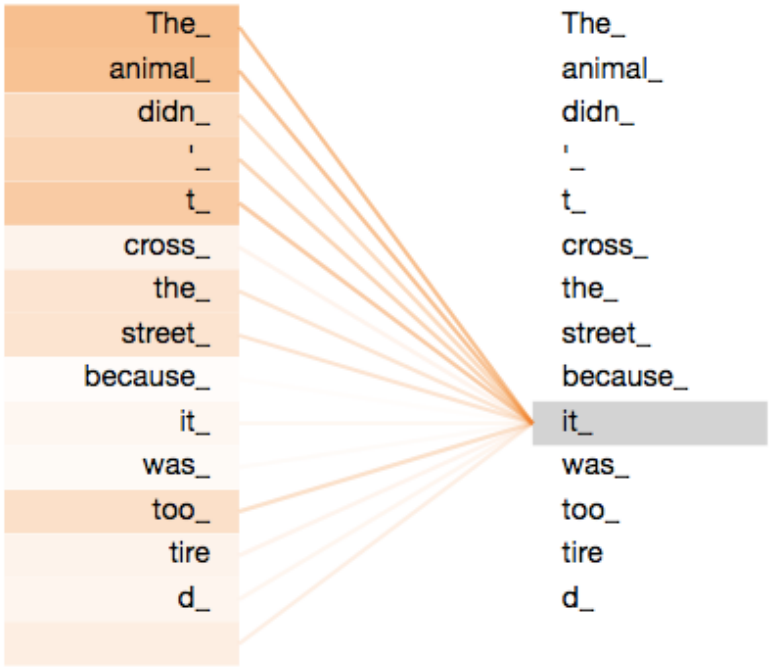
Ulaz u sloj sa mehanizmom pažnje ćemo označiti sa X, ovo je dakle vektor dimenzija 512*768, sam model sadrži fiksne matrice $W_q, W_k, W_{v,} b_q, b_k \; i \; b_v$. Ove matrice služe su naučene za vreme pretreniranja i služe za dobijanje ključnih matrica u sloju sa mehanizmom pažnje, matrica Q (query), K (Key) i matrice V (value):

$$Q = X * W_q + \; b_q, \;\; K = X * W_k + b_k, \; V = \; X * W_v + b_v$$

Tada je izlaz iz ovog sloja računa kao:

$$Z = Attention(Q, K, V) = softmax\left(\frac{Q * K^T}{\sqrt{d_k}}\right) * V$$

gde je d_k dimenzionalnost kvadratne matrice K (u našem slučaju sa $K_{768*768}$, $d_k = 786$).



Slika 9. Intuicija iza mehanizma ''self-attention'', da bi se enkodirao token ''it'' mehanizam pažnje se fokusira na tokene ''The'' i ''animal''. [5]

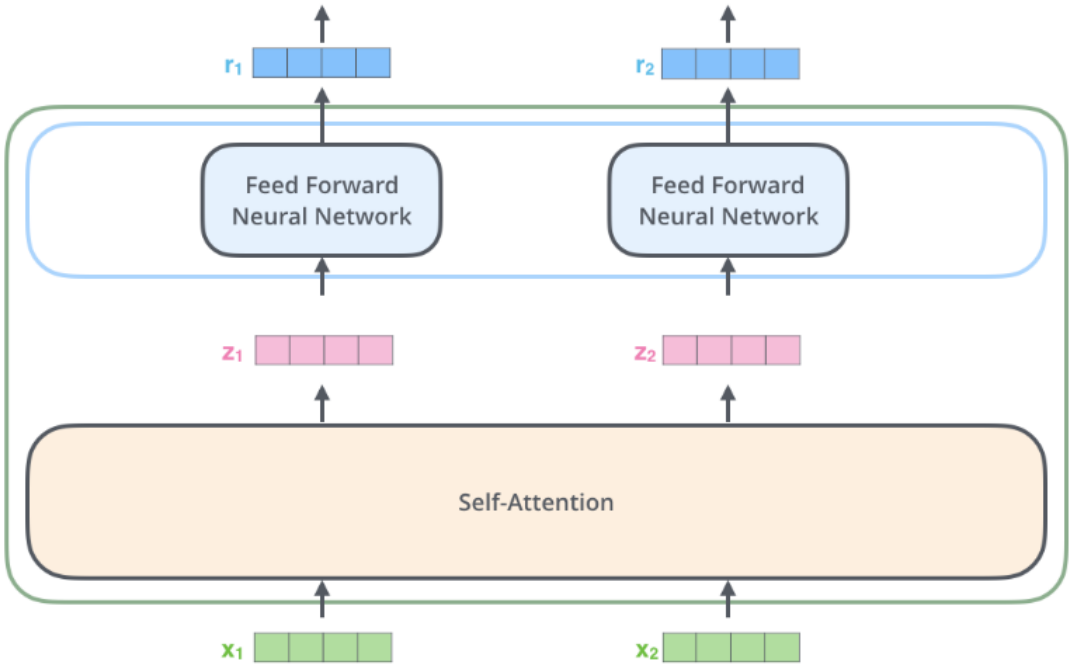
Dakle enkodiranje koje se vršilo u slojevima ulazne reprezentacije se sada nastavlja u sloju sa mehanizmom pažnje. Za razliku od slojeva ulazne reprezentacije gde svako enkodiranje ulaznog tokena zavisi samo od tog tokena, u mehanizmu pažnje enkodiranje tog tokena će zavisiti i od svih ostalih tokena u ulaznoj sekvenci. Za token kome odgovara ulazna reprezentacija x_i i ''Query'' vektor q_i, njegovo dalje enkodiranje će se više osloniti na ''Value'' vrednost v_j koja odgovara tokenu sa ulaznom reprezentacijom x_j, ako je skalarni proizvod $q_i \circ k_j$ veći u poređenju sa skalarnim proizvodima vektora q_i i drugih vektora $k_{l \neq j}$.

Mehanizam višeglave pažnje (eng. multi-head self-attention), kod BERT-base arhikteture u svakom enkoderskom bloku imamo po 12 glava. Svaka glava ima svoje sopstvene matrice $W_q, W_k, W_v, b_q, b_k \; i \; b_v$, dakle u svih 12 blokova enkodera imamo po 12 različitih skupova matrice za dobijanje matrica Q, K i V. Sada ove matrice imaju drugačije dimenzije:

$$W_{q,768*64}, W_{k,768*64}, W_{v,768*64}, b_{q,64*1}, b_{k,64*1}, b_{v,64*1}$$

pa će matrice Q, K i V u svakoj glavi imati dimenzije:

$$Q_{\,512*64,}\, K_{\,512*64}, V_{\,512*64}$$



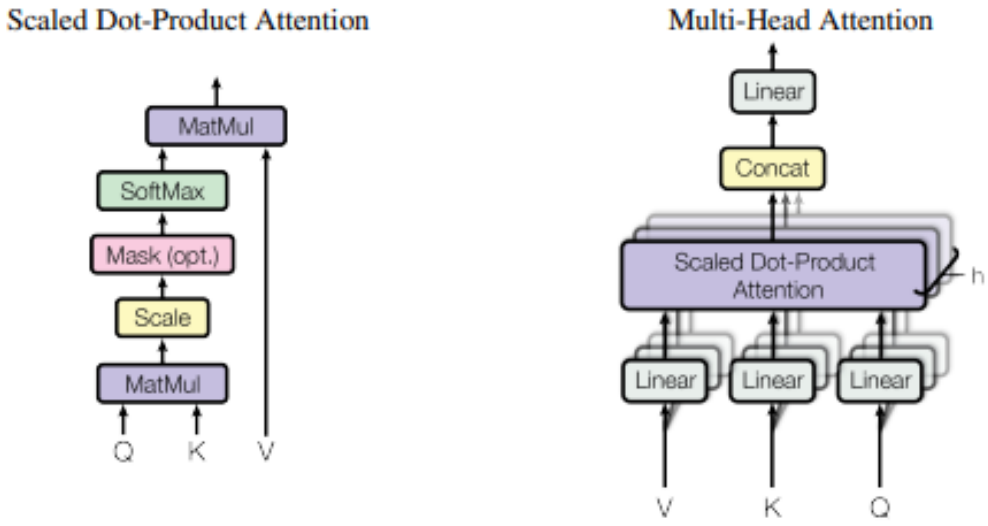
Slika 10. Svaki red u matrici ulazne reprezentacije X je ulazna reprezentacija specifičnog tokena u ulaznoj sekvenci x1, x2, ... x512, svaka ova reprezentacija ima odgovarajući izlaz iz sloja sa mehanizmom pažnje z1, z2, ... z512 koji zajedno čine matricu Z. [5]

Kada se izvrši funkija pažnje za sve glave, nad dobijenim matricama se izvrsava konkatenacija kako bi se dimenzija izlaza iz sloja sa mehanizmom višeglave pažnje vratila na dimenzije ulaznog vektora X (512*768).

Nakon ovoga imamo linearnu transformaciju (768 * 768) nad konkateniranom matricom, ova matrica se naziva ''output projection'' i obeležava se sa W^o.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_{12})^* W^o$$

Iako je sav fokus na mehanizmu pažnje ne treba zaboraviti **potpuno povezanu neuralnu mrežu** u okviru enkoderskog bloka. Zapravo ova mreža zauzima oko 2/3 ukupnog broja parametara svakog enkoderskog bloka. Ova mreža u svakom enkoderskom bloku se sastoji od jednog skrivenog sloja od 3072 neurona (4*768, gde je 768 dimenzionalnost ulaza), i izlaznog sloja od 768 neurona (Tabela 4).



Slika 11. Grafički prikaz skaliranog mehanizma pažnje sa skalarnim proizvodom i višeglave pažnje kao proširenje ovog mehanizma [1]

Tabela 4, odnos broja parametara između slojeva jednog enkoderskog bloka

sloj	Broj parametara	Ukupan udeo u enkoderskom bloku
Multi-head self-attention	oko 1.8M	oko 27%
FFNN	oko 4.7M	oko 73%
Ukupno	oko 6.5M	100%

2.8 Izlaz iz poslednjeg enkoderskog bloka prilagođen klasifikaciji

Kako se u ovom radu razmatra zadatak binarne klasifikacije to je broj neurona u poslednjem sloju 2. Implementacija BERT-a u Hugging Face Transformer biblioteci dodatno uključuje pooler između zadnjeg enkoderskog bloka i 2 neurona za binarnu klasifikaciju (Slika 6). Pooler je zapravo potpuno povezana neuralna mreža sa istim dimenzijama na ulazu i izlazu i koristi tanh aktivaciju, ovaj sloj se dodaje kao dodatna priprema za klasifikaciju.

```
Sloj: bert.pooler.dense.weight | Veličina: torch.Size([768, 768])
Sloj: bert.pooler.dense.bias | Veličina: torch.Size([768])
Sloj: classifier.weight | Veličina: torch.Size([2, 768])
Sloj: classifier.bias | Veličina: torch.Size([2])
```

Slika 12. Izlaz Bert-base prilagođen binarnoj klasifikaciji

2.9 Metodologija finog podešavanja jezičkog modela

Model je na početku kvantizovan. Kvantizacija modela je metoda koja se koristi iz više razloga:

- Smanjenje memorijskog otiska – sve težine u modelu su konvertovane iz 32-bitne preciznosti u 4-bitne preciznosti.
- Ubranje treninga
- Kvantizacija kao metoda regularizacije – smanjivanjem preciznosti težina modela se uvodi šum kvantizacije, ovime se blago poboljšava mogućnost modela da bolje generalizuje na neviđenim podacima

Model se fino podešava metodom LoRA (Low-Rank Adaptation of Large Language Models):

- Prvo se uvode dve dodatne matrice niskih dimenzija, u specifične slojeve modela
- Ostatak modela se zamrzava
- Samo parametri koje zovemo LoRA matrice se treniraju, ovime se štedi vreme treniranja uz relativno mali pad performansi u odnosu na potpuno fino podešavanje (eng. full fine tuning)

Recimo da se linearni sloj modela može matematički opisati kao:

$$y = Wx + b$$

gde je W težinska matrica dimenzija: $d_{out} \times d_{in}$, nakon dodavanja LoRA matrica imaćemo:

$$y = (W + BA)x + b$$

pri čemu je matrica B dimenzija: $d_{out} \times r$, i matrica A dimenzija: $r \times d_{in}$, r se naziva LoRA rang, sa ovim parametrom se podešava broj parametara koji će se kasnije trenirati, odnosno što je r manje to će manje parametara biti trenirano i obrnuto. Dakle LoRA tehnikom se samo

treniraju dodatne matrice B i A, dok težinska matrica W ostaje zamrznuta, i ovim postupkom se drastično ubrzava trening.

Da bi se dodatno kontrolisao uticaj LoRA tehnike dodaje se parametar α koji se naziva: LoRA scalirajući factor. Kako se pokazije da je za malo r potreban veći skalirajući faktor i obrnuto – skalirajući faktor se skalira sa rangom r. Konačan izraz za LoRA tehniku izgleda ovako:

$$y = \left(W + \frac{\alpha}{r} BA\right) x + b$$

LoRA matrice se podrazumevano dodaju na matrice Q (Query) i V (Value) u slojevima mehanizma pažnje u okviru enkodera, naravno moguće ih je dodati na druga mesta ali u ovom radu su dodate upravo na ove podrazumevane matrice.

Ovo je važno razumeti jer je sledeći korak finog podešavanja upravo analiza koji je najbolji odnos skalirajućeg faktora i ranga LoRA matrica za naš kvantizovani model.

Takođe LoRA matrice se ne moraju dodati u sve slojeve modela, nego ih je moguće dodati u samo jedan ili nekoliko slojeva. Na ovaj način se dodatno balansira brzina treninga/količina potrebnih resursa i performanse modela.

Zadnji korak u finom podešavanju se odnosi na optimizaciju standardnih hiperparametara kao što su brzina učenja (eng. learning rate), broj epoha, i veličina batch-a, kako bi se dodatno povećale performanse modela.

3 REZULTATI

U ovom poglavlju pokazujemo finalne rezultate svih eksperimenata koji su opisani u prethodnom poglavlju.

3.1 Poređivanje kvantizacije sa drugim metodama regularizacije

U ovom eksperimentu su modelu dodate LoRA matrice na sedmom i dvanaestom sloju mehanizma pažnje (eng. attention layer), model se prvo obučava bez ikakve regularizacije, potom se ispočetka obučava ali se na LoRA matrice primenjuje dropout metoda, potom se model prvo kvantizuje pa obučava bez regularizacije. Potom se upoređuje efekat regularizacije između ova 3 načina podešavanja modela:

Tabela 5, Poređenje efekata Kvantizacije i efekata regularizacije

Metoda	Train f1_score	Validation f1_score	Test f1_score
Model Bez regularizacije i bez kvantizacije	0.827295	0.787330	0.779816
Lora dropout = 0.1	0.812296	0.774774	0.8
Kvantizacija	0.811128	0.780082	0.811475

Dakle može se primetiti da kvantizacija zaista približava f1 vrednosti na trening skupu, validacionom skupu i testirajućem skupu – u odnosu na model bez kvantizacije i bez regularizacije kod koga je razlika između f1 vrednosti na trening i validacionom/testirajućem skupu najveća.

3.2 Analiza uticaja broja dodatih LoRA slojeva na performanse modela

Kako se LoRA matrice ne moraju dodati na sve slojeve sa mehanizmom pažnje (eng, attention layer), nije loše ispitati na koje ih je slojeve najefikanije dodati. Najefikasnije u ovom slučaju znači da ih dodamo na što manje slojeva a što manje pokvarimo performanse u odnosu na slučaj kada bi ih dodali na sve slojeve. U ovom ekperimentu se koristi kvantizovani model, takođe ostali hiperparametri su fiksirani poput: lora_rank = 8, lora_scaling_factor = 32, lora_dropout = 0.1, learning_rate = 0.001 itd.

Tabela 6, Na koliko slojeva je najbolje primeniti QLoRA tehniku za naš model

Broj slojeva u koje dodajemo LoRA matrice	Train f1_score	Validation f1_score	Test f1_score
1	0.796374	0.786920	0.797410
2	0.822685	0.791162	0.797319
3	0.834480	0.782460	0.788317
4	0.830192	0.795648	0.797229
6	0.863524	0.808681	0.836097
12 (ukupan broj slojeva)	0.935970	0.856301	0.866177

Kako performanse modela treniranog sa LoRA tehnikom nisu sjajne ni u slučaju dodavanja LoRA matrica u sve slojeve sa mehanizmom pažnje, znači da je neophodno u nastavku koristiti model sa LoRA matricama u svim slojevima.

Sada otprilike znamo da nam fali parametara za treniranje da bi se postigli dobri rezultati. Povećanje broja trenirajućih parametara se vrši povećanjem LoRA ranga r, takođe važan je odnos LoRA skalirajućeg faktora i LoRA ranga $\frac{\alpha}{r}$, kako se uticaj LoRA parametara izbalansirao.

3.3 Podešavanje LoRA ranga i skalirajućeg faktora

Dakle fiksiran je broj slojeva na koje se dodaju LoRA matrice – na sve slojeve se dodaju. Sada je potrebno povećavati LoRA rang do prihvatljivih rezultata i za njega naći ideali skalirajući faktor.

Tabela 7, Traženje najboljih vrednosti za LoRA rank i skalirajući faktor

α/r	Train f1_score	Validation f1_score	Test f1_score
$r = 4, \alpha = 8$	0.853616	0.808637	0.801355
$r = 4, \alpha = 12$	0.892160	0.808463	0.870689
$r = 4, \alpha = 16$	0.903837	0.838981	0.853175
$r = 4, \alpha = 20$	0.913497	0.852073	0.857629
$r = 8, \alpha = 16$	0.909652	0.834770	0.853448
$r = 8, \alpha = 24$	0.925089	0.843288	0.853175
$r = 8, \alpha = 32$	0.935970	0.856301	0.866177
$r = 8, \alpha = 40$	0.920361	0.851262	0.848504
$r = 16, \alpha = 32$	0.935023	0.8650924	0.844781
$r = 16, \alpha = 48$	0.950966	0.856388	0.879301
$r = 16, \alpha = 64$	0.942252	0.847477	0.861904
$r = 16, \alpha = 80$	0.946604	0.856061	0.866257

Iz rezultata se primećuje da se najbolji rezultati dobijaju za $\frac{\alpha}{r} = 3$, tako da se u nastavku fiksiramo na: $r = 16, \alpha = 48$.

3.4 Podešavanje ostalih hiperparametara

Pretragom za najboljim hiperparametrima se dobija:

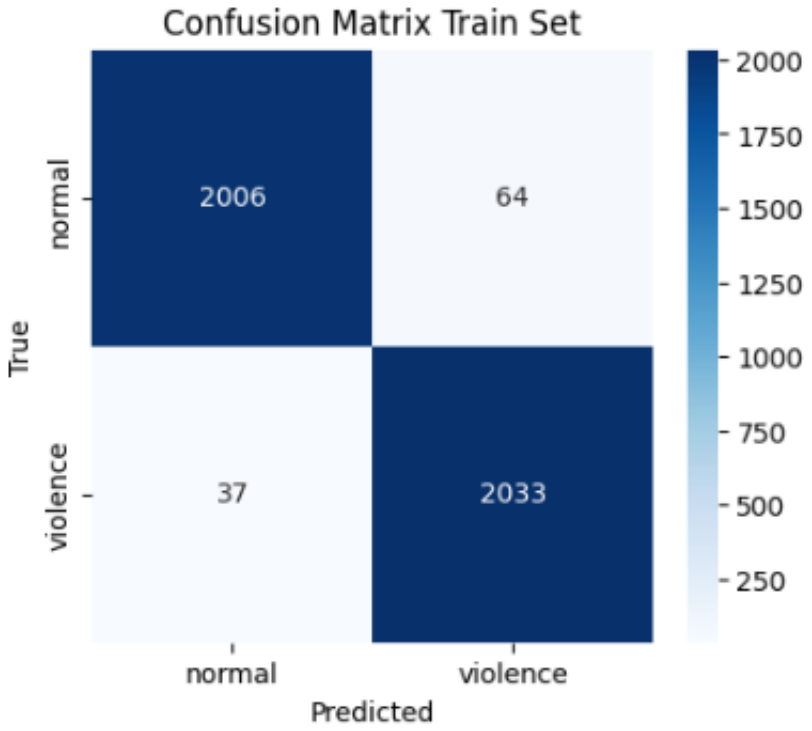
- best_learning_rate = 0.0004857
- best_batch_size = 16
- best_weight_decay = 0.00845265

Model se potom ponovo trenira sa većim brojem epoha i sa najboljim hiperparametrima:

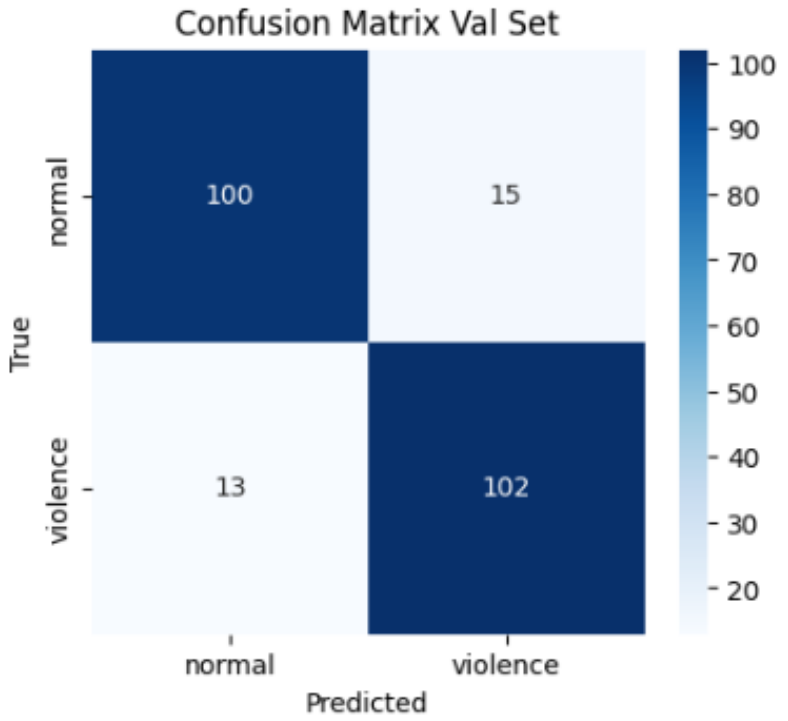
Tabela 8, Finalna evaluacija

Skup podataka	f1_score	accuracy
Trening skup	0.975761	0.975603
Validacioni skup	0.879310	0.878260
Test Skup	0.888888	0.887931

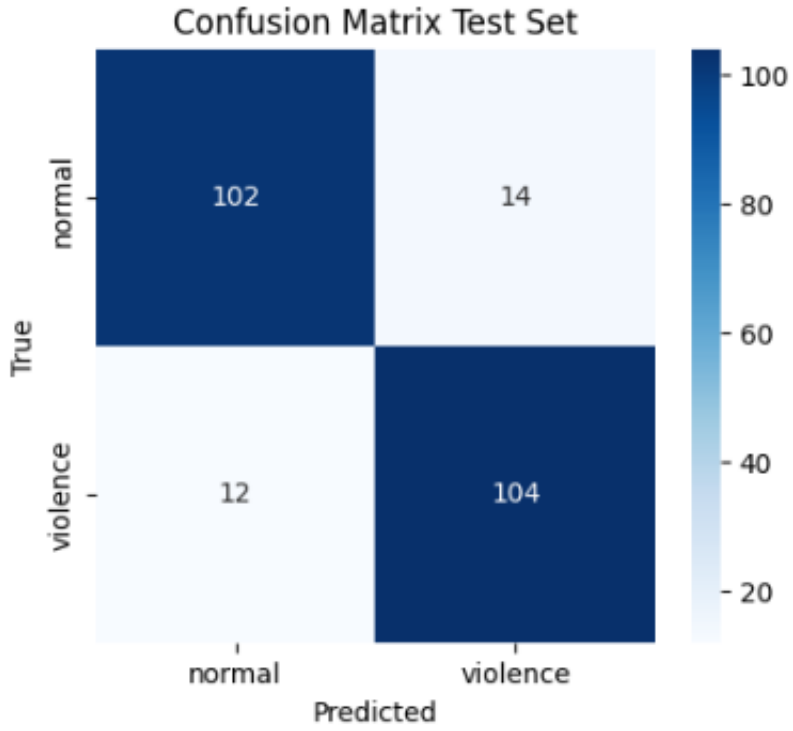
Izgled konfuzionih matrica na ova 3 skupa:



Slika 13. Finalna evaluacija na trening skupu



Slika 14. Finalna evaluacija na validacionom skupu



Slika 15. Finalna evaluacija na testirajućem skupu

4 DISKUSIJA

Očekivanja navedena u uvodu su delimično ispunjena.

Ovim radom je pokazano da se sa samo malo računarskih resursa može obučiti jezički model – u ovom slučaju model koji ima 110 miliona parametara je obučen na zadatku klasifikacije koristeći svega 129 hiljada parametara za treniranje (parametri dodatih LoRA matrica), i pritom su parametri modela bili 4-bitno kvantizovani – dakle preciznost težina u modelu je 8 puta manja nego što bi inače bila – samim tim su memorijski resursi za skladištenje ovih težina 8 puta manji.

Ovaj rad se fokusira na fino podešavanje jednog modela – u praksi to nikada nije slučaj, prvi je korak uvek probati mnoge modele sa LoRA i QloRA tehnikama pa onda na modelu koji se najbolje pokazao bi se nastavilo komplikovanije prilagođavanje. Na osnovu vokabulara drugih BERT modela sam zaključio da je model koji sam koristio u ovom radu za fino podešavanje (classla/bcms-bertic) jedina opcija pošto vokabulari drugih modela niti imaju karaktere č, ć, š, đ, ž u okviru tokena, niti tokeni u vokabularu odgovaraju smislenim delovima reči srpskog jezika.

Glavni nedostatak rada jeste nedovoljno raznolik skup podataka – u kojem ima svega 5000 primera teksova. Za obučavanje jezičkog modela je ovo generalno jako malo – pogotovu zato što u jeziku postoji ogroman broj reči, specifičnih izraza, izreka, poslovica koje su specifične, i ako model ne vidi dovoljno primera neće moći dobro da razume zadatak za koji se obučava. Problem kod pravljenja skupa podataka je što je to previše naporan i spor proces, čak i da se ima gotov skup podataka – mora se kroz njega proći ručno - ispraviti greške pri anotaciji. Ovo je posebno problem za manje poznate jezike poput srpskog za koje skupovi podataka se moraju ručno praviti ili automatski translacijom sa drugog jezika itd...

Problem detekcije pretnji fizičkim nasiljem se vidi na primeru para rečenica: „Stavi prst na čelo!“ i „Stavi prst u struju!“ – ovakvih parova primera ima dosta i pokazuju da model mora da bude treniran na ogromnom broju podataka kako bi pokazao inzvanredne rezultate.

5.1 Analiza eksperimentalnih rezultata

Rečenice koje se nisu tačno klasifikovale u finalnom modelu posle podešavanja svih hiperparametara (Tabela 9):

Tabela 9, Prikaz nekoliko pogrešno klasifikovanih podataka iz testirajućeg skupa

Rečenica	Pogrešno klasifikovana kao klasa:	Moguće objašnjenje
Bolje se ne sukobljavaj s njim, deo je grupe i mogli bi da te povrede.	1	Rečenica jeste teža za klasifikaciju, nije baš najjasnije da li je upućena pretnja ili ne.
Moj ti je savet da se ubiješ.	0	Pretpostavljam da je nedovaljan broj ovakvih primera u skupu za treniranje, gde se osoba nagovara na somoubistvo... Mada i ovo je teži primer...
Naći ću te čim te vidim na mapi, baš da isprobam novi item.	1	Ova rečenica pripada posebnom delu skupa podataka koji se naziva ''Geming Contex'', model treba da prepozna da se radi o nekoj igrici i da onda to nije pretnja fizičkim nasiljem...
Ubij se, ne vredi ti ništa što radiš.	0	Isto potencialno objašnjenje, potrebno bi bilo više primera nagovaranja na samoubistvo u trening skupu...
Izbrisaću ti nalog ovaj onda.	1	Nije pretnja fizičkim nasiljem tako da ne bi trebalo da se klasifikuje kao pretnja...
Ovako: Uhvatićemo je i prebaciti preko granice gde ćemo je prodati	0	Ovo deluje kao lakši primer, očigledno se misli na čoveka koji će biti bačen i prodat, ali model se ipak nije snašao...
Bog vidi sve i kazniće one koji su zli.	1	Ovo pripada posebnom delu skupa podataka koji je fokusiran na religijski kontekst, definitivno ima premalo primera ovakvog tipa, pa se model nije snašao.

Vidimo da trenutne performanse modela direktno zavise od broja trenirajućih parametara.

Ovo je dobar znak, generalno LoRA je takva tehnika da uvek znamo koji su nam sledeći koraci, ako model ima loše performanse samo povećavamo LoRA rank r dok ne postignemo željene rezultate, zbog nedostatka računarskih resursa ja sam se ograničio na r = 16, ali kako vidimo da se performanse vidno povećavaju zajedno sa povećanjem ranka r (Tabela 7) znamo da je moguće postići dosta bolje rezultate...

Isto tako ako vidimo da trening traje predugo možemo uvek smanjiti rank r, što se idealnog LoRA skalirajućeg faktora tiče on je obično 4*r ali treba se utvrditi eksperimentalno uvek...

Ako vidimo da imamo preobučavanje trebamo eksperimentisati sa dropout-om, ovo sam takođe izostavio na kraju zbog nedostatka računarskih resursa, ali poenta ovog rada nije bila dobijanje bilo kakvih poželjnih rezultata nego demonstriranje činjenice da je to moguće metodama parametarski efikasnog finog podešavanja sa veoma malim procentom trenirajućih parametara od ukupnog broja parametara modela.

Iako sam jasno definisiao klasu ''nasilnog govora'' kao pretnje ubistvom i fizičkim nasiljem, nagovaranje na saboubistvo, samopovređivanje i fizičko nasilje, opet su postojali primeri koji i mene zbunjuju i takve primere sam klasifikovao kao nenasilne, ovo samo demonstrira značaj skupa podataka jer uvek će postojati primeri koji su i za čoveka zbunjujući i ako se ti primeri nekoinzistento označe, to će uticati negativno na model.

5 ZAKLJUČAK

Moguća unapređenja rezultata rada uključuju:

1. Proširenje skupa podataka: Trenutni dataset može biti dopunjen dodavanjem više primera koji obuhvataju različite kontekste, kao što su politički, pravni, religijski, ''Geming'' itd... Ovo je generalno najvažniji deo u treniranju jezičkog modela iz prostog razloga što ako je loš skup podataka onda ništa neće moći da pomogne modelu da dobro generalizuje...

2. Dinamičko generisanje skupa podataka za vreme treninga, bilo bi lepo obučiti generativni model baš za generaciju skupa podataka za zadatak za koji treniramo model. Tada u idealnom slučaju možemo generisati dodatne primere a da ih dodatno ne proveravamo ručno. Dakle onda bi bilo moguće za vreme treniranja dodavati u trening skup rečenice koje pretstavljaju augmentaciju rečenica koju model nije prepoznao u trenutnoj epohi – dakle bilo bi moguće skup podataka dinamički menjati iz epohe u epohu.

3. Dodatna eksperimentisanja sa hiperparametrima:

   Povećanje LoRA ranka do postizanja dobrih performansi – ovo naravno za cenu ima usporeno vreme treniranja, eksperimentisanje sa LoRA dropout-om zbog bolje regularizacije koja fali u rezultatima, eksperimentisanje sa različitim mestima za dodavanja LoRA matrica kao što je matrica ''Output projection'' u enkoderskim slojevima...

4. Eksperimentisanje sa različitim modelima koji podržavaju više jezika.

6 LITERATURA

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017, pp. 5998–6008

[2] C. McCormick, *The Inner Workings of BERT*, v1.0.1 ed. 2020, Accessed: Oct. 06, 2024. [Online]. Available: https://www.chrismccormick.ai/bert-ebook

[3] JAIGANESAN, "BERT: In-depth exploration of Architecture, Workflow, Code, and Mathematical Foundations," *Medium*, Jun. 26, 2024. https://pub.towardsai.net/bert-in-depth-exploration-of-architecture-workflow-code-and-mathematical-foundations-0c67ad24725b

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[5] J. Alammar, "The Illustrated Transformer," *jalammar.github.io*, Jun. 27, 2018. https://jalammar.github.io/illustrated-transformer/

[6] "Introduction - Hugging Face NLP Course," *huggingface.co*. https://huggingface.co/learn/nlp-course/chapter1/1

[7] E. J. Hu, Y. Shen, P. Wallis, Z. Yao, D. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proceedings of the 2021 International Conference on Learning Representations (ICLR 2021)*, 2021, pp. 1-20.

[8] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," arXiv preprint arXiv:2305.14314, 2023. [Online]. Available: https://arxiv.org/abs/2305.14314

[9] Classla/bcms-bertic · hugging face (2024) classla/bcms-bertic · Hugging Face. Available at: https://huggingface.co/classla/bcms-bertic (Accessed: 30 September 2024).

[10] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, "Learning from the Worst: Dynamically Generated Datasets to Improve Online Hate Detection," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL 2021)*, 2021.

[11] (2024) *OpenAI*. Available at: https://openai.com/ (Accessed: 30 September 2024).

[12] OpenAI, "openai Python Library," [Online]. Available: https://pypi.org/project/openai/0.28.0/. [Accessed: Oct. 04, 2024].

[13] PyTorch, "PyTorch," [Online]. Available: https://pytorch.org. [Accessed: Oct. 04, 2024].

[14] Hugging Face, "Transformers," [Online]. Available: https://huggingface.co/transformers. [Accessed: Oct. 04, 2024].

[15] Hugging Face, "PEFT: Parameter-Efficient Fine-Tuning Library," [Online]. Available: https://github.com/huggingface/peft. [Accessed: Oct. 04, 2024].

[16] Tim Dettmers, "bitsandbytes: 8-bit optimizers and matrix multiplication routines," [Online]. Available: https://github.com/TimDettmers/bitsandbytes. [Accessed: Oct. 04, 2024].

[17] Optuna, "Optuna: A hyperparameter optimization framework," [Online]. Available: https://optuna.org. [Accessed: Oct. 04, 2024].