

Korištenje jezičnih modela za otkrivanje anomalija u vremenskim serijama

Rebernak, Vid

Master's thesis / Diplomski rad

2025

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:168:533726>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-20**



Repository / Repozitorij:

[FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repository](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br.

Zagreb, siječanj 1970.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br.

Zagreb, siječanj 1970.

Zagreb, 1. siječnja 1970.

DIPLOMSKI ZADATAK br.

Pristupnik: ()

Studij:

Profil:

Mentor:

Zadatak:

Opis zadatka:

Rok za predaju rada: 1. siječnja 1970.

Hvala mojem mentoru izv. prof. dr. sc. Vladimiru Čeperiću za svu njegovu pomoć, usmjeravanje, i izdvojeno vrijeme. Također, hvala mojoj majci Katarini, ocu Robertu i sestri Juliji za njihovu podršku i pomoć kada god je bila potrebna. Posebno hvala bakama Evi i Mariji te djedovima Ivanu i Joži, koji su me podržavali kroz cijelu moju edukaciju i slavili moje uspjehe. Konačno, hvala svim prijateljima i kolegama koji su uvijek bili tu za mene i uljepšali mi cjelokupno obrazovanje.

Sadržaj

1. Uvod	4
2. Teorijska pozadina	6
2.1. Vremenske serije	6
2.2. Otkrivanje anomalija	7
2.3. Veliki jezični modeli	9
2.3.1. Povijest, struktura i značajke velikih jezičnih modela	9
2.3.2. Primjena velikih jezičnih modela u analizi podataka	10
2.4. Povezivanje velikih jezičnih modela i vremenskih serija	12
3. Priprema podataka i osnovna metodologija	14
3.1. Korišteni alati i tehnologije	14
3.1.1. Razvojno okruženje	14
3.1.2. Programski jezik i biblioteke	14
3.2. Priprema podataka	15
3.2.1. Odabrani skupovi podataka	15
3.2.2. Predobrada	16
3.2.3. Podjela i konačan oblik	17
3.3. Prilagodba velikih jezičnih modela za vremenske serije	17
3.3.1. Motivacija za korištenje velikih jezičnih modela u vremenskim se- rijama	18
3.3.2. Klizni prozori i tekstualni prikaz	19
3.3.3. Načini jezičnog modeliranja vremenskih serija	20
3.3.4. Perpleksnost kao indikator anomalija	21
3.3.5. Određivanje pragova i označavanje	22

3.3.6.	Sažetak i sljedeći koraci	23
4.	Implementacija	25
4.1.	Arhitektura sustava	25
4.2.	Implementacija jezičnih modela	27
4.2.1.	<i>GPT-2</i>	27
4.2.2.	<i>DistilBERT</i>	34
4.2.3.	<i>T5</i>	41
4.2.4.	<i>XLNet</i>	47
4.3.	Integracija tradicionalnih metoda	52
4.3.1.	Izolacijska šuma	52
4.3.2.	Faktor lokalnog odstupanja	54
4.3.3.	Jednoklasni stroj potpornih vektora	55
4.3.4.	Eliptična omotnica	56
4.4.	Sažetak cjelokupne implementacije	58
5.	Eksperimentalna evaluacija	59
5.1.	Metrike evaluacije	60
5.1.1.	Točnost	60
5.1.2.	Preciznost	61
5.1.3.	Odziv	61
5.1.4.	F1-rezultat	62
5.2.	Rezultati	62
5.2.1.	Tradicionalne metode	62
5.2.2.	Metode temeljene na velikim jezičnim modelima	65
5.3.	Analiza performansi	69
5.3.1.	Usporedba velikih jezičnih modela i klasičnih metoda	70
5.3.2.	Računalni zahtjevi	71
5.4.	Ključne spoznaje	71
6.	Zaključak	73
	Literatura	75
	Sažetak	83

Abstract **84**

1. Uvod

U posljednjih nekoliko godina, s povećanjem količine podataka u svim područjima ljudskog života, pojavila se globalna potreba za pronalaskom novih pristupa otkrivanja stršćih vrijednosti u podacima vremenskih serija, pogotovo onih koji koriste napredne metode. Anomalije ili stršće vrijednosti značajna su odstupanja od normalnih uzoraka i mogu ukazivati na događaje poput prijevare, kvara sustava ili narušavanja sigurnosti. Osnovne statističke metode za otkrivanje anomalija, iako bitne, nisu vrlo učinkovite i ne mogu se suočiti sa složenošću i velikom dimenzionalnošću podataka s kojima se ljudi susreću u stvarnom svijetu. U proteklih godinu dana došlo je do iznimnog porasta primjene strojnog učenja, naročito velikih jezičnih modela (engl. *large language models*), u svrhu kvalitetnijeg otkrivanja anomalija u podacima vremenskih serija.

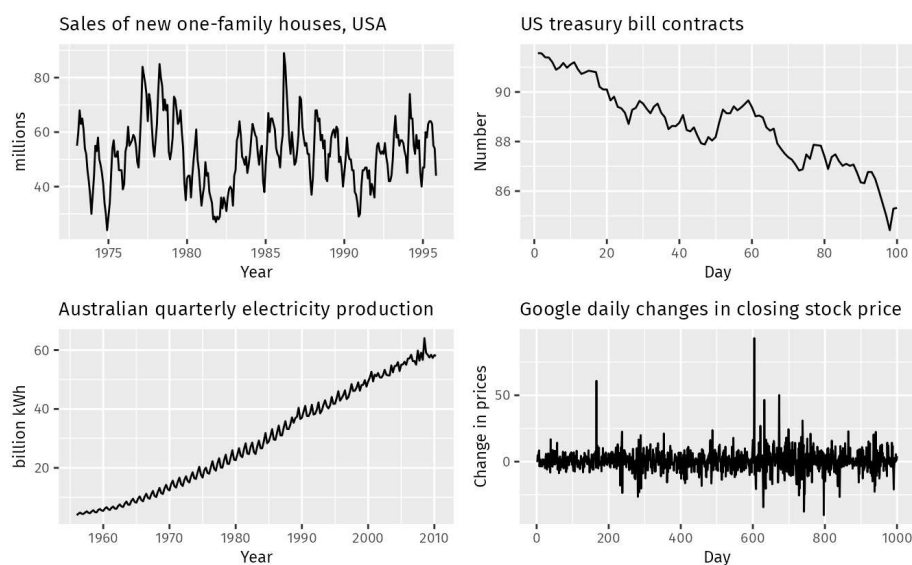
Stoga je fokus ovog diplomskog rada na ograničenjima uobičajenih pristupa otkrivanju anomalija koji uključuju korištenje određenih statističkih pragova ili jednostavnih pravila temeljenih na distribuciji podataka. Navedene metode nisu u mogućnosti uzeti u obzir promjenjiva obilježja podataka, što može rezultirati propustom detekcije pojedinih stršćih vrijednosti ili pojavom lažnih upozorenja. Nasuprot tome, veliki jezični modeli sposobni su prepoznati složene strukture u podacima, a samim time i anomalije u vremenskim serijama (engl. *time series*). Posljedično, postavlja se pitanje jesu li jezični modeli bolja opcija prilikom otkrivanja anomalija te hoće li njihova upotreba u tom području davati točnije rezultate. Ovo istraživanje proučit će na koji način jezični modeli funkcioniraju u prepoznavanju stršćih vrijednosti i usporediti dobivene rezultate s tradicionalnim metodama kako bi se utvrdila njihova učinkovitost. Primjeri vremenskih serija koji pokazuju različite obrasce prikazani su na slici 1.1.

Tri glavna cilja u ovome radu su sljedeća:

1. osigurati cjeloviti pregled literature o relevantnim tehnikama otkrivanja anomalija s posebnim naglaskom na vremenske serije,
2. precizno prilagoditi (engl. *fine-tune*) i primijeniti velike jezične modele u otkrivanju anomalija,
3. usporediti izvođenje navedenih modela s klasičnim pristupima na temelju njihove točnosti i računalne složenosti.

Kroz navedene zadatke, ovo istraživanje nastoji pridonijeti sve češćoj raspravi o korištenju jezičnih modela u analitičke svrhe i njihovom utjecaju na napredak tehnika uočavanja nepravilnosti u podacima.

Prvo će se obraditi teorijska pozadina koja će pružiti potrebne informacije o vremenskim serijama, otkrivanju anomalija i velikim jezičnim modelima u svrhu definiranja osnovnih pojmova na kojima se temelji analiza. Nakon toga, pregled literature omogućit će razumijevanje dosadašnjih istraživanja u ovom području, naglašavajući nedostatke koje ovaj diplomski rad nastoji eliminirati. U idućim poglavljima bit će opisani korišteni modeli i prethodna obrada podataka (engl. *data pre-processing*), načini implementacije i vrednovanja rezultata, rasprava o dobivenim ishodima te preporuke za buduća istraživanja. Na samom kraju rada pružit će se precizno objašnjenje koje naglašava prednosti i nedostatke primjene jezičnih modela prilikom detekcije anomalija u podacima vremenskih serija.



Slika 1.1. Četiri primjera vremenskih serija koji pokazuju različite obrasce (Izvor: [1])

2. Teorijska pozadina

2.1. Vremenske serije

Vremenske serije predstavljaju listu podataka prikupljenih ili zabilježenih u pravilnim vremenskim intervalima. Navedena vrsta podataka sadrži precizan vremenski redak elemenata koji je vrlo bitan jer se u njemu nalaze informacije o točnom trenutku pojedinih događaja u mjerenom sustavu. Primarna obilježja u analizi vremenskih serija su trendovi, sezonalnost, cikličnost i šum. Trend opisuje općeniti smjer u kojemu se nešto razvija ili mijenja tijekom vremena. Projekcija trendova zapravo označava predviđanje budućih promjena u izmjerenim podacima. Trendovi i projekcije prikazuju se najčešće pomoću linijskih grafikona u kojima vodoravna os predstavlja vrijeme. Nadalje, sezonalnost označava redovite promjene u određenim razdobljima koje mogu biti dnevne, mjesečne, godišnje ili slično. S druge strane, ciklički uzorci definiraju se kao nepravilne fluktuacije uočene tijekom duljeg razdoblja čija učestalost nije redovita i pretežito su posljedica dokazano utvrđenih vanjskih čimbenika. Konačno, šum opisuje slučajnu komponentu čije vrijednosti ostaju prisutne nakon odvajanja sezonalnosti, trendova i cikličkih uzoraka od vremenskih serija. Šum je zapravo posljedica nepoznatih ili nemjerljivih čimbenika čiji razlog pojavljivanja nije moguće definirati.

Analiza vremenskih serija može imati mnogobrojne i raznolike primjene. Najviše se koristi u upravljanju financijama, ekonomiji, znanosti o okolišu, zdravstvu, energetici i tehnološkoj industriji. Primjerice, na financijskom tržištu podaci o vremenskim serijama koriste se za analizu cijena dionica kako bi se ulagačima pomoglo u prepoznavanju trendova i donošenju profitabilnih odluka. U zdravstvu, podaci vremenskih serija mogu se koristiti za praćenje vitalnih znakova pacijenata tijekom određenog razdoblja i otkrivanje potencijalnih odstupanja koja mogu upućivati na zloćudnu bolest. Također, važno je napomenuti da je analiza vremenskih serija ključan proces za određivanje vremen-

ske prognoze u meteorologiji ili prilikom praćenja rada pojedinih uređaja u industrijskoj proizvodnji.

2.2. Otkrivanje anomalija

Otkrivanje anomalija proces je ispitivanja specifičnih podatkovnih točaka (engl. *data points*) i identifikacija rijetkih pojava koje se čine sumnjivima jer se razlikuju od utvrđenog obrasca ponašanja. Detekcija stršećih vrijednosti nije nova ideja, ali s rastom količine podataka, ručno praćenje postaje neizvedivo. Iz tog razloga, u današnje se vrijeme navedeni proces oslanja prvenstveno na strojno učenje (engl. *machine learning*). Anomalije se mogu podijeliti u tri glavne skupine: točkaste, kontekstualne i kolektivne. Točkaste se anomalije definiraju kao pojedinačne podatkovne točke čije vrijednosti značajno odstupaju od preostalih članova u skupu podataka. Nadalje, postoje pojedine podatkovne točke koje se smatraju stršećim vrijednostima isključivo u određenom okruženju i one se nazivaju kontekstualnim anomalijama. S druge strane, kolektivne anomalije predstavljaju skup podatkovnih točaka koje prikazuju neuobičajen obrazac ponašanja samo u slučaju zajedničkog proučavanja njihovih vrijednosti.

Tradicionalni načini detekcije stršećih vrijednosti uključuju statističke pristupe, tehnike grupiranja i metode temeljene na strojnom učenju. Statističke metode za otkrivanje anomalija temelje se na matematičkim modelima koji pretpostavljaju normalnu distribuciju podataka. Navedeni pristup često obuhvaća postavljanje pragova koji definiraju raspon uobičajenih vrijednosti te ako se određeni podatak nalazi izvan tog intervala, bit će označen kao odstupanje. Statistički pristup otkrivanja anomalija većinom je jednostavno implementirati i može biti koristan kod vrlo malenih skupova podataka ili u slučaju unaprijed utvrđene statističke distribucije. Među uobičajenim načinima takvog pristupa ističu se metoda percentila i metoda interkvartilnog raspona (IQR). Tehnike grupiranja uključuju grupiranje sličnih podatkovnih točaka u grupe (engl. *clusters*), nakon čega se pokušavaju identificirati elementi koji ne pripadaju niti jednom klasteru ili se nalaze u grupi koja se značajno razlikuje od ostalih. Spomenuti pristup dobro funkcionira u slučaju otkrivanja lokalnih anomalija unutar određenih područja skupa podataka, ali može prepoznati i globalne anomalije koje se pojavljuju na razini cijelog skupa podataka. Metode temeljene na strojnom učenju podrazumijevaju upotrebu nadziranih ili nenadzira-

nih algoritama strojnog učenja u svrhu automatskog uočavanja stršćih vrijednosti bez oslonca na klasičnu statistiku podataka. Istaknuti pristup često zahtijeva skup podataka za obuku koji uključuje uobičajene i neuobičajene podatkovne točke. Na temelju takvog skupa gradi se model sposoban prepoznati anomalije u novim, još neviđenim podacima.

Međutim, korištenje navedenih tradicionalnih metoda može uzrokovati nailazak na pojedine prepreke. Najzastupljeniji izazovi koji se javljaju prilikom njihovog korištenja uključuju pretpostavku o specifičnoj distribuciji podataka, problem skalabilnosti i visoke dimenzionalnosti, nemogućnost prilagodbe na dinamičko okruženje podataka, osjetljivost parametara i ograničeno rukovanje složenim anomalijama.

1. Mnoge statističke metode zasnivaju se na pretpostavci da podaci slijede neku specifičnu distribuciju (primjerice, normalnu distribuciju). Ali u slučaju odstupanja podataka iz stvarnog svijeta od navedene pretpostavke, učinkovitost takvih metoda značajno se smanjuje.
2. Tehnike grupiranja, poput k-najbližih susjeda (kNN), postaju iznimno računski zahtjevne s povećanjem količine podataka, što smanjuje njihovu efikasnost kod velikih skupova podataka.
3. U visoko dimenzionalnim prostorima, koncept udaljenosti skoro u potpunosti gubi svoj smisao, a samim time i mnoge metode temeljene na udaljenosti postaju neprikladne. Navedeno obilježje općenito se naziva prokletstvo dimenzionalnosti i može rezultirati vrlo lošom izvedbom korištenih modela.
4. Tradicionalne metode često nemaju mogućnost prilagodbe na promjene u distribucijama podataka koje su nastale kao posljedica dinamičkih okruženja, čija je glavna karakteristika pojava fluktuacije u izmjerenim vrijednostima tijekom vremena.
5. Veliki broj klasičnih algoritama izrazito je osjetljiv na promjenu vrijednosti parametara (primjerice, broj klastera u tehnikama grupiranja). Spomenute parametre često je potrebno ručno postaviti i vjerojatno su specifični za korišteni skup podataka, što stvara problem kod pokušaja primjene prilagođenih vrijednosti na druge podatkovne skupove.
6. Mnogi standardni pristupi otkrivanja anomalija skloni su lošoj izvedbi kod stršćih

vrijednosti koje ovise o kontekstu ili u slučajevima kada nije moguće identificirati odstupanje na temelju promatranja isključivo jedne varijable.

Ovi izazovi doveli su do razvoja novih tehnika, uključujući pristupe temeljene na velikim jezičnim modelima, s ciljem kvalitetnije detekcije anomalija.

2.3. Veliki jezični modeli

Veliki jezični modeli napredni su oblici umjetne inteligencije sposobni analizirati i generirati ljudski jezik do najjednostavnijeg stupnja. Posljednjih godina zabilježen je porast njihove upotrebe koji predstavlja značajan napredak u razvoju umjetne inteligencije, posebice u obradi prirodnog jezika (engl. *natural language processing*, NLP). Postali su iznimno važni u mnogim područjima ljudskog djelovanja, poput strojnog prevođenja, stvaranja sadržaja i razgovornih agenata.

2.3.1. Povijest, struktura i značajke velikih jezičnih modela

Povijest razvoja velikih jezičnih modela sastoji se od nekoliko bitnih prekretnica.

Rani razvoj: Tijekom 1990-ih godina, tvrtka *International Business Machines Corporation* (IBM) uvela je koncept modeliranja statističkog jezika. Do 2001. godine, izgladeni n-gramski modeli (engl. *smoothed n-gram models*), trenirani na skupu od 0.3 milijarde riječi, dosegli su najvišu razinu izvedbe u jezičnim zadacima.

Integracija neuronskih mreža: Primjena neuronskih mreža na jezično modeliranje početkom 2010-ih godina pridonijela je poboljšanju izvedbe zadataka poput strojnog prevođenja. Kompanija *Google* uspješno je pokazala mogućnost korištenja neuronskog strojnog prevođenja (engl. *Neural Machine Translation*) 2016. godine, čime je uvelike utjecala na razvoj jezičnih modela.

Arhitektura transformatora: Godine 2017. računalni znanstvenik Vaswani i njegovi suradnici predstavili su arhitekturu transformatora u svojem radu *Attention Is All You Need*, koja je uspjela radikalno unaprijediti jezično modeliranje, omogućivši učinkovitiju obuku modela na velikim skupovima podataka. Navedeno otkriće popraćeno je razvojem modela poput BERT-a 2018. godine, a potom i poznatog

GPT-a, koji je ubrzo postao najzastupljeniji diljem svijeta.

Veliki jezični modeli primarno su građeni na principu arhitekture transformatora, koja se sastoji od mehanizma samopozornosti (engl. *self-attention mechanism*), strukture koder-dekoder i višeglavog mehanizma pozornosti (engl. *multi-head attention*). Mehanizam samopozornosti koristi se za određivanje važnosti pojedine riječi u rečenici, preciznim proučavanjem njezine kontekstualne ovisnosti o drugim dijelovima rečenice. Transformator se po definiciji sastoji od kodera, koji obrađuje ulazne podatke, i dekodera, koji proizvodi izlaz. Međutim, postoje određeni modeli čija se arhitektura temelji samo na jednom od navedenog. Primjerice, BERT u sebi sadrži isključivo koder, dok GPT koristi samo dekoder prilikom jezičnog modeliranja. Višeglavi mehanizam pozornosti osigurava istovremenu analizu različitih dijelova ulaznog niza i na taj način omogućuje učinkovitu identifikaciju nepoznatih obrazaca u podacima.

Značajke velikih jezičnih modela nisu ograničene isključivo na stvaranje tekstualnog sadržaja, već obuhvaćaju i sposobnost logičkog razmišljanja, generiranje programskog koda i rješavanje složenih problema. Konkretno, nedavno istraživanje [2] pokazalo je da veliki jezični modeli mogu pomoći u formalizaciji matematičkih dokaza i generiranju pojedinih isječaka koda te prilikom rješavanja kreativnih i analitičkih zadataka. Međutim, njihova izvedba nailazi na određena ograničenja, poput pristranosti u podacima iz skupa za treniranje, pojave netočnih ili izmišljenih informacija (engl. *hallucination problem*) i problema nedostatka jasnih i logičnih objašnjenja za dobivene zaključke. Iz navedenih razloga, još uvijek postoje veliki problemi koje istraživači ističu kao područja za poboljšanje. Također, etička pitanja primjene velikih jezičnih modela, pretežito u visokorizičnim područjima kao što su zdravstvo i obrazovanje, dodatno otežavaju njihovu upotrebu. Razradom tih tema pokušava se analizirati utjecaj jezičnih modela na društvo u cjelini i nužnost uvođenja određenih regulacija kojima je glavni cilj zaštita osobnih podataka.

2.3.2. Primjena velikih jezičnih modela u analizi podataka

Korištenje velikih jezičnih modela u analizi podataka dovelo je do otkrivanja novih načina ekstrakcije informacija iz velikih količina nestrukturiranih podataka. Pokazali su se vrlo korisnima u brojnim područjima poput zdravstva, financija i društvenih zna-

nosti zbog svoje sposobnosti jednostavne obrade i analize tekstualnih podataka. Primjerice, u zdravstvu, jezični modeli koriste se u obradi pacijentove povijesti bolesti te analizi znanstvenih i kliničkih dokumenata u svrhu efikasnijeg donošenja odluka i planova liječenja. Njihova integracija u sustave zdravstvene skrbi istaknula je značajan potencijal u smanjenju ljudskih pogrešaka i automatizaciji rutinskih zadataka, čime se zdravstvenim radnicima omogućuje usmjerivanje energije na brigu o pacijentima.

Unutar financijske domene, jezični modeli koriste se za prepoznavanje tržišnog sentimenta iz novinskih članaka, analizu objava na društvenim mrežama i obradu financijskih izvješća. Navedena sposobnost pomaže financijskim analitičarima kod utvrđivanja raspoloženja javnosti i donošenja racionalne odluke o ulaganju na temelju podataka u stvarnom vremenu. Također, izrazito su učinkoviti u pružanju pomoći kod otkrivanja prijevara identificiranjem sumnjivih obrazaca u transakcijskim podacima, čime se uvelike poboljšava sigurnost financijskih sustava. Brza prilagodba velikih jezičnih modela u radu s različitim vrstama podataka dokaz je njihove sposobnosti unapređenja i modernizacije konvencionalnih pristupa analizi podataka.

Nadalje, veliki jezični modeli ostvarili su veliki potencijal za primjenu u obrazovnim sustavima, u kojima se mogu koristiti kod analize podataka o uspjehu studenata i pružanju individualnih preporuka za učenje. Uz njihovu pomoć, nastavnici će biti u mogućnosti razumjeti studentove poteškoće u učenju i osmisliti načine za njihovo rješavanje kroz prilagođene nastavne materijale kako bi se olakšao cijeli proces. Prediktivne funkcije jezičnih modela također je moguće iskoristiti prilikom upravljanja resursima u obrazovnim institucijama u svrhu pomaganja administrativnim zaposlenicima kod donošenja odluka o alokaciji sredstava za infrastrukturu i razvoja kurikuluma.

Unatoč velikom broju potencijalnih primjena, korištenje velikih jezičnih modela u analizi podataka uzrokuje i određene probleme poput spomenute privatnosti podataka, etičkih dilema i potrebe za transparentnošću algoritamskog odlučivanja. Budući da se modeli moraju učiti na golemim količinama podataka, pojavila se zabrinutost oko etike korištenja i posjedovanja takvih tehnologija. Osim toga, načelo crne kutije (engl. *black box principle*) također je jedan od problema zbog kojih se postavlja pitanje odgovornosti i razumljivosti samih rezultata, osobito u visokorizičnim domenama poput zdravstva i financija. Navedene izazove potrebno je prvo savladati kako bi se osiguralo ispravno

korištenje jezičnih modela u analizi podataka.

Opisana obilježja velikih jezičnih modela ističu njihov revolucionarni utjecaj na različita područja, s posebnim naglaskom na analizu podataka. Njihova sposobnost razumijevanja i generiranja ljudskog jezika omogućila je široki raspon primjena u zdravstvu, financijama, obrazovanju i mnogim drugim područjima ljudskog djelovanja. Međutim, poput svakog novog izuma, njihova primjena nosi određene moralne dileme koje je potrebno oprezno preispitati kako bi se osiguralo pravilno i odgovorno korištenje navedenih alata.

2.4. Povezivanje velikih jezičnih modela i vremenskih serija

Područje istraživanja velikih jezičnih modela u kombinaciji s analizom vremenskih serija polako postaje aktivna tema zbog njihove rastuće upotrebe u radu s različitim vrstama podataka, uključujući i one koji nisu samo tekstualni. Klasična analiza vremenskih serija zasniva se na statističkim pristupima i metodama strojnog učenja poput algoritama ARIMA i LSTM koji zahtijevaju detaljno poznavanje domene i ručnu obradu značajki pojedinog skupa podataka. S druge strane, veliki jezični modeli prikladni su za navedene probleme zbog svoje sposobnosti prilagodbe različitim zadacima, koristeći prethodno naučeno znanje. To ih čini osobito korisnima u predviđanju vremenskih serija te klasifikaciji i otkrivanju anomalija.

Veliki jezični modeli izvorno su stvoreni za rad s diskretnim tekstualnim podacima, što stvara problem kod pokušaja njihove primjene na kontinuirane numeričke varijable poput vremenskih serija. S ciljem rješavanja opisanog problema, stručnjaci u tom području [3] i [4] predložili su korištenje metoda poput strategije tokenizacije, kvantizacije i poravnanja ugrađenih prostora (engl. *embedding space alignment*). Primjerice, numeričke vrijednosti vremenskih serija mogu se pretvoriti u tekstualne tokene, što se ostvaruje korištenjem metode pod nazivom brojevno specifična tokenizacija. Navedeni pristupi osiguravaju ispravno upravljanje numeričkim vrijednostima pomoću arhitekture velikih jezičnih modela koja se prvenstveno zasniva na tekstualnim podacima. Na taj način omogućuje se provedba predviđanja i klasifikacije putem paradigme temeljene

na upitima (engl. *prompt-based paradigm*).

Kao što je opisano u literaturi [3] i [5], veliki jezični modeli pokazali su se učinkovitim u zadacima poput analize vremenskih serija, čije vrijednosti pokazuju prisutnost pojedinih sezonalnosti ili trendova. To se može objasniti njihovom sposobnošću uočavanja određenih obrazaca i predikcije periodičkih trendova korištenjem visoko razvijenog oblikovanja upita i učenja iz konteksta (engl. *in-context learning*). Drugim riječima, pružanje informacija o vremenskim karakteristikama podataka ili izražavanje numeričkih nizova prirodnim jezikom značajno pridonosi kvalitetnijoj izvedbi velikih jezičnih modela. Navedeni pristupi povećavaju preciznost predviđanja, ali i pojednostavljuju interpretaciju dobivenih predikcija.

Nadalje, velike jezične modele moguće je pripremiti za poravnanje podataka vremenskih serija s njihovim vlastitim ugrađenim prostorima, što se postiže korištenjem koodera koji prevode vremenske serije u odgovarajući oblik za jezične modele. To omogućuje razvoj multimodalnih aplikacija u kojima veliki jezični modeli istovremeno obrađuju tekstualne, vizualne i vremenske podatke, čime se povećavaju mogućnosti njihove primjene u različitim područjima.

Iako se veliki jezični modeli mogu primijeniti u različitim postavkama, nailaze na poteškoće u slučaju više različitih obrazaca ili prisutnosti velikog šuma. Općenito su manje učinkoviti u obradi složenih vremenskih struktura koje zahtijevaju dugotrajne zavisnosti ili visoku razinu granularnosti. U svrhu ublažavanja tih nedostataka, upotrebljavaju se analiza Gaussovog šuma i tehnike kliznog prozora za otkrivanje osjetljivih područja u ulaznim nizovima. Očekuje se da će buduća istraživanja nastaviti razvoj otpornosti velikih jezičnih modela usavršavanjem njihove sposobnosti modeliranja složenih obrazaca.

3. Priprema podataka i osnovna metodologija

U ovome poglavlju definiraju se odabrane metode i resursi potrebni prilikom pripreme podataka i modela za detekciju anomalija. Navedene informacije obuhvaćaju opis softverskog okruženja, tehnike predobrade podataka te prilagodbu velikih jezičnih modela za analizu vremenskih serija.

3.1. Korišteni alati i tehnologije

3.1.1. Razvojno okruženje

Za cjelokupan razvoj i eksperimentalni dio rada korišten je alat *Google Colab Pro*, koji nudi okruženje *Jupyter* bilježnice temeljene na oblaku, s dodatkom grafičkog procesora *NVIDIA A100*. Navedena platforma temelji se na operacijskom sustavu *Linux* i omogućuje upotrebu zajedničkog skupa funkcionalnosti za analizu podataka i znanstveno računarstvo.

3.1.2. Programski jezik i biblioteke

- **Python (inačica 3.11.11):** programski jezik za implementaciju praktičnog dijela rada i analizu podataka, odabran radi njegove sveobuhvatne podrške strojnom učenju.
- **Biblioteke za analizu podataka:** *NumPy* i *pandas* za rukovanje poljima i podatkovnim okvirima te *matplotlib* za vizualizaciju dijagrama.
- **Scikit-learn:** biblioteka za provedbu tradicionalnih algoritama detekcije anomalija i izračun evaluacijskih pokazatelja uspješnosti modela.

- **Integracija biblioteka *PyTorch* i *Hugging Face Transformers*:** omogućuje realizaciju procesa dubokog učenja, osobito učitavanje velikih jezičnih modela, tokenizaciju, preciznu prilagodbu i zaključivanje.
- ***Hugging Face Datasets*:** osigurava pouzdano učitavanje podataka i sinkronizaciju s bibliotekom *Transformers*.
- ***Jupyter Notebook*:** interaktivno *Python* okruženje za postupno izvođenje eksperimenata, neposrednu vizualizaciju i pohranu rezultata.

3.2. Priprema podataka

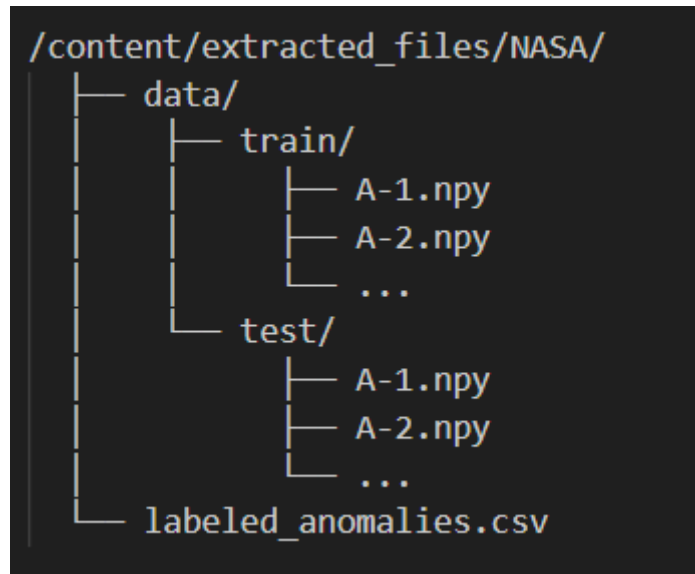
3.2.1. Odabrani skupovi podataka

U svrhu otkrivanja stršećih vrijednosti, korišteni su skupovi podataka pod nazivom *Soil Moisture Active Passive* (SMAP) i *Mars Science Laboratory* (MSL).

SMAP je satelit za praćenje okoliša koji mjeri vlažnost tla diljem Zemlje, pod vlasništvom državne civilne uprave Sjedinjenih Američkih Država za zrakoplovna i svemirska istraživanja i razvoj (NASA). Osmišljen je na način da skuplja globalne podatke o vlažnosti tla svakih nekoliko dana.

MSL je operacija lansiranja robotske svemirske istraživačke jedinice na Mars, koju je izvela NASA, 26. studenog 2011. godine. Glavni cilj navedene inicijative bio je istražiti nastanjivost tog planeta, proučiti njegovu klimu i geologiju te prikupiti podatke radi moguće ljudske misije na Mars. Opisan telemetrijski sustav u sebi sadrži niz znanstvenih instrumenata, koji služe za mjerenje željenih vrijednosti.

U odabranim skupovima podataka nalaze se višestruki "kanali" (senzori ili skupovi značajki), spremljeni u obliku `.npy` datoteke. SMAP kanali imaju 25-dimenzionalne podatke, a MSL kanali sastoje se od 55 dimenzija. Nadalje, `.csv` datoteka pod nazivom *labeled_anomalies* sadrži intervale `[start_idx, end_idx]`, koji označavaju točno vrijeme pojave stršećih vrijednosti u određenom kanalu. Struktura opisanog direktorija prikazana je na slici 3.1.



Slika 3.1. Struktura direktorija u kojem se nalaze korišteni skupovi podataka

3.2.2. Predobrada

Svaka `.npy` datoteka učitana je u `NumPy` polje. Za SMAP, polja moraju biti oblika $[N, 25]$, a za MSL $[N, 55]$. Ako pojedini podatak ne odgovara navedenom formatu, bit će preskočen. Zatim su dva odvojena polja *train* i *test* konceptualno povezana u jedan *pandas* podatkovni okvir s novostvorenim stupcem *split*, koji označava je li podatak pripadao skupu za treniranje ili skupu za testiranje. Nakon toga se raščlanjuju intervali učitani iz datoteke *labeled_anomalies* i označavaju se pripadni uzorci na način postavljanja vrijednosti varijable *is_anomaly*. Također, važno je naglasiti eliminaciju svih anomalija iz skupa za učenje kod pojedinih tradicionalnih metoda, poput izolacijske šume i faktora lokalnog odstupanja, budući da one očekuju isključivo normalne podatke prilikom treniranja modela. Implementacija opisanog postupka vidljiva je u primjeru 1.

```

def build_smap_dict_for_classical(data_folder: str, labels_csv: str)
    -> Dict[str, pd.DataFrame]:
    anomalies_df = pd.read_csv(labels_csv)
    chan_intervals = {}
    for _, row in anomalies_df.iterrows():
        sc = row.get("spacecraft", "Unknown")
        if sc != "SMAP":
            continue

```

```

chan_id    = row["chan_id"]
intervals = parse_anomaly_sequences(row["anomaly_sequences"])
if chan_id not in chan_intervals:
    chan_intervals[chan_id] = []
chan_intervals[chan_id].extend(intervals)
...
smap_dict  = {}
for cid in all_chs:
    ...
    labels  = np.zeros(full_len, dtype=int)
    for (s_idx, e_idx) in chan_intervals[cid]:
        s = max(0, s_idx)
        e = min(full_len-1, e_idx)
        labels[s:e+1] = 1
    df_tr["is_anomaly"] = labels[:len(df_tr)]
    df_te["is_anomaly"] = labels[len(df_tr):]
    df_channel = pd.concat([df_tr, df_te], ignore_index=True)
    smap_dict[cid] = df_channel
return smap_dict

```

Primjer 1: Isječak koda koji prikazuje prethodnu obradu podataka

3.2.3. Podjela i konačan oblik

Nakon prethodne obrade podataka, svaki kanal pretvoren je u podatkovni okvir koji sadrži telemetrijske značajke (*value_0*, *value_1*, ..., *value_24*), pripadnost skupu za treniranje ili testiranje (*split*), vremensku oznaku mjerenja (*timestep*) i pokazatelj odstupanja (*is_anomaly*).

3.3. Prilagodba velikih jezičnih modela za vremenske serije

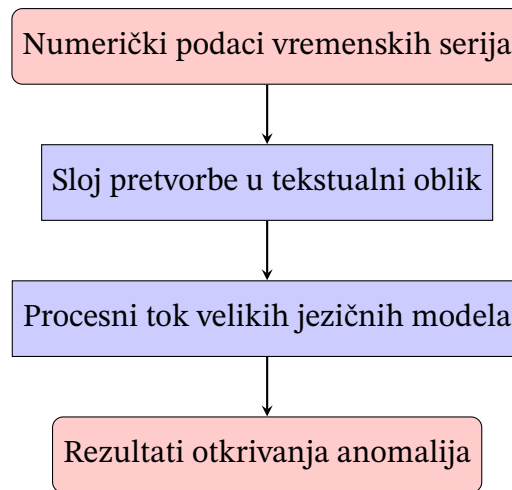
Mnogobrojni istraživači u području računarne znanosti redovito koriste velike jezične modele za zadatke poput predviđanja, sažimanja i transformacije tekstualnih po-

dataka zbog njihove iznimno kvalitetne sposobnosti učenja složenih obrazaca iz sekvencijskih podataka. Unatoč činjenici da su navedeni modeli prvenstveno izvježbani na tekstualnim podacima, njihova sposobnost prepoznavanja dugoročnih ovisnosti čini ih potencijalno prikladnima i za vremenske serije. U ovome poglavlju pojasnit će se konceptualni koraci implementirani tijekom procesa prilagodbe velikih jezičnih modela za otkrivanje stršćih vrijednosti u vremenskim podacima, uključujući klizne prozore, pretvorbu numeričkih vrijednosti u tekstualni oblik, ciljeve pojedinih modela i izvođenje dobivenih rezultata. Shematski prikaz procesnog toka velikih jezičnih modela za otkrivanje anomalija vidljiv je na slici 3.2.

3.3.1. Motivacija za korištenje velikih jezičnih modela u vremenskim serijama

Vremenske serije često uključuju višedimenzionalne signale sa zamršenim odnosima između značajki i pojavu nelinearnih vremenskih obrazaca. Veliki jezični modeli temeljeni na arhitekturi transformera donose nekoliko prednosti.

1. **Kontekstualno učenje:** Na temelju spomenutog mehanizma samopozornosti, veliki jezični modeli mogu procijeniti važnost različitih dijelova niza kako bi otkrili skrivene odnose u kratkoročnim i dugoročnim vremenskim razmacima.
2. **Prenosivost:** Unaprijed istrenirani modeli naučili su mnogobrojne načine predstavljanja različitih sekvenci. Njihova precizna prilagodba na brojevnim podacima pretvorenima u tekst može učinkovito iskoristiti navedene sposobnosti. Time se postiže uspješno smanjivanje zahtjeva za velikom količinom podataka, za razliku od algoritama koji počinju svoju obuku od nule (engl. *training from scratch*).
3. **Otpornost na šum:** U tekstualnim podacima, veliki jezični modeli iznimno dobro rješavaju problem nepravilnosti poput prisutnosti pravopisnih pogrešaka, razumijevanja različitih dijalekata i uporabe žargona. Analogno, u analizi vremenskih serija ponekad mogu uspješno obraditi mjerenja u kojima je prisutan šum ili djelomično neispravni senzori. Ukoliko su pravilno prilagođeni na navedenu vrstu podataka, u takvim slučajevima postižu puno bolje rezultate od većine tradicionalnih algoritama.



Slika 3.2. Konceptualni prikaz prilagodbe procesnog toka velikih jezičnih modela za otkrivanje anomalija u vremenskim serijama

3.3.2. Klizni prozori i tekstualni prikaz

Kako bi se omogućilo ispravno rukovanje potencijalno neograničenim vremenskim serijama, koristi se pristup kliznog prozora za izdvajanje dijelova koji se mogu učinkovito analizirati. Svaki prozor sastoji se od točno pedeset uzastopnih uzoraka (*window_size*). Veličina koraka (*step_size*) postavljena je na dvadeset i pet, što znači da se uzastopni prozori preklapaju na polovici.

Matematički, ako $X = (x_1, x_2, \dots, x_T)$ predstavlja vremensku seriju duljine T , svaki $x_t \in \mathbb{R}^d$ je d -dimenzionalni vektor značajki u trenutku t . Za $t \in \{1, \dots, T - w + 1\}$ s pomakom vrijednosti s , dobivaju se prozori:

$$W_t = (x_t, x_{t+1}, \dots, x_{t+w-1}). \quad (3.1)$$

To daje približno $\lceil (T - w)/s \rceil$ prozora, od kojih svaki prepoznaje pripadni lokalni vremenski kontekst.

Svaki prozor, sada $(w \times d)$ -dimenzionalno brojevno polje, pretvara se u znakovni niz od nekoliko redaka, pri čemu jedan redak predstavlja jedan vremenski trenutak.

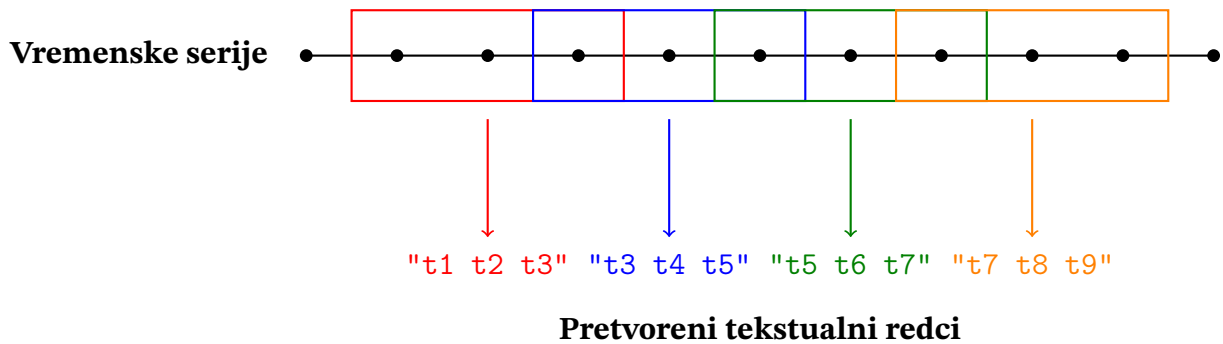
Jedan red $x_t = (x_{t,1}, x_{t,2}, \dots, x_{t,d})$ postaje:

$$"x_{t,1} \quad x_{t,2} \quad \dots \quad x_{t,d}." \quad (3.2)$$

Dakle, svaka značajka odvojena je razmakom i na taj način postiže se čuvanje granica pojedinih obilježja u tekstualnom obliku.

Redovi su povezani s razdjelnicima za novi redak ($\backslash n$) kako bi se dobio strukturirani isječak teksta. Ovaj se isječak zatim prosljeđuje tokenizatoru, koji dijeli znakovne nizove s pomičnom točkom u manje podnizove (primjerice, "1.234" \rightarrow ["1", ".", "234"]), osim ako se ne koristi numerički tokenizator specifičan za pojedinu domenu.

Kodiranjem numeričkih obrazaca u tekstualni oblik otvara se mogućnost iskorištavanja unaprijed naučenih ugrađenih reprezentacija rječnika (engl. *vocabulary embeddings*) i mehanizama pažnje (engl. *attention mechanisms*), dok se istovremeno čuva vremenski poredak (redak po redak).



Slika 3.3. Shematska reprezentacija podjele vremenske serije u preklapajuće prozore i pretvorbe u tekstualne uzorke

3.3.3. Načini jezičnog modeliranja vremenskih serija

Nakon završene pretvorbe prozora u tekstualni oblik, moguće je primijeniti korake standardnog jezičnog modeliranja. Postoji više različitih načina za navedeni postupak.

1. **Kauzalno jezično modeliranje (CLM):** Modeli poput GPT-2 obrađuju niz slijeva nadesno, s ciljem predviđanja sljedećeg znaka, pritom uzimajući u obzir sve prethodno promatrane znakove. Za tokenizirani prozor $z = (z_1, z_2, \dots, z_N)$, cilj učenja zasniva se na minimizaciji negativne logaritamske vjerojatnosti:

$$L_{\text{CLM}}(\theta) = - \sum_{i=1}^N \log P_{\theta}(z_i | z_1, \dots, z_{i-1}). \quad (3.3)$$

Ovdje θ označava parametre modela. U kontekstu numeričkih vremenskih serija,

elementi z_i predstavljaju podznakove koji kodiraju decimalne znamenke.

2. **Maskirano jezično modeliranje (MLM)**: Pojedine arhitekture (poput one u modelu DistilBERT) nasumično maskiraju dio znakova iz ulaznog niza, čime se pokušava naučiti model na rekonstrukciju tih znakova iz nemaskiranog konteksta:

$$L_{\text{MLM}}(\theta) = -\mathbb{E}_m \left[\sum_{i \in m} \log P_{\theta}(z_i | z_{\setminus i}) \right], \quad (3.4)$$

pri čemu je m skup maskiranih indeksa, a $z_{\setminus i}$ označava skup svih znakova osim z_i . Dvosmjerna struktura maskiranog jezičnog modeliranja može učinkovitije obuhvatiti ovisnosti u cijelom prozoru, umjesto isključivo slijeva nadesno.

3. **Modeliranje sekvence u sekvencu (Sequence-to-Sequence)**: Modeli kao što su T5 i XLNet mogu na sličan način naučiti prikaze numeričkih prozora, često koristeći strukturu koder-dekoder ili jezično modeliranje temeljeno na permutaciji. Iako su izvorno namijenjeni zadacima prevođenja i sažimanja, mogu se prilagoditi za učenje brojevni obrazaca, što se najčešće postiže rekonstrukcijom čitavog prozora iz maskiranog ili djelomično vidljivog ulaza.

Svaki od navedenih pristupa implicitno kodira ono što čini "normalan" prozor. U trenutku nailaska na neuobičajen niz, naučeni obrasci rezultiraju izračunom veće vrijednosti prediktivne nesigurnosti, koja se potom koristi za otkrivanje anomalija.

3.3.4. Perpleksnost kao indikator anomalija

Osnovna mjera u jezičnom modeliranju je perpleksnost, koja numerički procjenjuje kvalitetu predviđanja određenog niza. Za prozor z duljine N , perpleksnost (PPL) je definirana kao:

$$\text{PPL}(z) = \exp \left(-\frac{1}{N} \sum_{i=1}^N \log P_{\theta}(z_i | z_{<i}) \right). \quad (3.5)$$

Intuitivno, ako je vrijednost $\text{PPL}(z)$ visoka, model je "iznenađen" nizom. S druge strane, ukoliko je dobivena vrijednost niska, obilježja niza usklađena su s obrascima naučenim u postupku treniranja modela.

Pojedini prozori nakon tokenizacije mogu premašiti maksimalno dozvoljenu duljinu niza. U tom slučaju, tekst se dijeli na segmente. Ukupna negativna logaritamska vjero-

jatnost (engl. *Negative Log-Likelihood*, NLL) dobiva se zbrajanjem vrijednosti unutar navedenih dijelova, a perpleksnost se zatim izračunava na temelju agregiranih vrijednosti. Svakom prozoru iz skupa za testiranje odgovara točno jedna mjera perpleksnosti. Ponavljanjem opisanog postupka za sve prozore u skupu za ispitivanje, dobiva se distribucija vrijednosti perpleksnosti, iz čega je moguće označiti stršeće vrijednosti u podacima.

3.3.5. Određivanje pragova i označavanje

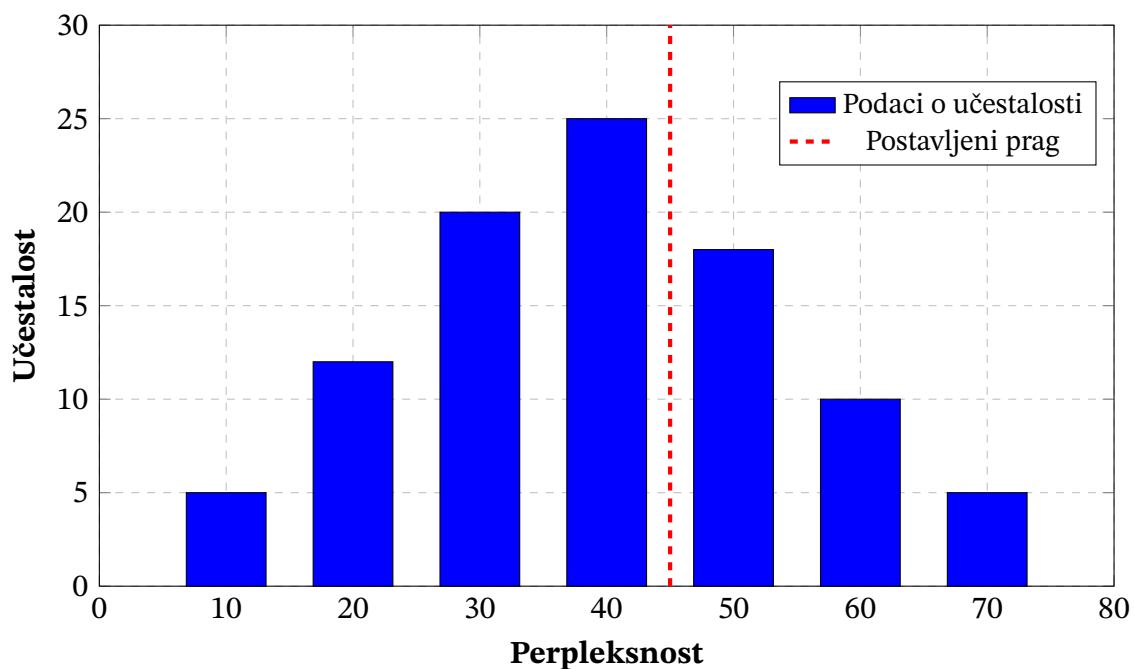
U nenadziranim ili polunadziranim tehnikama učenja, odabir vrijednosti koja će predstavljati prag za anomalije iznimno je bitan. Najzastupljeniji pristup je definiranje praga τ kao, primjerice, devedeset i devetog percentila distribucije perpleksnosti na validacijskom skupu ili na dijelu podataka za koje se pretpostavlja da su uobičajeni. Iz toga slijedi

$$y(z) = \begin{cases} 1, & \text{ako } PPL(z) > \tau, \\ 0, & \text{inače,} \end{cases} \quad (3.6)$$

gdje $y(z)$ predstavlja oznaku anomalije za prozor z .

Ukoliko je dostupan mali i označeni skup podataka, varijabla τ može se podešavati maksimiziranjem željene vrijednosti (primjerice F1-rezultata) na validacijskom skupu. Takav pristup može postići bolje rezultate od naivnih pristupa temeljenih na izračunu percentila, pod uvjetom da su referentne anomalije reprezentativne.

U oba slučaja, perpleksnost omogućuje korištenje izravnog pristupa za klasifikaciju prozora od najobičnijih (najniža perpleksnost) do najsumnjivijih (najveća perpleksnost).



Slika 3.4. Ilustrativni primjer histograma koji prikazuje različite vrijednosti perpleksnosti za prozore iz skupa za testiranje i pripadnu liniju praga

3.3.6. Sažetak i sljedeći koraci

Prilagodba velikih jezičnih modela za otkrivanje anomalija u vremenskim serijama obuhvaća izdvajanje međusobno preklapajućih prozora, pretvaranje brojevnih nizova u tekstualni oblik i primjenu različitih načina standardnog jezičnog modeliranja u svrhu učenja uobičajenih obrazaca. Perpleksnost tada postaje ključna vrijednost u nenadziranom učenju koja služi za ocjenjivanje svakog prozora, otkrivajući moguće stršće vrijednosti bez potrebe za dodatnim označenim podacima. Unatoč navedenim prednostima, još uvijek postoji nekoliko područja u kojima su potrebna poboljšanja i daljnja istraživanja.

Gotovi tokenizatori mogu dijeliti numeričke podatke na neoptimalan način, povećavajući duljinu ulaznog niza i složenost procesa učenja. Razvijanje brojno specifičnih strategija tokenizacije moglo bi riješiti ovaj problem.

Unaprijed naučeni veliki jezični modeli često zahtijevaju obilne resurse koje ponekad nije moguće ostvariti. Tehnike poput destilacije modela ili kvantizacije mogu se primijeniti u slučajevima kada je ograničena dostupnost potrebnih sredstava.

Povezivanje značajki temeljenih na velikim jezičnim modelima i klasičnih metoda

detekcije anomalija (primjerice, korištenje perpleksnosti kao ulaz u algoritam *Isolation Forest*) stvara potpuno novu priliku za poboljšanje otpornosti i razumljivosti korištenih pristupa. Iscrpna analiza navedenih funkcionalnosti bit će detaljno razrađena u poglavlju 5.

Zaključno, jezično modeliranje predstavlja prilagodljiv i sofisticiran alat za otkrivanje anomalija u vremenskim serijama. U sljedećem poglavlju detaljno će se objasniti postavljanje čitavog okruženja, struktura programskog koda te integracija velikih jezičnih modela i tradicionalnih metoda detekcije stršećih vrijednosti, nadograđujući konceptualne temelje postavljene u ovom odjeljku.

4. Implementacija

Ovo poglavlje pruža detaljan pregled implementacije procesnog toka korištenog za otkrivanje anomalija u odabranim skupovima podataka pod vlasništvom državne civilne uprave Sjedinjenih Američkih Država za zrakoplovna i svemirska istraživanja i razvoj. Precizno su opisane tradicionalne metode otkrivanja anomalija i primjena velikih jezičnih modela u istu svrhu. Pododjeljak 4.1. predstavlja arhitekturu cjelokupnog sustava i njegove glavne funkcionalne cjeline. Zatim je u potpoglavlju 4.2. temeljito razrađena sama realizacija velikih jezičnih modela u kontekstu detekcije stršećih vrijednosti, pozivajući se na izdvojene isječke programskog koda. Nakon toga, pododjeljak 4.3. objašnjava vrste tradicionalnih pristupa odabrane za implementaciju i njihovu integraciju u procesni tok za komparativnu analizu s velikim jezičnim modelima. Konačno, potpoglavlje 4.4. navodi tehničke pojedinosti koje su bile potrebne za uspješnu provedbu ciljeva.

4.1. Arhitektura sustava

Cjelokupni sustav sastoji se od tri osnovna dijela.

1. Funkcionalna cjelina za **prikupljanje i predobradu podataka**:

- odgovorna za izdvajanje podataka vremenskih serija iz .npy datoteka, njihovo povezivanje s oznakama anomalija iz .csv datoteke *labeled_anomalies* te provedbu postupaka čišćenja i formatiranja podataka,
- generira strukturirane *pandas* podatkovne okvire za svaki kanal koji sadrži vremenske trenutke, stupce pojedinih značajki, oznake skupa za treniranje ili testiranje i stvarne oznake anomalija.

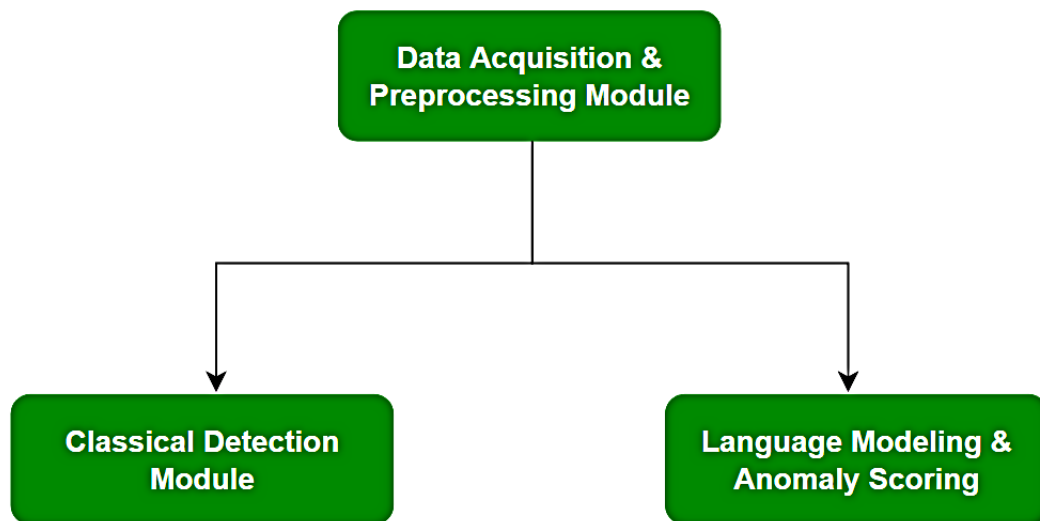
2. Funkcionalna cjelina za **klasičnu detekciju anomalija**:

- implementira tradicionalne algoritme prepoznavanja stršećih vrijednosti poput izolacijske šume, faktora lokalnog odstupanja, jednoklasnog stroja potpornih vektora i eliptične omotnice,

- trenira navedene metode na normalnim i filtriranim podacima iz skupa za učenje, a zatim ih primjenjuje na podatke iz skupa za testiranje s ciljem generiranja oznake odstupanja ili binarne klasifikacije.

3. Funkcionalna cjelina za **jezično modeliranje i pripadno označavanje anomalija**:

- pretvara numeričke prozore vremenskih serija u tekstualni oblik, provodi tokenizaciju i precizno prilagođava unaprijed naučene jezične modele (GPT-2, DistilBERT, T5 i XLNet),
- izračunava perpleksnost svakog prozora iz skupa za testiranje te postavlja razinu praga, pomoću kojega se identificiraju stršeće vrijednosti.



Slika 4.1. Dijagram koji prikazuje arhitekturu sustava na visokoj razini apstrakcije

Metode *extract_zip*, *build_smap_dict_for_classical* i *build_msl_dict_for_classical* upravljaju raspakiravanjem skupova podataka i konstruiranjem podatkovnih okvira *train* i *test*. Vremenske serije pohranjene su u rječnik. Redni broj kanala predstavlja ključ, a pripadne značajke i oznaka *is_anomaly* čine vrijednost pojedinog elementa u rječniku.

Za svaki kanal, podskup njegovih značajki dovodi se na ulaz klasičnih modela, koji potom generiraju binarne predikcije. Dobivena predviđanja pohranjuju se u novostvorene stupce *pred_iforest*, *pred_lof*, *pred_ocsvm* i *pred_elliptic* unutar istog podatkovnog okvira.

Kanali se dalje obrađuju stvaranjem tekstualnih prozora (kao što je opisano u poglavlju 3.). Jezični modeli precizno se prilagođavaju na prozorima iz skupa za treniranje, a vrijednosti perpleksnosti naknadno se računaju za prozore iz skupa za testiranje. Dobiveni rezultati preslikavaju se natrag na pojedinačne vremenske trenutke u izvornom obliku.

Iako će eksperimentalna evaluacija biti opisana u sljedećem odjeljku (5.), u ovome dijelu bitno je naglasiti da sustav prema dobivene predikcije iz klasičnih modela i predviđanja ostvarena korištenjem velikih jezičnih modela u isti podatkovni okvir, olakšavajući njihovu usporedbu i izvođenje zaključka u kasnijim poglavljima.

4.2. Implementacija jezičnih modela

Dio sustava temeljen na velikim jezičnim modelima koristi istaknute transformer arhitekture, unaprijed istrenirane na iznimno velikim količinama teksta. Zatim ih precizno prilagođava na numeričkim prozorima pretvorenim u tekstualni oblik, s ciljem detekcije anomalija. Nakon opisanog postupka, izračunava se perpleksnost za prozore s kojima se modeli još nisu susreli te se nakon toga određuje prag za označavanje potencijalnih anomalija. Ovo poglavlje fokusira se na konkretnu implementaciju navedenih modela i detaljno se razrađuje proces specifične prilagodbe za svaki odabrani model: *GPT-2*, *DistilBERT*, *T5-small* i *XLNet*.

4.2.1. *GPT-2*

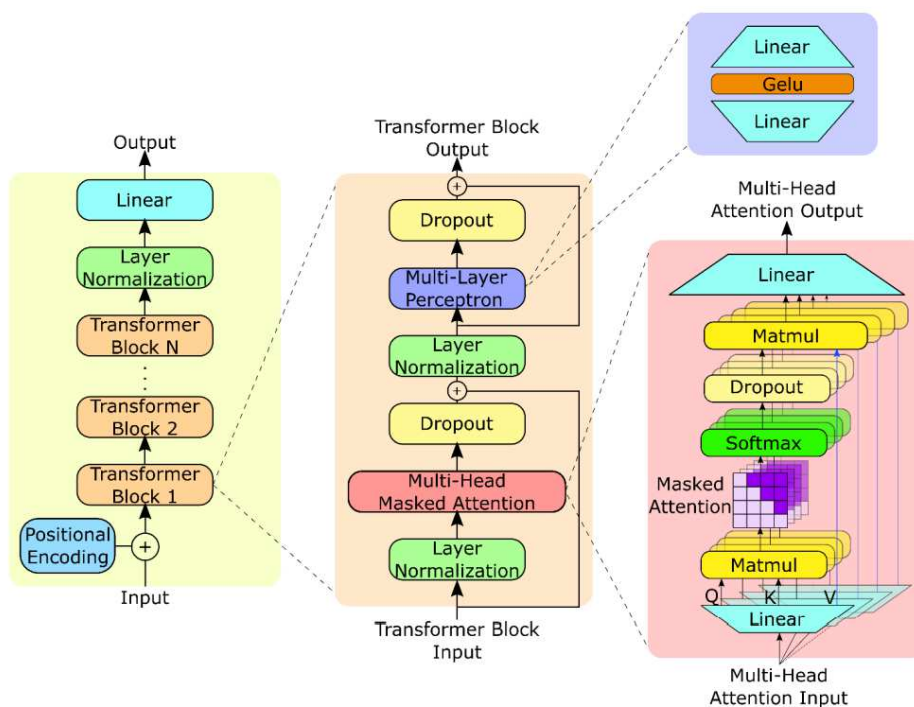
Općenito o modelu *GPT-2*

S razvojem generativnih modela velikih razmjera, obrada prirodnog jezika uspješno je napredovala u zadacima poput dopunjavanja teksta, prijevoda i sažimanja. Jedan od najutjecajnijih modela u toj domeni je *GPT-2* (*Generative Pretrained Transformer 2*), kojega je izumila američka tvrtka *OpenAI*. *GPT-2* je nenadzirani jezični model temeljen na transformatoru i kreiran prvenstveno za generiranje tekstualnog sadržaja. Za razliku od tradicionalnih modela obrade prirodnog jezika koji zahtijevaju postojanje označenih podataka za uspješno izvođenje pojedinih zadataka, *GPT-2* koristi nenadzirano prethodno treniranje na obilnim količinama teksta. Navedeni proces ostvaruje generiranje smislenog i kontekstualno bogatog teksta u različitim domenama ljudskog djelovanja. Opisani

model značajno je nadmašio prethodne generativne modele, postignuvši nevjerovatne sposobnosti generiranja ljudskog jezika.

GPT-2 temelji se na transformatorskoj arhitekturi, sadržavajući u sebi isključivo de-koder. Suprotno od dvosmjernog *BERT*-a, *GPT-2* radi na autoregresivan način, predviđajući jedan po jedan token slijeva nadesno. Ključna obilježja njegove arhitekture uključuju sljedećih pet značajki.

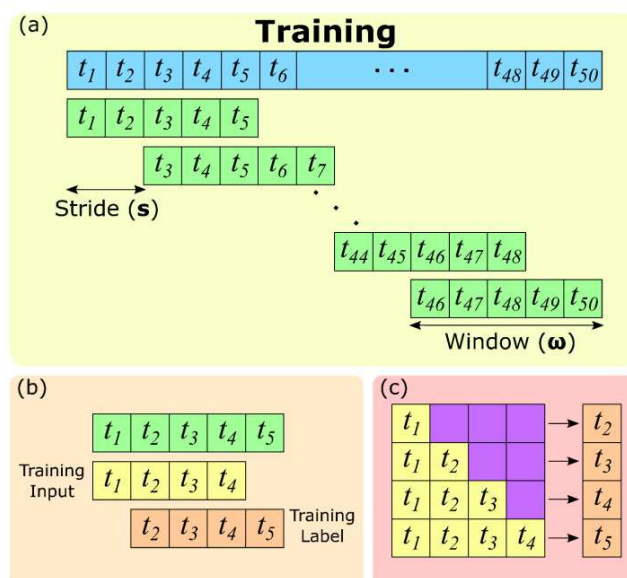
- **Višeslojni transformatorski blokovi:** model se sastoji od maksimalno 48 slojeva transformatorskih blokova (u svojoj najvećoj verziji).
- **Mehanizmi samopozornosti:** koristi višestruki mehanizam samopozornosti, dopuštajući modelu fokusiranje na relevantne dijelove ulaznog niza.
- **Funkcije normalizacije i aktivacije slojeva:** koristi aktivacijske funkcije *Layer-Norm* i *GELU* u svrhu poboljšanja stabilnosti procesa treniranja modela.
- **Tokenizacija *Byte Pair Encoding (BPE)*:** *GPT-2* upotrebljava metodu tokenizacije podnizova koja omogućuje učinkovito predstavljanje rijetkih nizova i izbjegava probleme s nedostatkom rječnika (engl. *out-of-vocabulary, OOV*).
- **Nema prethodno definiranih slojeva specifičnih za pojedine zadatke:** Za razliku od precizno prilagođenih modela, *GPT-2* ne primjenjuje slojeve prilagođene specifičnom zadatku, što ga čini jezičnim modelom opće namjene.



Slika 4.2. Dijagram koji prikazuje arhitekturu modela *GPT-2* (Izvor: [6])

Proces treniranja modela *GPT-2* sastoji se od dva koraka: nenadzirano prethodno treniranje i učenje bez primjera (engl. *zero-shot learning*) ili učenje s nekoliko primjera (engl. *few-shot learning*). Model je prvotno treniran na 40 GB tekstualnog sadržaja s interneta iz različitih izvora, pomoću jednostavnog i učinkovitog kauzalnog jezičnog modeliranja. Takav pristup omogućuje predviđanje sljedeće riječi u nizu, što je zapravo temeljni princip autoregresivnog modeliranja. Za razliku od ostalih modela koji su zahtijevali preciznu prilagodbu na određenim zadacima, *GPT-2* pokazao se učinkovitim u učenju bez ikakvih primjera. *Zero-shot learning* izvođenje je zadataka obrade prirodnog jezika bez eksplicitne precizne prilagodbe. S druge strane, *few-shot learning* odnosi se na postizanje visokih performansi uvjetovanjem samo nekoliko upita. Navedena sposobnost čini *GPT-2* pogodnim za različite zadatke generiranja i razumijevanja teksta.

Opisani model predstavio je značajan napredak u generativnim modelima obrade prirodnog jezika, dokazujući da prethodno nenadzirano treniranje velikih razmjera može omogućiti snažne sposobnosti učenja bez primjera ili učenja s nekoliko primjera. Njegovi etički problemi i računalni izazovi utemeljili su podlogu za daljnja usavršavanja u kasnijim modelima. Nakon modela *GPT-2*, tvrtka *OpenAI* razvila je *GPT-3* i kasnije *GPT-4*, nadograđujući njihovu arhitekturu naprednijim mogućnostima, strategijama precizne prilagodbe i poboljšanjima u činjeničnoj točnosti. Međutim, *GPT-2* i dalje ostaje temeljni model u istraživanju generiranja teksta umjetnom inteligencijom, utječući na razvoj snažnih jezičnih modela opće namjene koji se danas koriste.



Slika 4.3. Proces treniranja modela *GPT-2* (Izvor: [6])

Razred modela *GPT-2* i pripadni tokenizator

GPT-2 učitani je pomoću klase `AutoModelForCausalLM`, zajedno s objektom `AutoTokenizer` iz biblioteke *Hugging Face*.

```
from transformers import AutoTokenizer, AutoModelForCausalLM
model_name = "gpt2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token or tokenizer.unk_token
model.config.pad_token_id = tokenizer.pad_token_id
```

Primjer 2: Isječak koda koji prikazuje odabir modela *GPT-2* i pripadnog tokenizatora

Naredba `AutoTokenizer.from_pretrained(model_name)` preuzima *GPT-2* tokenizator podnizova koji dijeli decimalne brojeve u više tokena. Prethodno učenje modela ne zahtijeva ispunu za poravnanje niza, budući da je treniran na kontinuiranom tekstualnom sadržaju. Međutim, za skupnu obradu nizova promjenjive duljine moramo definirati varijablu `pad_token`. Izvođenjem naredbe `model.config.pad_token_id = tokenizer.pad_token_id` osigurava se ispravno rukovanje manjim nizovima različitih duljina, eliminirajući pogreške tijekom unaprijednog prolaza.

Postavljanje precizne prilagodbe za *GPT-2*

Precizna prilagodba modela na numeričkim prozorima odvija se slično kao i kod ostalih velikih jezičnih modela u ovom procesnom toku, ali uz korištenje kauzalnog jezičnog modeliranja.

```
from transformers import DataCollatorForLanguageModeling
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False
)
```

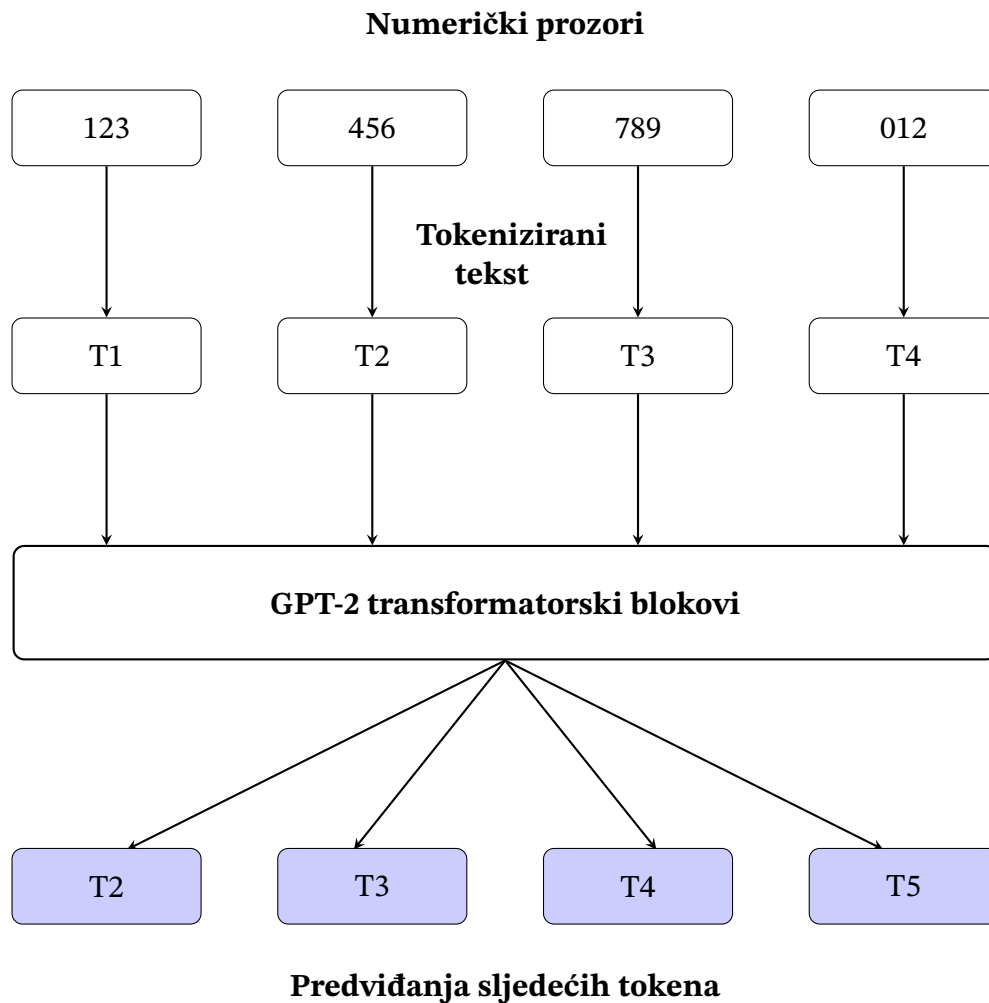
Primjer 3: Isječak koda koji prikazuje prvi korak precizne prilagodbe modela *GPT-2*

Naredba `mlm=False` signalizira da se ne maskiraju znakovni nizovi, već model uči predviđanjem svakog tokena na temelju svih prethodnih nizova (slijeva nadesno).

```
from transformers import Trainer, TrainingArguments
training_args = TrainingArguments(
    output_dir="./checkpoints_gpt2",
    num_train_epochs=2,
    per_device_train_batch_size=4,
    logging_steps=50,
    save_steps=100,
    seed=42
)
trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset
)
trainer.train()
```

Primjer 4: Isječak koda koji prikazuje drugi korak precizne prilagodbe modela *GPT-2*

GPT-2 često može razumno dobro konvergirati na malim skupovima numeričkih tekstualnih podataka u samo nekoliko epoha, iako optimalni broj može varirati ovisno o veličini skupa podataka. Naredbom `logging_steps=50` definira se ispis napretka procesa treniranja svakih pedeset koraka, pomažući u uklanjanju pogrešaka i praćenju izvršavanja modela. Također, postavljanjem varijable `save_steps=100` određuje se spremanje kontrolnih točaka svakih sto koraka, dopuštajući nastavak nakon iznenadne stanke te mogućnost analiziranja međustanja. Personalizirana klasa skupa podataka `SmappedLanguageModelingDataset` osigurava izradu tokeniziranih tekstualnih prozora. Svaka stavka u navedenom skupu sastoji se od rječnika ili tenzora koji sadrži identifikatore pojedinih tokena, oblikovanog prema postavljenoj vrijednosti varijable `block_size`.



Slika 4.4. Precizna prilagodba modela *GPT-2* na numeričkom tekstu

Prepoznavanje anomalija pomoću perpleksnosti u modelu *GPT-2*

Nakon što je *GPT-2* precizno prilagođen, sustav zaključuje o anomalijama računajući perpleksnost u svakom prozoru iz skupa za testiranje. Formalno, za tokenizirani prozor $z = (z_1, z_2, \dots, z_N)$, gubitak kauzalnog jezičnog modeliranja u modelu *GPT-2* je sljedeći:

$$L_{\text{CLM}}(z) = - \sum_{i=1}^N \log P(z_i | z_{<i}; \theta), \quad (4.1)$$

pri čemu $z_{<i} = (z_1, \dots, z_{i-1})$ predstavlja sve prethodne tokene, a θ označava parametre modela. Prosječna negativna logaritamska vjerojatnost definira se kao:

$$\ell(z) = \frac{1}{N} \sum_{i=1}^N -\log P(z_i | z_{<i}), \quad (4.2)$$

a perpleksnost se dobiva sljedećom formulom:

$$\text{PPL}(z) = \exp(\ell(z)) = \exp\left(\frac{1}{N} \sum_{i=1}^N -\log P(z_i | z_{<i})\right). \quad (4.3)$$

1. Podjela na manje dijelove

Ukoliko numerički prozor rezultira većim brojem tokena nego što *GPT-2* može obraditi u jednom prolazu (512 ili 1024, ovisno o inačici modela), izvršava se podjela prozora z na manje dijelove (engl. *chunks*). Gubitak svakog dijela procjenjuje se prema njegovoj duljini i kombinira s ostalim gubitcima u svrhu dobivanja globalnog prosjeka $\ell(z)$.

2. Određivanje praga

Nakon računanja perpleksnosti za svaki prozor iz skupa za testiranje, sustav identificira odstupanja uspoređivanjem dobivenih vrijednosti s postavljenim pragom (primjerice, 99. percentil). Prozor čija je vrijednost perpleksnosti znatno veća od ostalih implicira da je model "iznenađen" znakovnim nizom, što upućuje na zaključak o odstupanju numeričkih obrazaca od onih naučenih tijekom procesa treniranja. Takvi su prozori kandidati za upozorenja o anomalijama.

3. Integracija s prozorima vremenskih serija

Svaki prozor iz skupa za treniranje odgovara rasponu redaka u izvornom skupu podataka. Ako perpleksnost određenog prozora prelazi postavljenju vrijednost praga, svi redovi u pripadnom prozoru klasificirani su kao anomalije (`pred_llm_gpt2=1`).

```
def compute_perplexity_for_window_chunked(text, model, tokenizer,
device, chunk_size=512):
    encoding = tokenizer(text, return_tensors="pt",
        add_special_tokens=False)
    input_ids = encoding["input_ids"].squeeze(0).to(device)
    total_loss = 0.0
    total_tokens = 0
    start = 0
    while start < input_ids.size(0):
        end = min(start + chunk_size, input_ids.size(0))
        inputs = input_ids[start:end].unsqueeze(0)
        with torch.no_grad():
```

```

        outputs = model(inputs, labels=inputs)
        loss = outputs.loss
        segment_length = (end - start)
        total_loss += loss.item() * segment_length
        total_tokens += segment_length
        start = end

    avg_loss = total_loss / total_tokens
    ppl = torch.exp(torch.tensor(avg_loss))
    return ppl.item()

```

Primjer 5: Isječak koda koji prikazuje izračun perpleksnosti u modelu *GPT-2*

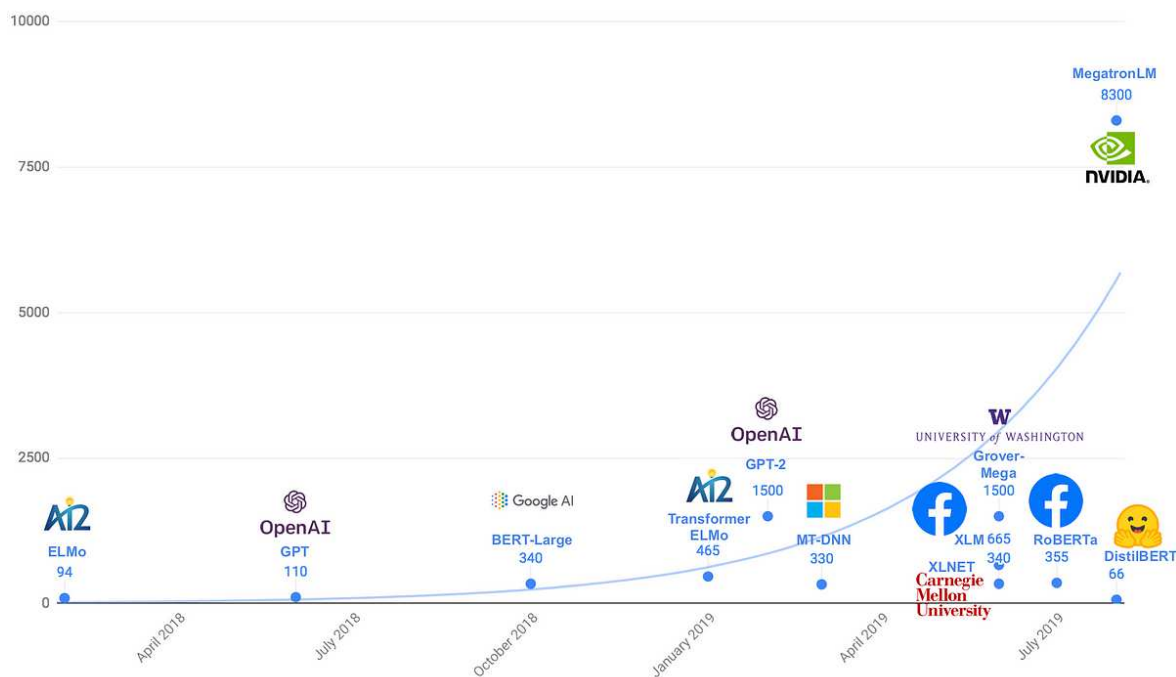
GPT-2 djeluje u obliku kauzalnog modela, predviđajući znakovne nizove isključivo slijeva nadesno. Suprotno od nekih drugih jezičnih modela, nema ugrađenu inicijalizaciju varijable `pad_token`, sugerirajući upotrebu ručnog definiranja navedene vrijednosti. Njegov proces treniranja slijedi standardne korake jezičnog modeliranja uz postavljanje vrijednosti `mlm=False`, što znači da se ne koristi maskirano jezično modeliranje poput onoga u modelu *DistilBERT*. Tijekom zaključivanja, perpleksnost se mjeri kroz cjelokupne prozore iz skupa za testiranje, koji se po potrebi dijele na manje segmente. Odstupanja se otkrivaju identificiranjem prozora s iznimno visokom perpleksnosti, klasificirajući ih kao anomalije. Opisana metodologija omogućuje učinkovito hvatanje vremenskog redoslijeda u numeričkom tekstu, što ga čini prikladnim odabirom za detekciju neobičnih segmenata vremenskih serija kada pojedini obrasci odstupaju od naučene distribucije modela.

4.2.2. *DistilBERT*

Općenito o modelu *DistilBERT*

S porastom već obučениh jezičnih modela velikih razmjera poput BERT-a, zadaci obrade prirodnog jezika doživjeli su značajna poboljšanja. Međutim, takvi modeli zahitjevaju obilne količine računalnih sredstava (slika 4.5.), što predstavlja veliki problem u slučajevima rubnog računarstva (engl. *edge computing*) ili u ograničenim računalnim okruženjima. U svrhu eliminacije navedenih poteškoća, francusko-američka tvrtka *Hugging Face* razvila je model *DistilBERT*, manju i bržu verziju BERT-a, čiji su detalji imple-

mentacije opisani u znanstvenom radu [7].



Slika 4.5. Potreban broj parametara za uspješno izvođenje pojedinih jezičnih modela (Izvor: [7])

DistilBERT je osmišljen da zadrži 97% BERT-ovih mogućnosti razumijevanja jezika, uz smanjenje veličine modela za 40% i povećanje brzine zaključivanja za 60%. To se postiže kroz proces pod nazivom "destilacija znanja", koji se primjenjuje u fazi prije treniranja modela.

Destilacija znanja tehnika je kompresije u kojoj manji model "učenik" pokušava oponašati veći model "učitelj" (ili skup modela). Osnovni je učenikov cilj aproksimirati ponašanje učitelja, umjesto treniranja isključivo na označenim podacima. Osim korištenja uobičajenog gubitka unakrsne entropije između predviđenih i stvarnih oznaka, destilacija znanja uvodi distribuciju vjerojatnosti naslijeđenu iz učiteljskog modela. Funkcija gubitka korištena u modelu *DistilBERT* kombinacija je sljedećih triju gubitaka.

1. **Gubitak maskiranog jezičnog modeliranja (L_{MLM}):** potječe iz originalnog procesa učenja modela BERT, u kojemu se nasumične riječi maskiraju i predviđaju (opisano u potpoglavlju 3.3.3.).
2. **Gubitak destilacije (L_{CE}):** učenički model istreniran je da odgovara vjerojatnosnoj distribuciji učiteljskog modela.
3. **Gubitak ugrađenog kosinusa (L_{COS}):** poravnava skrivene vektore stanja učenič-

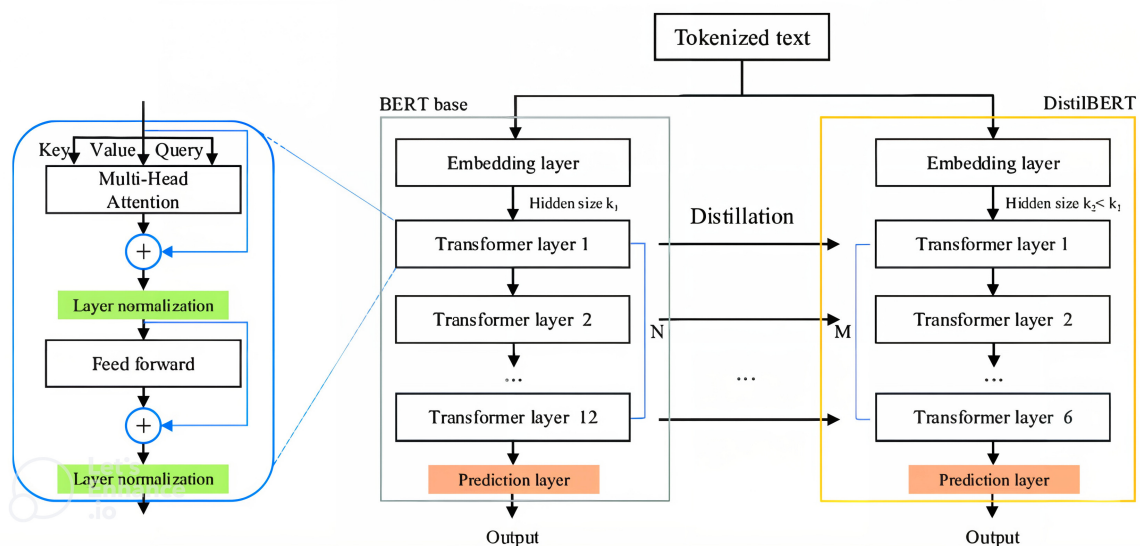
kog i učiteljskog modela.

Konačna funkcija gubitka povezuje tri navedene vrijednosti s ciljem zadržavanja maksimalne količine informacija, uz smanjenje složenosti modela.

Sama arhitektura DistilBERT-a vrlo je slična BERT-u, ali uz određena pojednostavljenja:

- broj transformatorskih slojeva smanjen je za 50%,
- uklonjeni su ugrađeni znakovni vektori i sloj sažimanja,
- proces treniranja koristi dinamičko maskiranje u svrhu postizanja bolje generalizacije (slično modelu RoBERTa),
- učenički model inicijaliziran je korištenjem svakog drugog sloja iz učiteljskog modela.

Ove izmjene rezultiraju dobivanjem lakšeg modela s minimalnim pogoršanjem performansi, uz značajno poboljšanje efikasnosti. DistilBERT zapravo predstavlja veliki korak prema učinkovitim i pristupačnim modelima obrade prirodnog jezika, prilagođavajući zahtjevne jezične modele za primjenu u svakodnevnom životu. Shematski prikaz opisanog postupka vidljiv je na slici 4.6.



Slika 4.6. Arhitektura modela DistilBERT uz označeni proces destilacije (Izvor: [8])

Razred modela *DistilBERT* i pripadni tokenizator

U biblioteci *Hugging Face Transformers*, *DistilBERT* se može učitati pomoću klase *AutoModel* koja podržava maskirano jezično modeliranje. Odgovarajući tokenizator osi-

gurava pravilnu predobradu teksta.

```
from transformers import AutoTokenizer, AutoModelForMaskedLM
model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForMaskedLM.from_pretrained(model_name)
model.config.pad_token_id = tokenizer.pad_token_id
```

Primjer 6: Isječak koda koji prikazuje odabir modela *DistilBERT* i pripadnog tokenizatora

Klasa *AutoTokenizer* automatski dohvaća ispravni tokenizator za model *distilbert-base-uncased*, osiguravajući dosljedan proces tokenizacije znakovih podnizova. Nakon toga, *AutoModelForMaskedLM* učitava unaprijed dobivene težine modela DistilBERT, specifično konfigurirane za maskirano jezično modeliranje, umjesto moguće sekvencijske klasifikacije ili predviđanja sljedeće rečenice. DistilBERT najčešće uključuje prethodno definiranje tokena <pad>, ali u ovom slučaju eksplicitno je postavljen kako bi se izbjeglo dobivanje pogrešaka prilikom pokretanja programskog koda.

Postavljanje precizne prilagodbe za *DistilBERT*

Precizna prilagodba opisanog modela za detekciju anomalija sastoji se od dva koraka. Prvi korak je inicijalizacija varijable *data_collator*.

```
from transformers import DataCollatorForLanguageModeling
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=True,
    mlm_probability=0.15
)
```

Primjer 7: Isječak koda koji prikazuje prvi korak precizne prilagodbe modela *DistilBERT*

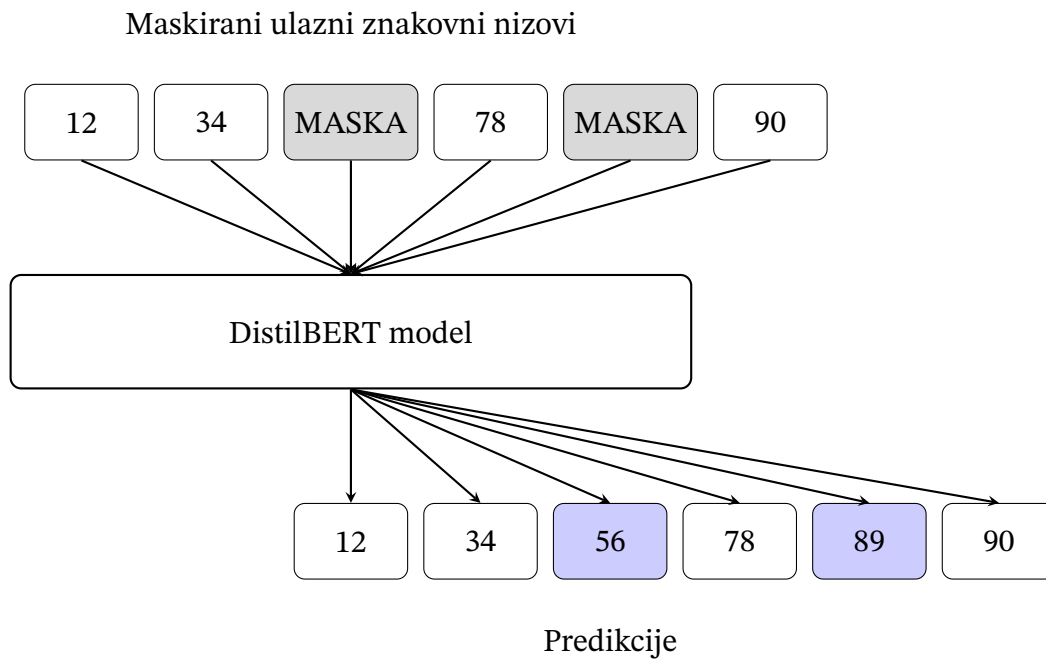
U svakoj iteraciji treniranja, 15% tokena zamijenjeno je maskom, nasumičnim znakovima ili je ostavljeno u izvornom obliku (prateći originalnu shemu učenja modela BERT). Na ovaj način pokušava se naučiti model na predviđanje maskiranih tokena iz okolnog konteksta, omogućujući učinkovito prepoznavanje dvosmjernih obrazaca u numeričkom tekstu.

Upravljanje iteracijama kroz manje serije te ažuriranje gradijenata i kontrolnih točaka pomoću biblioteke *Hugging Face Transformer* čine drugi korak precizne prilagodbe modela.

```
from transformers import Trainer, TrainingArguments
training_args = TrainingArguments(
    output_dir="./checkpoints_distilbert",
    num_train_epochs=2,
    per_device_train_batch_size=4,
    save_steps=100,
    logging_steps=50,
    seed=42
)
trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset
)
trainer.train()
```

Primjer 8: Isječak koda koji prikazuje drugi korak precizne prilagodbe modela DistilBERT

Varijabla `train_dataset` predstavlja skup podataka iz okvira *PyTorch*, u kojemu je svaki element zapravo tokenizirani prozor numeričkih podataka. Programski kod u primjeru 8 osigurava da se tekst svakog uzorka pretvori u slijed znakovnih podnizova (do definirane veličine `block_size`), pohranjen u varijablu `input_ids`. Postavljanje vrijednosti `mlm=True` u primjeru 7 služi kao uputa modelu da se znakovi maskiraju prema definiranoj vjerojatnosti `mlm_probability=0.15`, održavajući dosljednost načina obrade podataka tijekom cijelog procesa učenja.



Slika 4.7. Predviđanje maskiranih znakovnih nizova na temelju konteksta u modelu DistilBERT

Prepoznavanje anomalija pomoću perpleksnosti u modelu *DistilBERT*

Unatoč DistilBERT-ovom maskiranju tijekom obuke, zaključivanje o detekciji stršećih vrijednosti uključuje korištenje čitavog numeričkog teksta bez umjetnog uvođenja maski. Proces identificiranja odstupanja sastoji se od sljedeća četiri koraka.

1. Nemaskirani ulaz

Tijekom ispitivanja, svaki prozor w duljine N šalje se na ulaz u DistilBERT, bez dodatnih modifikacija. Unaprijednim prolaskom modela, uz zadane vrijednosti `labels=input_ids`, izračunava se negativna logaritamska vjerojatnost za svaki znakovni niz, prema formuli:

$$\ell(w) = \frac{1}{N} \sum_{i=1}^N -\log P(z_i | z_{\setminus i}; \theta), \quad (4.4)$$

pri čemu $z_{\setminus i}$ predstavlja druge tokene koji pomažu u predviđanju z_i . Unutar dvosmjernog okvira u modelu DistilBERT, svi nemaskirani tokeni u pripadnom prozoru mogu pružiti određene informacije o kontekstu analiziranog niza.

2. Računanje perpleksnosti

Prosječna negativna logaritamska vjerojatnost $\ell(w)$ potencirana je u svrhu dobiva-

nja perpleksnosti, kao što je prikazano u formuli:

$$\text{PPL}(w) = \exp(\ell(w)). \quad (4.5)$$

Visoka vrijednost perpleksnosti ukazuje na činjenicu da DistilBERT smatra prozor w neuobičajenim ili nedosljednim s naučenim obrascima iz procesa treniranja modela, što upućuje na prisutnost potencijalne anomalije. Niska vrijednost perpleksnosti implicira da su numerički znakovni nizovi iz prozora w dobro usklađeni s unaprijed naučenim normalnim obrascima.

3. Podjela na manje dijelove

Ako prozor u tokeniziranom obliku premašuje maksimalno dozvoljenu duljinu niza (primjerice, 512 tokena), izvršava se njegova podjela na manje dijelove. Vrijednosti gubitaka iz svakog od navedenih dijelova međusobno se zbrajaju i računa se njihov prosjek radi dobivanja globalnog $\ell(w)$.

4. Određivanje praga

Konačno, sustav prikuplja dobivene rezultate perpleksnosti za sve prozore iz skupa za testiranje, određuje vrijednost praga (primjerice, 99. percentil), i označava prozore čije vrijednosti prelaze navedenu granicu. Svaki vremenski korak unutar označenog prozora klasificira se kao anomalija (`pred_llm_distilbert=1`), prema formuli:

$$\hat{y}(w) = \begin{cases} 1, & \text{ako } \text{PPL}(w) \geq \tau, \\ 0, & \text{inače,} \end{cases} \quad (4.6)$$

pri čemu τ označava vrijednost postavljenog praga temeljenog na percentilu.

Ukratko, DistilBERT-ovo maskirano jezično modeliranje pruža dvosmjernu perspektivu u analizi numeričkog teksta, diskretno prepoznavajući multivarijatne ovisnosti značajki u svakom prozoru. Tijekom procesa precizne prilagodbe, sustav nasumično maskira 15% znakovnih nizova s ciljem postizanja svojstva otpornosti u treniranju modela. Prilikom zaključivanja, perpleksnost se izračunava na potpuno nemaskiranom ulaznom nizu, rezultirajući visokim vrijednostima za prozore čiji sadržaj odstupa od uobičajenih obrazaca. Opisana analiza zatim se integrira u cjelokupni procesni tok programa, pohranjujući dobivene predikcije stršećih vrijednosti zajedno s rezultatima ostalih jezičnih

modela i klasičnih metoda otkrivanja anomalija.

4.2.3. T5

Općenito o modelu T5

Većina tradicionalnih modela obrade prirodnog jezika osmišljena je za specifičnu namjenu, zahtijevajući prilagođenu arhitekturu i strategiju učenja za svaku pojedinu domenu. Navedeni nedostatak jedinstvenog okvira doveo je do neučinkovitosti u razvoju i implementaciji velikog broja različitih modela. Iz tog razloga, organizacija *Google Research* predstavila je snažan i prilagodljiv novi model koji preoblikuje sve zadatke obrade prirodnog jezika u format *text-to-text*. Umjesto tretiranja različitih *NLP* problema zasebno, model *T5* (engl. *Text-to-Text Transfer Transformer*) pretvara svaki zadatak u problem generiranja teksta. Na taj način omogućuje se istoj arhitekturi obrađivanje različitih zahtjeva poput prijevoda, sažimanja, odgovaranja na postavljena pitanja i klasifikacije. *T5* postiže vrhunsku izvedbu na višestrukim referentnim testovima obrade prirodnog jezika, dok uspijeva zadržati skalabilnu *end-to-end* strukturu koja pojednostavljuje treniranje i implementaciju modela.

Za razliku od modela kao što je *BERT*, koji se treniraju korištenjem različitih funkcija cilja ovisno o specifičnom zadatku, *T5* svaki *NLP* zadatak tretira kao problem generiranja teksta. Takav pristup pruža dosljednu paradigmu učenja, pri čemu su ulaz i izlaz uvijek strukturirani u obliku tekstualnog sadržaja.

Zadatak	Ulaz	Izlaz
Strojno prevođenje	"translate English to French: The cat is sleeping."	"Le chat dort."
Sažimanje teksta	"summarize: The research paper discusses NLP models."	"NLP models are discussed."
Odgovaranje na pitanja	"question: Who developed T5? context: T5 was introduced by Google Research."	"Google Research"
Analiza sentimenta	"classify sentiment: I love this movie!"	"positive"

Tablica 4.1. Primjeri formulacije *text-to-text*

Navedeni format čini model *T5* vrlo svestranim i eliminira potrebu za specifičnim arhitekturama za pojedini zadatak.

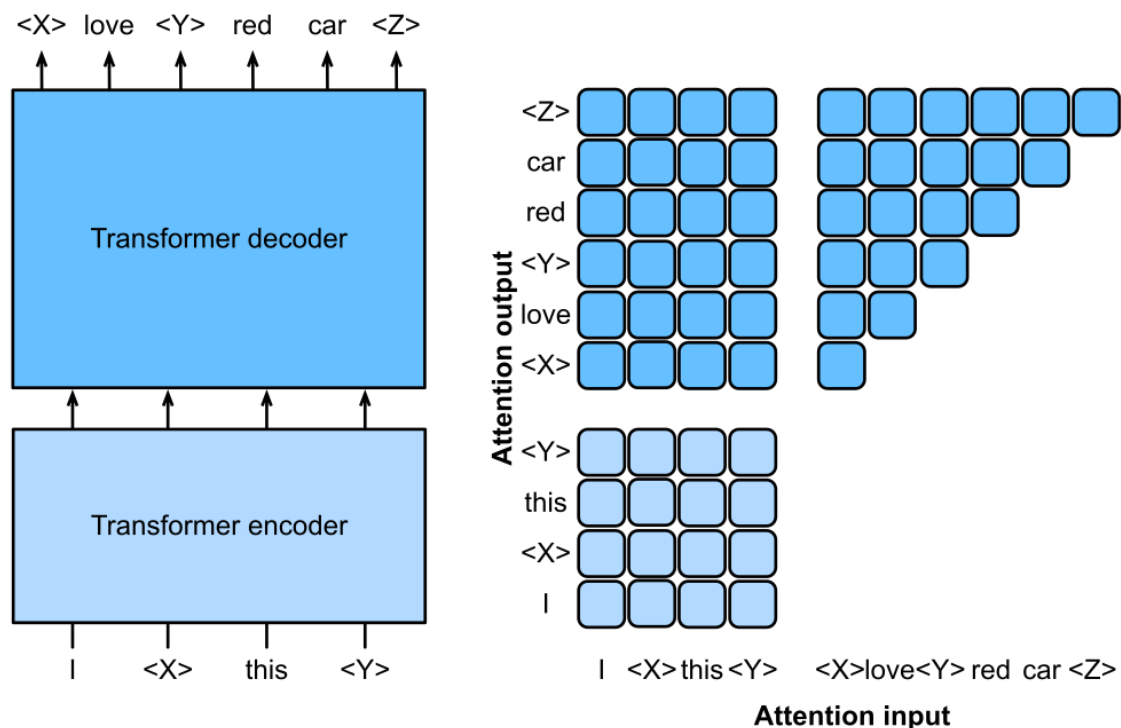
Model *T5* prethodno je uvježban na velikoj količini tekstualnih podataka koristeći pristup uklanjanja šuma pod nazivom *span corruption*. Umjesto maskiranja pojedinačnih tokena kao u *BERT*-u, *T5* nasumično uklanja dijelove teksta i traži od modela predviđanje sadržaja koji nedostaje. Primjer:

- **Ulaz:** "The <X> was introduced by Google <Y>."
- **Izlaz:** "<X> T5 model <Y> Research."

Ova metoda pridonosi razvoju snažnog razumijevanja jezičnih struktura, istovremeno zadržavajući mogućnost tumačenja dvosmjernog konteksta. Nakon prethodnog treniranja, *T5* se precizno prilagođava na specifične zadatke u kojima se označeni podaci koriste prilikom učenja modela za proces sažimanja, odgovaranja na postavljena pitanja ili klasifikaciju.

Model *T5* slijedi arhitekturu transformatora od sekvence do sekvence (engl. *sequence-to-sequence*, *Seq2Seq*), inspiriranu izvornim modelom *Transformer* ([9]). Temeljna obilježja navedene arhitekture opisana su u sljedećih pet natuknica.

- **Struktura koder-dekoder:** za razliku od *GPT* modela koji koristi samo dekoder ili *BERT* modela koji koristi samo koder, *T5* kombinira obje strukture s ciljem postizanja snažnijeg generiranja tekstualnog sadržaja.
- **Mehanizam samopozornosti:** upotrebljava mehanizam višestruke samopozornosti za korake kodiranja i dekodiranja.
- **Mreže s unaprijednim prosljeđivanjem (engl. *feedforward networks*):** primjenjuje potpuno povezane slojeve s *ReLU* aktivacijskim funkcijama.
- **Normalizacija slojeva i isključivanje jedinica (engl. *dropout*):** osigurava stabilan proces treniranja i smanjuje prenaučenosť modela.
- **Tokenizacija pomoću algoritma *SentencePiece*:** koristi metodu *Byte-Pair Encoding (BPE)* s efikasnim rječnikom.



Slika 4.8. Prikaz strukture koder-dekoder u modelu *T5* (Izvor: [10])

Razred modela *T5* i pripadni tokenizator

Model *T5-small* moguće je učitati pomoću klase `AutoModelForSeq2SeqLM` iz biblioteke *Hugging Face*. Njegov odgovarajući tokenizator, `AutoTokenizer`, upravlja podjelom tekstualnog sadržaja na manje jedinice.

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
model_name = "t5-small"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
model.config.pad_token_id = tokenizer.pad_token_id
```

Primjer 9: Isječak koda koji prikazuje odabir modela *T5-small* i pripadnog tokenizatora

Klasa `AutoTokenizer` dohvaća rječnik i logiku tokenizacije za model *T5*, dijeleći svaki brojevni niz u manje podnizove na točno određen način, opisan u dobivenom rječniku. Nadalje, klasa `AutoModelForSeq2Seq` omogućuje korištenje sučelja za pristup *sequence-to-sequence*, odnosno koderu koji čita ulaze i dekoderu koji generira izlaze. Prema zadanim postavkama, model očekuje ulazni niz i "prefiks zadatka" koji je opcionalan (primjerice, "prevedi engleski na njemački"). U pripadnom brojevnom kontek-

stu moguće je izravno unijeti numerički tekst bez specijaliziranog prefiksa. Naredba `model.config.pad_token_id = tokenizer.pad_token_id` osigurava ispravno rukovanje vremenskim serijama različitih duljina. *T5* definira vrijednost tokena `<pad>` prema zadanim postavkama, stoga navedena instrukcija jednostavno potvrđuje usklađenost između tokenizatora i postavki modela.

Postavljanje precizne prilagodbe za *T5*

Cilj je naučiti model *T5* na rekonstrukciju istih numeričkih tekstualnih prozora koje prima na ulazu. Ovim pristupom pokušava se natjerati model da nauči reprezentaciju "normalnih" numeričkih obrazaca. Sva odstupanja u procesu predviđanja dovode do većeg gubitka rekonstrukcije, a samim time dobiva se i veća vrijednost perpleksnosti.

Pozivom funkcije `model(inputs, labels=inputs)` efektivno se zahtijeva ispis istih tokena koje je model dobio na ulazu (samorekonstrukcija). Gubitak učenja zapravo je zbroj unakrsne entropije za svaki izlazni token:

$$\ell(w) = - \sum_{t=1}^N \log P(z_t | z_{<t}; \theta), \quad (4.7)$$

pri čemu w predstavlja numerički tekstualni prozor duljine N , a θ označava parametre modela *T5*. Koder unosi cijeli prozor, dok dekoder predviđa svaki token korak po korak.

- **Koder:** čita sve tokene, stvarajući skrivena stanja koja obuhvaćaju kontekst cjelokupnog brojevnog prozora.
- **Dekoder:** u svakom koraku t , stvara jedan token, uvjetovan prethodno predviđenim tokenima i skrivenim stanjima kodera.

Za zadatke tipa *seq2seq*, specijalizirani agregator podataka može obraditi ulazne nizove i pripadne oznake. Ukoliko se ulazni tekst tretira i kao izvor i kao željeni cilj, moguće je inicijalizirati varijablu `data_collator` na sljedeći način.

```
from transformers import DataCollatorForSeq2Seq
data_collator = DataCollatorForSeq2Seq(
    tokenizer=tokenizer,
    model=model
)
```

Primjer 10: Isječak koda koji prikazuje prvi korak precizne prilagodbe modela *T5*

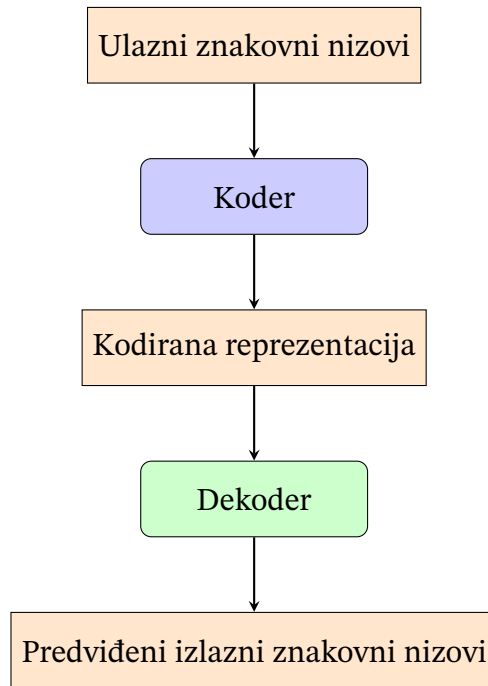
Alternativno, postoji i mogućnost korištenja klase `DataCollatorForLanguageModeling`, ali `DataCollatorForSeq2Seq` precizno je prilagođen paradigmi koder-dekoder, osiguravajući ispravnu ispunu ulaza i izlaza.

```
from transformers import Trainer, TrainingArguments
training_args = TrainingArguments(
    output_dir="./checkpoints_t5_small",
    num_train_epochs=2,
    per_device_train_batch_size=4,
    logging_steps=50,
    save_steps=100,
    seed=42
)
trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset
)
trainer.train()
```

Primjer 11: Isječak koda koji prikazuje drugi korak precizne prilagodbe modela *T5*

Varijabla `train_dataset` sarži prozore s numeričkim tekstom. Svaki uzorak obuhvaća "ulaz" i "oznaku" za model *T5*, odražavajući cilj rekonstrukcije. Određivanje broja epoha (`num_train_epochs`) i veličine paketa (`per_device_train_batch_size`) ovisi o količini podataka i dostupnoj *GPU* memoriji. U ovom slučaju odabrane su dvije epohe i veličina paketa postavljena je na četiri, što često čini izvedivu početnu točku za dosljedno izvršavanje potrebnih algoritama. Također, bitno je naglasiti da model *T5* obično koristi spomenuti prefiks zadatka. Međutim, kod rekonstruiranja numeričkih podataka taj dio nije nužan. Dovoljno je samo unijeti numerički niz kao ulaz i koristiti isti niz kao ciljnu vrijednost. Navedeni pristup upućuje na zaključak da model može učiti samo na temelju

ulaznih podataka i njihovih odgovarajućih oznaka, bez dodatnih instrukcija.



Slika 4.9. Shematski prikaz predikcije znakovnih nizova na temelju *koder-dekoder* arhitekture u modelu *T5*

Prepoznavanje anomalija pomoću perpleksnosti u modelu *T5*

Nakon treniranja modela *T5*, sustav vrednuje koliko dobro može rekonstruirati nemaskirane prozore iz skupa za testiranje. Ponovno se poziva funkcija `model(inputs, labels=inputs)` u načinu evaluacije (`model.eval()`), čime se dobiva ukupan gubitak unakrsne entropije. Perpleksnost se tada računa na sljedeći način:

$$PPL(w) = \exp\left(\frac{1}{N} \sum_{t=1}^N -\log P(z_t | z_{<t}; \theta)\right), \quad (4.8)$$

pri čemu je N broj tokena u prozoru w . U strukturi koder-dekoder unutar modela *T5* vrijede sljedeća četiri svojstva.

1. **Koder:** obrađuje čitavi numerički prozor.
2. **Dekoder:** predviđa svaki token uzastopno, pozivajući se na izlaze iz kodera.
3. **Gubitak:** izračunava se usporedbom svakog predviđenog tokena sa stvarnom oznakom z_t .
4. **Računanje prosjeka i podjela na manje dijelove:** ako tokenizirana duljina pro-

zora prelazi ograničenje (primjerice, 512 tokena), može se podijeliti na manje segmente, slično kao kod modela *GPT-2*. Zatim se zbraja i izračunava prosjek u svrhu dobivanja vrijednosti perpleksnosti za cijeli prozor.

Sažeto rečeno, model *T5* koristi strukturu koder-dekoder, u kojoj koder čita cijeli prozor, a dekoder ga rekonstruira token po token. Pokušava se rekonstruirati ulazni niz, pri čemu postavljanje varijable `labels=input_ids` osigurava da model uči distribuciju preko uobičajenih brojevni sekvenci. Tijekom zaključivanja, gubitak unakrsne entropije izračunava se prilikom pokušaja rekonstrukcije prozora iz skupa za testiranje te se potom pretvara u mjeru perpleksnosti. Veće vrijednosti perpleksnosti ukazuju na naglašeno odstupanje od distribucije dobivene u procesu treniranja. U procesu integracije vremenskih serija, izračunate vrijednosti uspoređuju se s postavljenim pragom, stvarajući oznaku anomalije za svaki pojedini prozor koji zadovoljava taj uvjet. Prilagodбом modela za numeričku rekonstrukciju prozora, iskorištava se njegova otporna *seq2seq* arhitektura, koja potencijalno može obuhvatiti složenije međuovisnosti od strogo jednosmjernih ili maskiranih modela.

4.2.4. *XLNet*

Općenito o modelu *XLNet*

Generalizirani autoregresijski jezični model pod nazivom *XLNet* predstavili su istraživači iz organizacije *Google Brain* i sveučilišta *Carnegie Mellon*. *XLNet* kombinira prednosti autoregresijskih (*AR*) modela poput *GPT*-a i autokodirajućih (*AE*) modela poput *BERT*-a, pokušavajući eliminirati njihove nedostatke. Navedeni model nadmašuje *BERT*-ove rezultate čak na nekoliko referentnih skupova podataka, uključujući *General Language Understanding Evaluation (GLUE)*, *Stanford Question Answering Dataset (SQuAD)* i *Reading Comprehension Dataset (RACE)*. Uspješna izvedba opisanog modela postiže se korištenjem mehanizama temeljenih na permutaciji i ponavljanju, koji poboljšavaju njegovu sposobnost modeliranja dugoročnih međuovisnosti u podacima.

XLNet uvodi novi cilj treniranja modela koji se temelji na permutacijskom jezičnom modeliranju (engl. *Permutation Language Modeling, PLM*). Za razliku od *BERT*-a koji nasumično maskira tokene, *XLNet* uči iz svih mogućih redoslijeda tokena, osiguravajući prepoznavanje dvosmjernih ovisnosti među podacima uz zadržavanje prednosti autore-

gresijskog procesa treniranja. Umjesto predviđanja maskiranih znakovnih nizova, ovaj model premješta ulazne tokene nasumičnim redoslijedom i pokušava predvidjeti svaki token na temelju njegovih prethodnih znakova. To omogućuje *XLNet*-u učenje ovisnosti među svim pozicijama bez narušavanja ulaznih podataka, što je ograničenje *BERT*-ove metode maskiranog jezičnog modeliranja. Model se trenira korištenjem redoslijeda faktORIZACIJE, što znači da ulazni niz ostaje isti, uz istovremeno učenje ovisnosti među različitim položajima, kroz više iteracija. Primjer analize ulaznog niza "*The cat sat on the mat.*" vidljiv je u tablici 4.2.

Pristup	Primjer
BERT-ovo maskirano jezično modeliranje	Ulaz: "The cat [MASK] on the mat." Model predviđa: "sat"
Prva instanca treniranja modela XLNet	"The sat cat on the mat"
Druga instanca treniranja modela XLNet	"on the mat cat The sat"
Treća instanca treniranja modela XLNet	"cat mat The on sat the"

Tablica 4.2. Usporedba maskiranog i permutacijskog jezičnog modeliranja

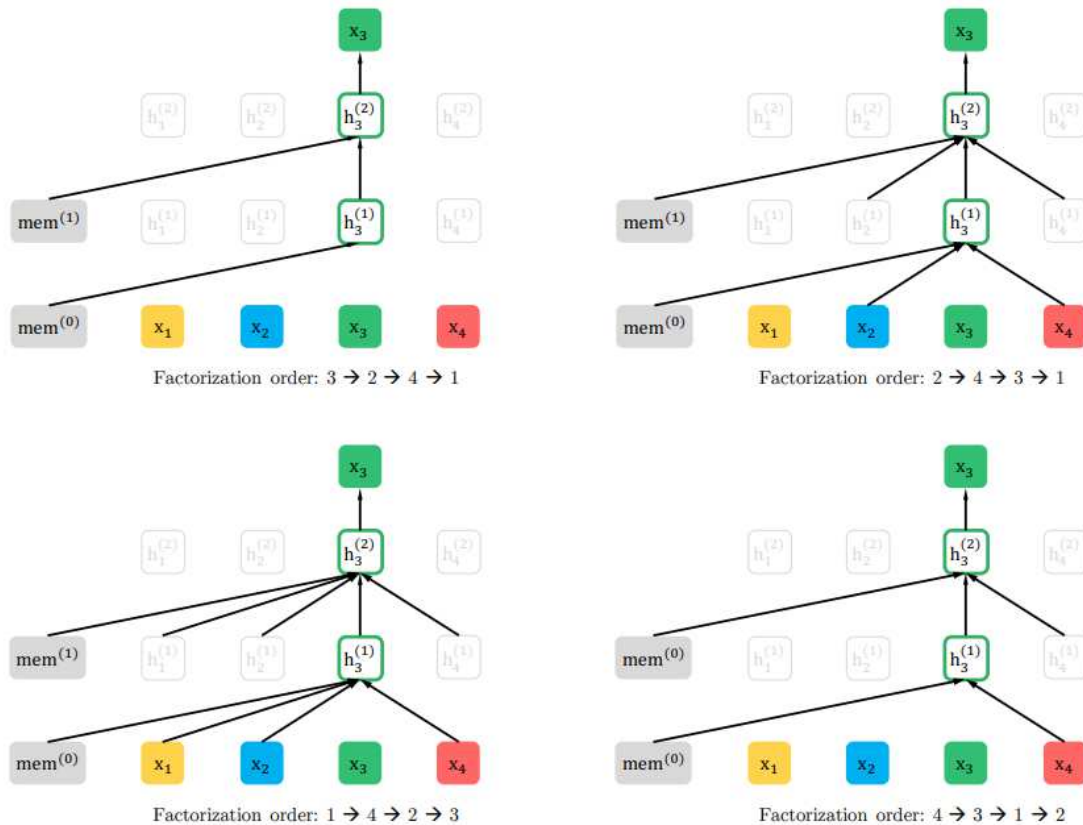
Permutiranjem redoslijeda tokena uz istovremeno autoregresijsko predviđanje, *XLNet* uči dvosmjerne odnose među nizovima bez potrebe za maskiranjem.

XLNet se oslanja na transformatorsku arhitekturu, sličnu modelima *BERT* i *GPT-2*, ali uz određena poboljšanja.

- **Treniranje temeljeno na permutaciji:** uči dvosmjerni kontekst bez maskiranih tokena.
- **Relativno pozicijsko kodiranje:** zamjenjuje apsolutne ugradnje položaja s relativnim pozicijskim kodiranjem, poboljšavajući dugotrajno modeliranje ovisnosti.
- **Mehanizam dvosmjerne samopozornosti:** koristi pozornost sadržajnog toka za stvarne tokene i pozornost upitnog toka za ciljana predviđanja, optimizirajući paralelizaciju.
- **Transformator s proširenom memorijom:** inspiriran modelom *Transformer-XL*, *XLNet* objedinjuje mehanizme ponavljanja, pomažući pri modeliranju dugoročnih ovisnosti na učinkovitiji način od *BERT*-a.

Navedene modifikacije rezultiraju postizanjem bolje generalizacije, usavršenom reprezentacijom tekstualnog sadržaja i unaprijeđenom izvedbom na referentnim vrijednos-

tima obrade prirodnog jezika.



Slika 4.10. Ilustracija permutacijskog cilja učenja za predviđanje tokena x_3 uz različite redoslijede faktorizacije (Izvor: [11])

Razred modela *XLNet* i pripadni tokenizator

Sljedeći isječak pokazuje standardne korake učitavanja modela *XLNet* sa sučeljem za permutacijsko jezično modeliranje.

```
from transformers import AutoTokenizer, AutoModelForPreTraining
model_name = "xlnet-base-cased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForPreTraining.from_pretrained(model_name)
model.config.pad_token_id = tokenizer.pad_token_id
```

Primjer 12: Isječak koda koji prikazuje odabir modela *XLNet* i pripadnog tokenizatora

Klasa `AutoModelForPreTraining` osigurava da se *XLNet* koristi kako je izvorno naučen, provođenjem permutacijskog jezičnog modeliranja. Nadalje, upotrebom razreda `AutoTokenizer` dohvaća se rječnik i primjenjuje strategija tokeni-

zacije za model *"xlnet-base-cased"*. Brojčani ulaz može se podijeliti u više manjih podnizova, osim ako se ne koristi prilagođeni numerički tokenizator. Naredbom `model.config.pad_token_id=tokenizer.pad_token_id` ponovno se osigurava ispravno upravljanje nizovima različitih duljina, pri čemu su kraće sekvence nadopunjene radi pravilnog usklađivanja tijekom treniranja modela i procesa zaključivanja.

Postavljanje precizne prilagodbe za *XLNet*

Cilj permutacijskog jezičnog modeliranja kod *XLNet*-a preoblikuje redosljed faktori-zacije tokena, učinkovito prepoznavajući i naprijed i unatrag usmjerene ovisnosti unutar samo jednog procesa. Gubitak opisanog modela računa se na sljedeći način:

$$L_{\text{XLNet}}(z) = - \sum_{i=1}^N \log P(z_i | z_{\setminus i}^{(\pi)}; \theta), \quad (4.9)$$

pri čemu π označava permutaciju položaja tokena, a $z_{\setminus i}^{(\pi)}$ predstavlja sve tokene osim z_i u permutiranom kontekstu.

```
from transformers import DataCollatorForPermutationLanguageModeling,
    Trainer, TrainingArguments
data_collator = DataCollatorForPermutationLanguageModeling(
    tokenizer=tokenizer,
    plm_probability=1.0,
    max_span_length=5
)
training_args = TrainingArguments(
    output_dir="./checkpoints_xlnet",
    num_train_epochs=2,
    per_device_train_batch_size=4,
    logging_steps=50,
    save_steps=100,
    seed=42
)
trainer = Trainer(
    model=model,
```

```

args=training_args,
data_collator=data_collator,
train_dataset=train_dataset
)
trainer.train()

```

Primjer 13: Isječak koda koji prikazuje preciznu prilagodbu modela *XLNet*

Klasa `DataCollatorForPermutationLanguageModeling` odgovorna je za pripremu tekstualnih podataka za permutacijsko jezično modeliranje. Naredbom `plm_probability=1.0` osigurava se da svaki ulazni niz prolazi kroz obradu tokena temeljenu na permutaciji. Određivanje vrijednosti `max_span_length=5` definira maksimalnu duljinu kontinuiranih raspona znakovnih nizova koji se koriste za predviđanje, omogućujući modelu učenje duljih ovisnosti među analiziranim podacima.

Prepoznavanje anomalija pomoću perpleksnosti u modelu *XLNet*

Za razliku od modela *GPT-2* koji koristi kauzalno jezično modeliranje, *XLNet* upotrebljava opisano permutacijsko jezično modeliranje. To znači da umjesto predviđanja sljedećeg tokena u nepromjenjivom redosljedju slijeva nadesno, on dinamički mijenja redosljede nizova tijekom procesa treniranja i zaključivanja. Posljedično, izračun perpleksnosti mora biti usklađen s navedenim pristupom.

Neka $z = (z_1, z_2, \dots, z_N)$ predstavlja tokenizirani numerički prozor duljine N . Perpleksnost za *XLNet* koji koristi permutacijsko jezično modeliranje izračunava se kao:

$$PPL(z) = \exp\left(\frac{1}{N} \sum_{i=1}^N -\log P(z_i | z_{<i}^{(\pi)})\right), \quad (4.10)$$

pri čemu $z_{<i}^{(\pi)}$ označava permutirani kontekst za i -ti token prema zadanim pravilima permutacijskog jezičnog modeliranja. Time se osigurava uvjetovanje različitih permutacija prethodnih tokena, umjesto da se model strogo oslanja na sekvencu slijeva nadesno.

Budući da *XLNet* ima zadanu maksimalnu duljinu ulaza (512 tokena za inačicu "*xlnet-base-cased*"), dugački nizovi moraju se također obrađivati u manjim dijelovima, kao i kod ostalih modela. Ako je z prevelik, dijeli se na manje segmente te se zatim iz-

računava gubitak negativne logaritamske vjerojatnosti svakog dijela zasebno. Nakon dobivanja perpleksnosti za svaki prozor iz skupa testiranje, postavlja se prag i označavaju se sekvence čija vrijednost prelazi tu granicu. Stršećim prozorima dodjeljuje se oznaka `pred_llm_xlnet=1` u podatkovnom okviru, dok neoznačenim prozorima i dalje ostaje vrijednost nula.

Permutacijsko jezično modeliranje *XLNet*-a omogućuje učenje dvosmjernog konteksta bez potrebe za maskiranjem tokena kao kod *BERT*-a, dok istovremeno zadržava autoregresijski pristup generiranju sadržaja, za razliku od modela *GPT-2*. Dinamičkim mijenjanjem redoslijeda tokena, *XLNet* prepoznaje dugotrajne ovisnosti među podacima na učinkovitiji način od modela s fiksnim redoslijedom. U procesu precizne prilagodbe, *XLNet* se učitava pomoću klase `AutoModelForPreTraining`, čuvajući svoju strategiju učenja temeljenu na permutaciji. Umjesto sekvencijske obrade znakovnih nizova, `DataCollatorForPermutationLanguageModeling` osigurava primjenu dinamičke permutacije, čime se potiče bolja generalizacija. Anomalije dobivene izračunom perpleksnosti preslikavaju se u podatkovni okvir vremenskih serija te se označavaju pomoću naredbe `pred_llm_xlnet=1`, čime se postiže mogućnost izravne usporedbe s modelima *GPT-2*, *DistilBERT* i *T5*. Navedena integracija olakšava analizu prednosti i nedostataka permutacijskog modeliranja u odnosu na kauzalni pristup slijeva nadesno ili maskirani pristup.

4.3. Integracija tradicionalnih metoda

Klasične metode otkrivanja anomalija predstavljaju optimalne referentne vrijednosti s kojima se uspoređuju pristupi temeljeni na velikim jezičnim modelima. U ovome radu korištene su četiri primarne metode iz biblioteke *scikit-learn*: izolacijska šuma, faktor lokalnog odstupanja, jednoklasni stroj potpornih vektora i eliptična omotnica. Navedeni pristupi odabrani su u svrhu isticanja njihove učinkovitosti prilikom detekcije stršećih vrijednosti u podacima vremenskih serija.

4.3.1. Izolacijska šuma

Izolacijska šuma je ansambl metoda koja izolira anomalije ponovljenim particioniranjem prostora značajki. Svako stablo u ansamblu nasumično dijeli podatkovne točke duž

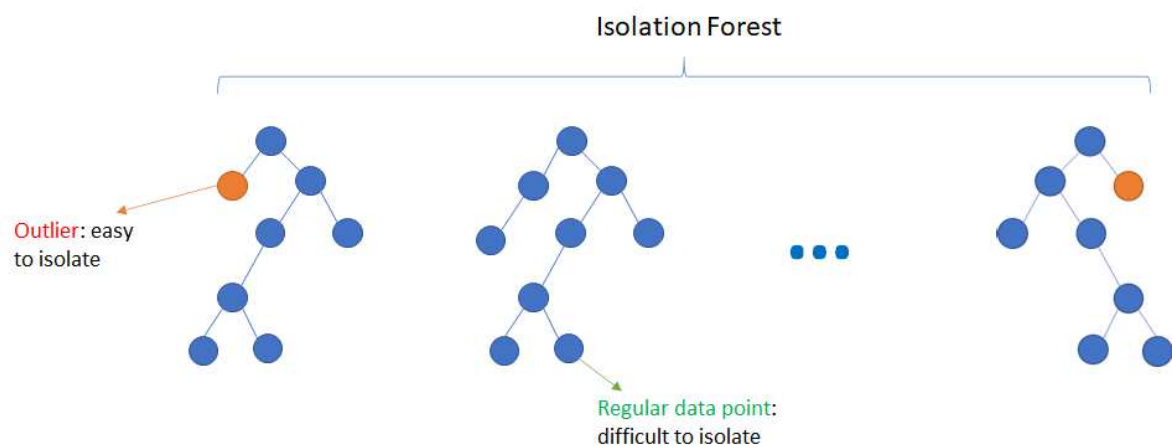
jedne značajke, s ciljem izdvajanja točaka koje zahtijevaju manji broj podjela. Algoritam očekuje da će se stršeće vrijednosti lakše izolirati.

```
def train_isolation_forest(X_train):  
    model = IsolationForest(  
        n_estimators=100,  
        contamination=0.01,  
        random_state=42  
    )  
    model.fit(X_train)  
    return model
```

Primjer 14: Isječak koda koji prikazuje provođenje algoritma izolacijska šuma

- `n_estimators=100`: Broj stabala u ansamblu.
- `contamination=0.01`: Gruba procjena da bi 1% podataka moglo biti odstupanje, što utječe na postavljanje praga.
- `random_state=42`: Osigurava nasumičnu inicijalizaciju.

Svako stablo u izolacijskoj šumi rekursivno dijeli podatke dok se ne izolira pojedinačna točka ili dok se ne ispuni kriterij zaustavljanja. Stršeće vrijednosti često završavaju u plićim granama, zbog čega dobivaju više ocjene anomalije.



Slika 4.11. Konceptualni dijagram koji prikazuje izvršavanje algoritma izolacijska šuma (Izvor: [12])

4.3.2. Faktor lokalnog odstupanja

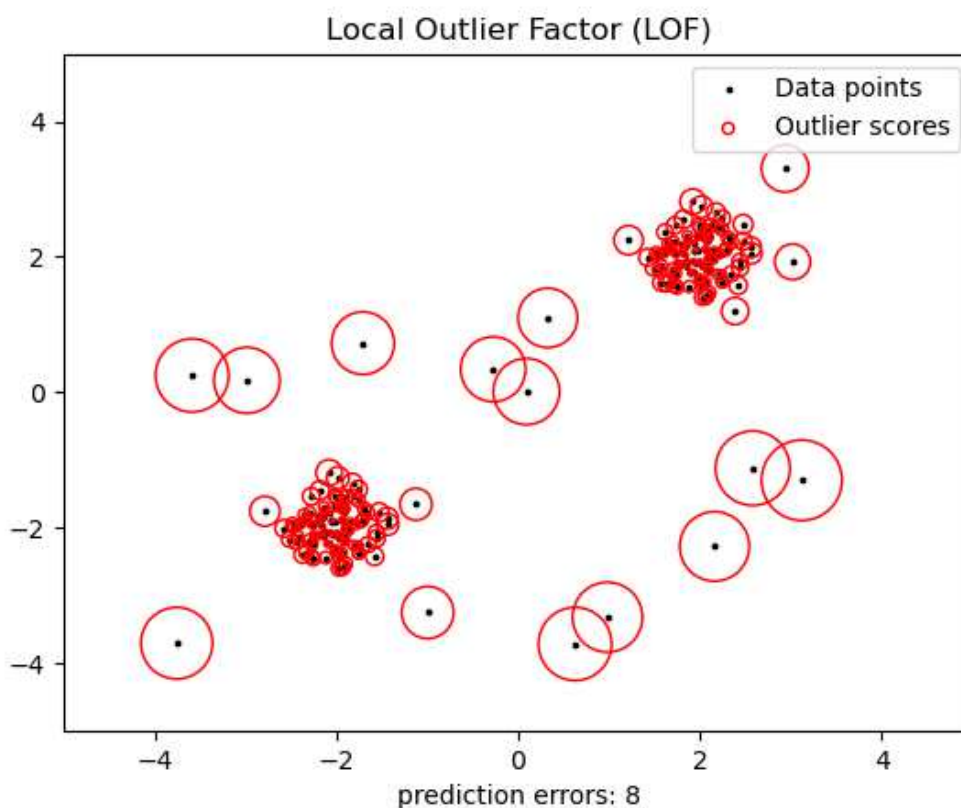
Faktor lokalnog odstupanja uspoređuje lokalnu gustoću podatkovne točke s gustoćom njezinih susjeda. Algoritam smatra da je pojedini uzorak anomalija ako je njegova lokalna gustoća znatno manja od gustoće njegovih susjeda.

```
def train_lof(X_train):  
    model = LocalOutlierFactor(  
        n_neighbors=20,  
        novelty=True,  
        contamination=0.01  
    )  
    model.fit(X_train)  
    return model
```

Primjer 15: Isječak koda koji prikazuje provođenje algoritma faktor lokalnog odstupanja

- `n_neighbors=20`: Broj susjeda koji se koristi za procjenu lokalne gustoće.
- `novelty=True`: Osigurava da se model može koristiti za otkrivanje odstupanja, dopuštajući predviđanja novih podataka nakon procesa treniranja.
- `contamination=0.01`: Slična uloga kao u izolacijskoj šumi, određuje udio stršćih vrijednosti.

Faktor lokalnog odstupanja dodjeljuje "vrijednost anomalije" na temelju usporedbe lokalne gustoće oko promatrane točke s gustoćom susjednih točaka. Manja lokalna gustoća sugerira potencijalno odstupanje.



Slika 4.12. Konceptualni dijagram koji prikazuje izvršavanje algoritma faktor lokalnog odstupanja (Izvor: [13])

4.3.3. Jednoklasni stroj potpornih vektora

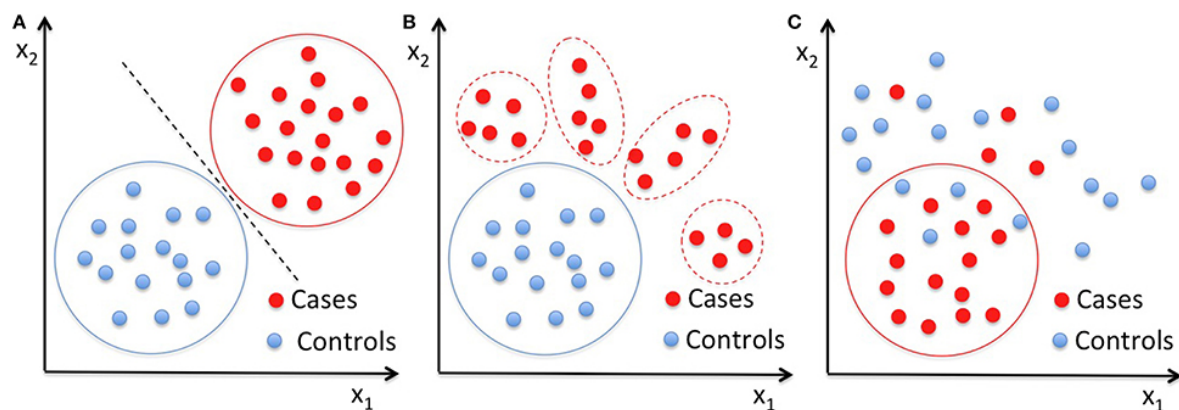
Jednoklasni stroj potpornih vektora modelira podatke iz skupa za treniranje unutar visokodimenzionalnog prostora značajki, pokušavajući naučiti granicu oko normalnih točaka. Točke koje se nalaze izvan navedene granice označavaju se kao odstupanja.

```
def train_ocsvm(X_train):
    model = OneClassSVM(
        kernel='rbf',
        gamma='scale',
        nu=0.01
    )
    model.fit(X_train)
    return model
```

Primjer 16: Isječak koda koji prikazuje provođenje algoritma jednoklasni stroj potpornih vektora

- `kernel='rbf'`: Jezgra radijalne bazne funkcije koja se najčešće koristi za nelinearne granice.
- `gamma='scale'`: Automatski skalira γ na temelju broja značajki.
- `nu=0.01`: Gornja granica količine anomalija u podacima, koja također utječe na marginu granice odlučivanja.

Stroj potpornih vektora pokušava obuhvatiti većinu točaka u području odlučivanja. Svaka točka koja se nalazi izvan te granice smatra se anomalijom. U kontekstu vremenskih serija, prostor značajki često odgovara višestrukim sensorima ili izvedenim metrikama.



Slika 4.13. Konceptualni dijagram koji prikazuje izvršavanje algoritma jednoklasni stroj potpornih vektora (Izvor: [14])

4.3.4. Eliptična omotnica

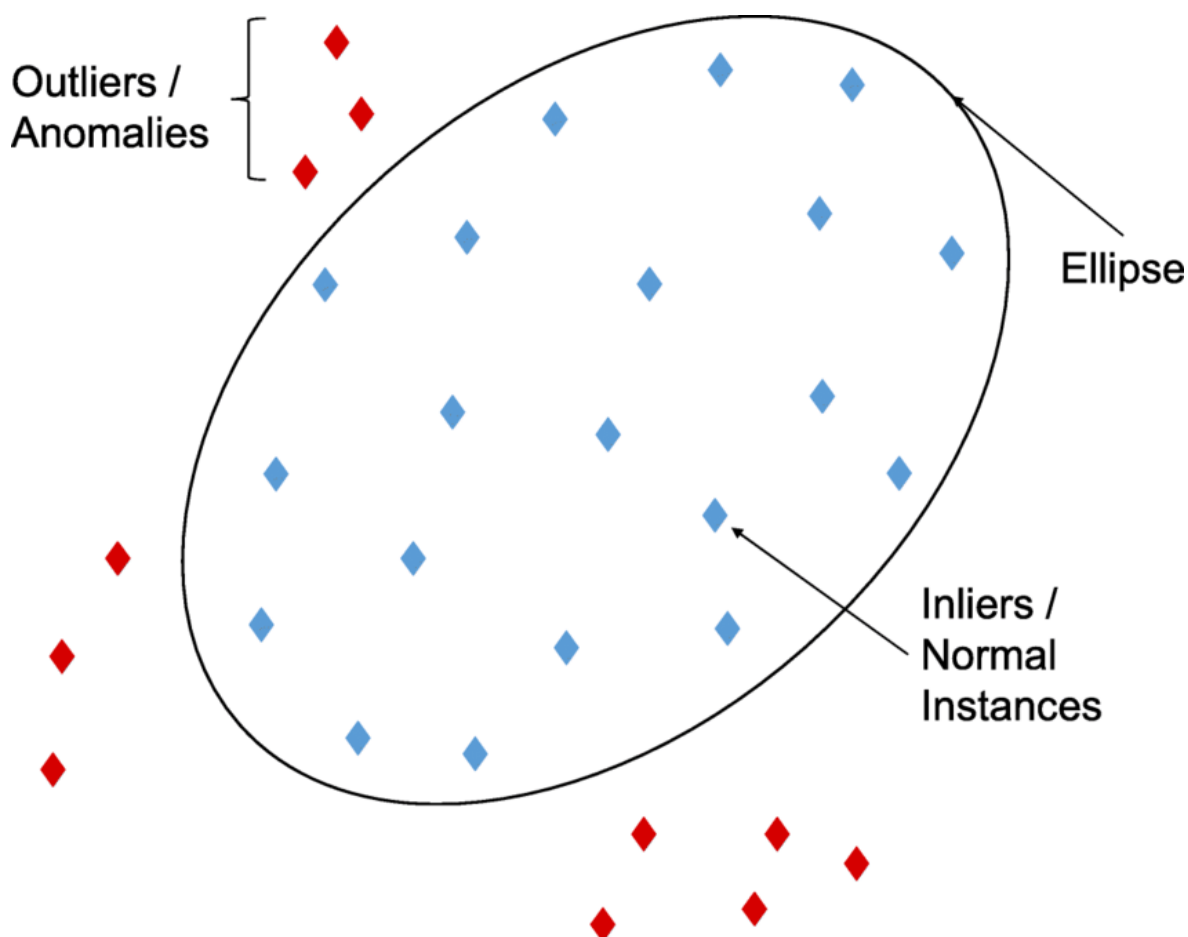
Eliptična omotnica pretpostavlja da podaci približno slijede Gaussovu distribuciju i prilagođava oblik elipse oko većine podataka u prostoru značajki. Podatkovne točke koje se nalaze izvan opisane elipse smatraju se stršećim vrijednostima.

```
def train_elliptic_envelope(X_train):
    model = EllipticEnvelope(
        contamination=0.01,
        support_fraction=0.7,
        random_state=42
    )
    model.fit(X_train)
    return model
```

Primjer 17: Isječak koda koji prikazuje provođenje algoritma eliptična omotnica

- `contamination=0.01`: Ponovno se pretpostavlja da je 1% podataka odstupanje.
- `support_fraction=0.7`: Podskup točaka korištenih za izračun robusne procjene kovarijance i položaja, pri čemu se zanemaruju ekstremne točke koje bi mogle iskriviti elipsu.

Eliptična omotnica izračunava vektor srednjih vrijednosti koji predstavlja lokaciju i matricu kovarijance na robustan način, zanemarujući izvanredne vrijednosti. Dobivena elipsa definira granicu normalnih podataka. Točke koje leže izvan elipse s velikim vrijednostima Mahalanobisove udaljenosti označavaju se kao anomalije.



Slika 4.14. Konceptualni dijagram koji prikazuje izvršavanje algoritma eliptična omotnica
(Izvor: [15])

4.4. Sažetak cjelokupne implementacije

U ovom poglavlju opisan je postupak implementacije metoda otkrivanja anomalija, uključujući različite jezične modele i tradicionalne pristupe strojnog učenja u svrhu procjene njihove učinkovitosti u procesu detekcije stršećih vrijednosti u podacima vremenskih serija.

Prvotno je opisana arhitektura sustava, uz detaljan prikaz protoka podataka, koraka predobrade i računalnog okvira koji se koristi za treniranje i evaluaciju. Poglavlje se zatim usredotočuje na četiri unaprijed uvježbana jezična modela: *GPT-2*, *DistilBERT*, *T5* i *XLNet*. Svaki od njih precizno je prilagođen za otkrivanje anomalija, korištenjem različitim strategija modeliranja temeljenih na tekstualnom sadržaju. *GPT-2* slijedi pristup kauzalnog jezičnog modeliranja, *DistilBERT* koristi maskirano jezično modeliranje s ciljem efikasnog kontekstualnog učenja, *T5* pretvara svaki zadatak u format *text-to-text*, a *XLNet* primjenjuje permutacijsko jezično modeliranje u svrhu prepoznavanja detaljnijih međuovisnosti u podacima.

Kako bi se uspostavile referentne vrijednosti, poglavlje izdvaja četiri klasične metode otkrivanja anomalija: izolacijska šuma, faktor lokalnog odstupanja, jednoklasni stroj potpornih vektora i eliptična omotnica. Izolacijska šuma identificira stršeće vrijednosti putem nasumične podjele podataka, faktor lokalnog odstupanja otkriva anomalije na temelju vrijednosti lokalnih gustoća, jednoklasni stroj potpornih vektora definira granicu oko normalnih podataka, a eliptična omotnica modelira Gaussovu distribuciju s ciljem označavanja anomalija.

U sljedećem poglavlju usporedit će se dobiveni rezultati velikih jezičnih modela i tradicionalnih metoda, ističući njihove prednosti i nedostatke. Na temelju ostvarenih ishoda procijenit će se mogu li veliki jezični modeli nadmašiti ili nadopuniti tradicionalne pristupe u procesu otkrivanja anomalija te će se predstaviti ideje za potencijalna poboljšanja implementiranih algoritama.

5. Eksperimentalna evaluacija

Ovaj odjeljak predstavlja kvantitativnu i kvalitativnu procjenu klasičnog pristupa otkrivanju anomalija i pristupa temeljenog na velikim jezičnim modelima. Analizirani skupovi podataka, pod vlasništvom američke organizacije *NASA*, nazivaju se *Soil Moisture Active Passive* (25 značajki) i *Mars Science Laboratory* (55 značajki).

Korištene su sljedeće tradicionalne metode detekcije anomalija:

- izolacijska šuma,
- faktor lokalnog odstupanja,
- jednoklasni stroj potpornih vektora i
- eliptična omotnica.

Također, implementirana su četiri velika jezična modela:

- *GPT-2*,
- *DistilBERT*,
- *T5* i
- *XLNet*.

Poglavlje 5.1. uvodi evaluacijske metrike korištene u analizi, objašnjavajući njihovu važnost u procjeni učinkovitosti otkrivanja anomalija. Mjerni podaci poput preciznosti (engl. *precision*), odziva (engl. *recall*), F1-rezultata (engl. *F1-score*) i točnosti (engl. *accuracy*) raspravljaju se u kontekstu njihove važnosti za klasične pristupe i velike jezične modele. U odjeljku 5.2. prikazani su empirijski rezultati kroz tablice i vizualizacije dobivenih vrijednosti. Nadalje, poglavlje 5.3. nudi usporednu analizu odabranih metoda, proučavajući razlike u točnosti, robusnosti i računalnoj učinkovitosti. Naposljetku, u odjeljku 5.4. tumače se dobiveni zaključci i predlažu načini nadogradnje postojećih algoritama, razmatrajući mogućnost praktičnog korištenja velikih jezičnih modela za detekciju stršećih vrijednosti u podacima vremenskih serija.

5.1. Metrike evaluacije

Točna procjena učinkovitosti otkrivanja anomalija zahtijeva kvantitativne metrike koje dokumentiraju učestalost ispravno identificiranih vrijednosti i broj pojavljivanja lažnih upozorenja. U navedenom kontekstu, svaka podatkovna točka označena je kao normalna ili odstupajuća. Nakon završenog procesa predviđanja, dobivene rezultate moguće je usporediti sa stvarnim oznakama pomoću matrice zabune (engl. *confusion matrix*).

	Predviđeno normalno	Predviđeno odstupajuće
Stvarno normalno	Stvarno negativno (TN)	Lažno pozitivno (FP)
Stvarno odstupajuće	Lažno negativno (FN)	Stvarno pozitivno (TP)

Tablica 5.1. Matrica zabune za detekciju anomalija

- **TP**: podatkovne točke označene kao odstupajuće, koje su zapravo isto odstupajuće.
- **TN**: podatkovne točke označene kao normalne, koje su zapravo isto normalne.
- **FP**: podatkovne točke označene kao odstupajuće, ali su zapravo normalne (lažna upozorenja).
- **FN**: podatkovne točke označene kao normalne, ali su zapravo odstupajuće (propuštene anomalije).

Iz ove četiri veličine izvode se točnost, preciznost, odziv i F1-rezultat, a svaki od njih opisan je u nastavku.

5.1.1. Točnost

Udio ispravno identificiranih uzoraka (i normalnih i odstupajućih) u ukupnom broju primjera:

$$\text{Točnost} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

Visoka točnost znači da model ispravno klasificira veliki dio uzoraka. Međutim, potencijalni problem kod ove metrike javlja se prilikom otkrivanja anomalija, budući da anomalije obično predstavljaju vrlo mali dio podataka. Posljedično, model koji pretežito

predviđa normalne vrijednosti može postići umjetno visoku točnost, iako ne uspijeva otkriti većinu pravih stršećih vrijednosti. Stoga, u slučaju visoko neuravnoteženih skupova podataka, u kojima prevladavaju normalni uzorci, sama točnost može biti pogrešna vrijednost za ispravnu analizu dobivenih rezultata.

5.1.2. Preciznost

Udio predviđenih anomalija koje su stvarno odstupajuće:

$$\text{Preciznost} = \frac{TP}{TP + FP}. \quad (5.2)$$

Visoka preciznost ukazuje na to da model generira nekoliko lažnih upozorenja, što znači da je samo mali broj normalnih točaka netočno označen kao anomalija. U praktičnim primjenama, osobito u slučajevima skupocjene provjere, preciznost postaje presudna u održavanju kontrole nad količinom lažno pozitivnih rezultata. Međutim, kod ove metrike javlja se suprotan problem nego kod točnosti. Model koji previđa vrlo malo anomalija može umjetno povećati svoju preciznost ako su anomalije koje detektira točne, dok i dalje ne uspijeva detektirati veliki broj stvarnih odstupanja. U takvom scenariju odziv poprima vrlo niske vrijednosti, budući da je obrnuto proporcionalan u odnosu na preciznost, što predstavlja najveći nedostatak ove metrike.

5.1.3. Odziv

Udio stvarnih anomalija koje model uspješno detektira:

$$\text{Odziv} = \frac{TP}{TP + FN}. \quad (5.3)$$

U situacijama u kojima je sigurnost ključna, poput pronalaženja grešaka u sustavu ili neovlaštenog pristupa, uočavanje maksimalnog broja anomalija presudno je za izbjegavanje lažno negativnih rezultata. Visoka vrijednost odziva upućuje na to da model može učinkovito prepoznati stvarne stršeće vrijednosti i smanjiti prisutnost propusta. Međutim, ako model pokušava otkriti sve anomalije, može proizvesti previše lažno pozitivnih rezultata, što dovodi do smanjenja preciznosti.

5.1.4. F1-rezultat

Harmonijska sredina preciznosti i odziva:

$$F1 = 2 \times \frac{\text{Preciznost} \times \text{Odziv}}{\text{Preciznost} + \text{Odziv}}. \quad (5.4)$$

F1-rezultat jedinstvena je metrika koja daje ravnotežu između preciznosti i odziva. Ako su preciznost ili odziv niski, F1-rezultat će također biti nizak (i obrnuto). Navedena vrijednost često se koristi u neuravnoteženim scenarijima poput otkrivanja anomalija, budući da sprječava pogrešno visoku ocjenu kada je vrijednost preciznosti ili odziva blizu nule.

5.2. Rezultati

Ovaj pododjeljak predstavlja numeričke i vizualne rezultate za otkrivanje anomalija pomoću klasičnih metoda i velikih jezičnih modela. Sve evaluacijske metrike izračunavaju se kako je opisano u prethodnom potpoglavlju.

5.2.1. Tradicionalne metode

Četiri klasične metode (eliptična omotnica, izolacijska šuma, faktor lokalnog odstupanja i jednoklasni stroj potpornih vektora) istrenirane su na normalnom dijelu podataka svakog kanala i testirane na određenom skupu za ispitivanje. Konačna predviđanja zatim su objedinjena i izvedena u obliku prosječne vrijednosti. Korištena su dva podskupa (*SMAP* i *MSL*), od kojih je svaki sadržavao 10 nasumično odabranih kanala.

1. *SMAP* podskup

- Kanali: *A-1, A-6, D-1, D-3, D-8, E-12, E-5, F-1, G-3, P-2*.
- Svaki kanal ima 25 stupaca značajki.

2. *MSL* podskup:

- Kanali: *C-1, D-14, D-16, F-5, F-8, M-2, M-4, M-6, P-10, P-14*.
- Svaki kanal ima 55 stupaca značajki.

Tablice 5.2. i 5.3. prikazuju prosječne vrijednosti evaluacijskih metrika za tradicionalne metode otkrivanja anomalija u odabranim podacima vremenskih serija, uz dosljednu

vrijednost kontaminacijskog parametra (0.01) i standardnim pragom (99. percentil) za metode temeljene na perpleksnosti.

Tablica 5.2. SMAP – Prosječne vrijednosti evaluacijskih metrika za klasične metode

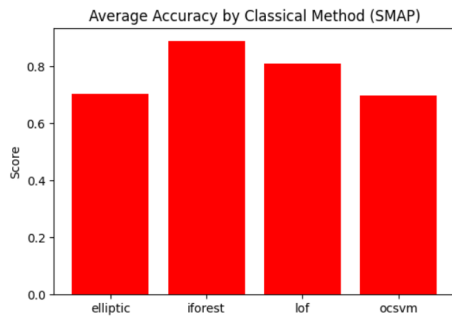
Metoda	Preciznost	Odziv	F1-rezultat	Točnost
eliptična omotnica	0.023476	0.116875	0.018298	0.702535
izolacijska šuma	0.095157	0.005562	0.008444	0.888641
faktor lokalnog odstupanja	0.042822	0.036425	0.021913	0.810648
jednoklasni stroj potpornih vektora	0.030894	0.152660	0.036262	0.696445

Eliptična omotnica pokazuje umjereni odziv (0.1169), ali vrlo nisku preciznost (0.0235). Izolacijska šuma postiže relativno visoku točnost (0.8886), ali izuzetno niski odziv. Faktor lokalnog odstupanja nudi skromnu ravnotežu, ali i dalje nije reprezentativan u kontekstu F1-rezultata (0.0219). Jednoklasni stroj potpornih vektora ističe se s vrijednosti odziva (0.1527), znatno višom od izolacijske šume ili faktora lokalnog odstupanja, iako žrtvuje preciznost (0.0309).

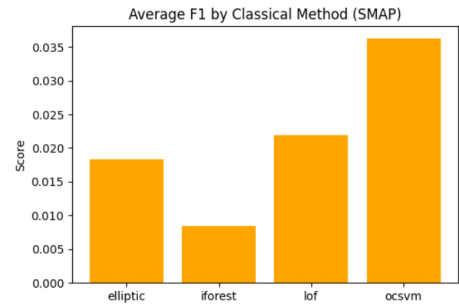
Tablica 5.3. MSL – Prosječne vrijednosti evaluacijskih metrika za klasične metode

Metoda	Preciznost	Odziv	F1-rezultat	Točnost
eliptična omotnica	0.019634	0.101657	0.008778	0.791337
izolacijska šuma	0.005602	0.011269	0.006370	0.955762
faktor lokalnog odstupanja	0.005661	0.005541	0.005218	0.900484
jednoklasni stroj potpornih vektora	0.009660	0.068756	0.015965	0.713414

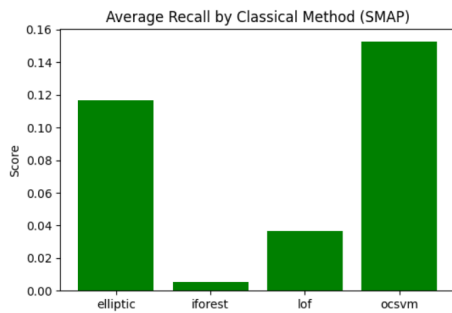
Eliptična omotnica opet ima malo bolju vrijednost odziva od izolacijske šume ili faktora lokalnog odstupanja, ali izuzetno nizak F1-rezultat (0.0088). Izolacijska šuma postiže vrlo visoku točnost (0.9558), a odziv je samo 0.0113, što ukazuje na značajno nedovoljnu detekciju anomalija. Vrijednosti faktora lokalnog odstupanja ostaju općenito vrlo niske. Jednoklasni stroj potpornih vektora postiže određeno poboljšanje u odzivu (0.0688) u usporedbi s izolacijskom šumom ili faktorom lokalnog odstupanja, nauštrb točnosti.



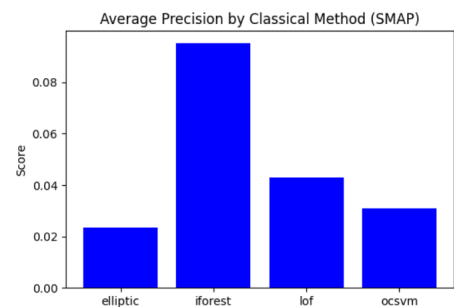
(a) Stupčasti dijagram točnosti klasičnih metoda na skupu *SMAP*



(b) Stupčasti dijagram F1-rezultata klasičnih metoda na skupu *SMAP*

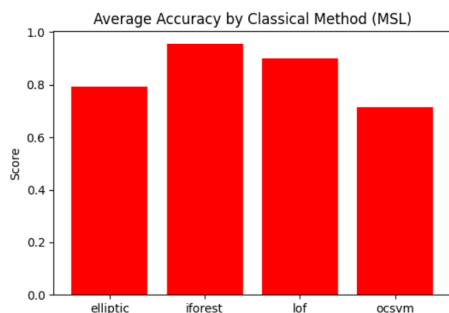


(c) Stupčasti dijagram odziva klasičnih metoda na skupu *SMAP*

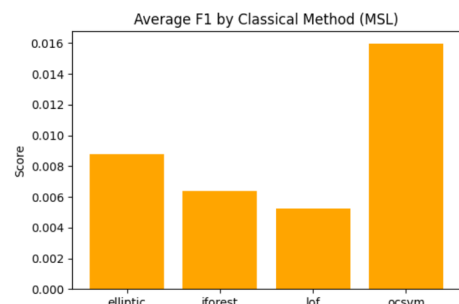


(d) Stupčasti dijagram preciznosti klasičnih metoda na skupu *SMAP*

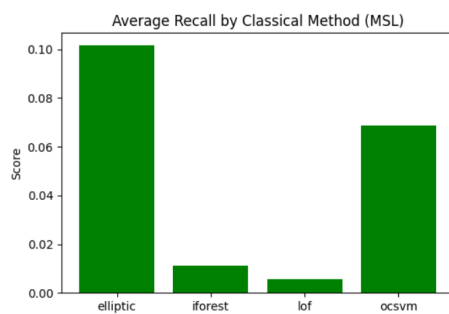
Slika 5.1. Evaluacijske metrike klasičnih metoda na skupu *SMAP*



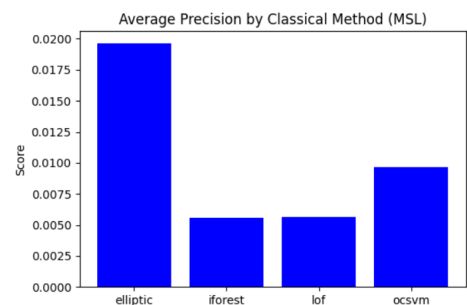
(a) Stupčasti dijagram točnosti klasičnih metoda na skupu *MSL*



(b) Stupčasti dijagram F1-rezultata klasičnih metoda na skupu *MSL*



(c) Stupčasti dijagram odziva klasičnih metoda na skupu *MSL*



(d) Stupčasti dijagram preciznosti klasičnih metoda na skupu *MSL*

Slika 5.2. Evaluacijske metrike klasičnih metoda na skupu *MSL*

Slike 5.1. i 5.2. služe za vizualnu usporedbu dobivenih rezultata. Vidljivo je da izolacijska šuma obično zauzima visoko mjesto u točnosti, zbog rijetkog označavanja anomalija. Jednoklasni stroj potpornih vektora može proizvesti najveće vrijednosti odziva u pojedinim podskupovima, ali ima problema s preciznošću. Ukupni F1-rezultati ostaju niski za sve metode, odražavajući poteškoće u istovremenom postizanju visoke preciznosti i visokog odziva prilikom korištenja klasičnih metoda detekcije anomalija na ovim konkretnim skupovima podataka.

Niske vrijednosti odziva u izolacijskoj šumi upućuju na rijetko označavanje anomalija, iz čega se može pretpostaviti da kritični događaji mogu proći neotkriveno. Jednoklasni stroj potpornih vektora postiže veću ravnotežu u vrijednostima odziva, ali zato može rezultirati lažnim upozorenjima zbog niske preciznosti. Iz tog razloga potrebno je pažljivo postaviti parametre ν i kernel za svaku pojedinu domenu. Pretpostavka o Gaussovoj distribuciji u eliptičnoj omotnici može biti restriktivna ako distribucija podataka izrazito ne podliježe navedenoj raspodjeli. Faktor lokalnog odstupanja prvenstveno se oslanja na lokalne pretpostavke o gustoći, koje mogu biti pretjerano pojednostavljene za multivarijatne vremenske serije sa složenim korelacijama između značajki.

Općenito, opisani pristupi pružaju referentne vrijednosti, ali svakako zahtijevaju specifičnu prilagodbu za pojedinu domenu ili dodatnu obradu značajki s ciljem učinkovitog prepoznavanja stršećih vrijednosti.

5.2.2. Metode temeljene na velikim jezičnim modelima

Veliki jezični modeli (*DistilBERT*, *GPT-2*, *T5* i *XLNet*) precizno su prilagođeni na prozorima s numeričkim tekstom i uspoređeni s postavljenom vrijednošću praga pomoću perpleksnosti. Tablice 5.4. i 5.5. prikazuju prosječnu izvedbu za odabrane skupove podataka.

1. **SMAP podskup**: 1024 prozora u skupu za treniranje, 3170 prozora u skupu za testiranje
2. **MSL podskup**: 1037 prozora u skupu za treniranje, 1268 prozora u skupu za testiranje

Tablica 5.4. SMAP – Prosječne vrijednosti evaluacijskih metrika za velike jezične modele

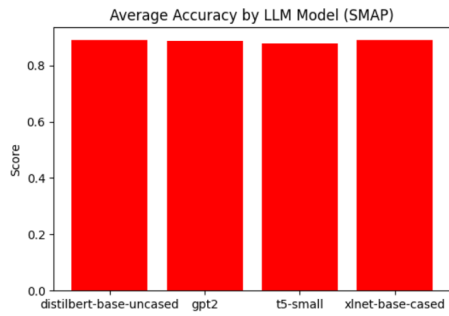
Model	Preciznost	Odziv	F1-rezultat	Točnost
DistilBERT	0.968750	0.081579	0.150485	0.889590
GPT-2	0.812500	0.068421	0.126214	0.886435
T5	0.437500	0.036842	0.067961	0.878864
XLNet	1.000000	0.084211	0.155340	0.890221

Model *XLNet* rezultira savršenom preciznošću 1.0, ali skromnim odzivom (0.0842). *DistilBERT* također postiže izrazito visoku preciznost (0.9688), s odzivom 0.0816. U modelu *GPT-2* i dalje ostaje slično visoka preciznost (0.8125) po cijenu slabog odziva (0.0684). *T5* postiže umjerenu točnost (0.8789), ali nižu preciznost (0.4375) i odziv (0.0368).

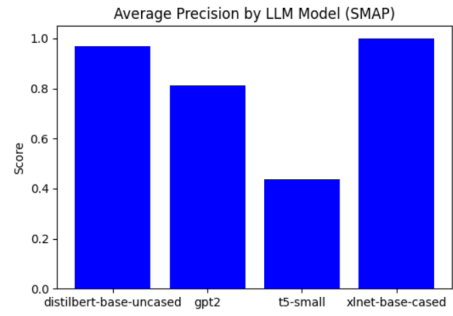
Tablica 5.5. MSL – Prosječne vrijednosti evaluacijskih metrika za velike jezične modele

Model	Preciznost	Odziv	F1-rezultat	Točnost
DistilBERT	0.769231	0.059524	0.110497	0.873028
GPT-2	0.461538	0.035714	0.066298	0.866719
T5	0.000000	0.000000	0.000000	0.857256
XLNet	0.615385	0.047619	0.088398	0.869874

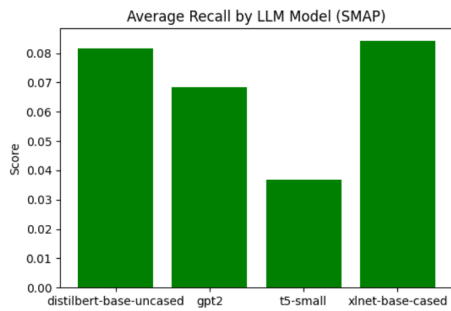
Na skupu *MSL*, *DistilBERT* predvodi s preciznošću 0.7692, ali odzivom 0.0595. Modeli *GPT-2* i *XLNet* održavaju relativno visoku točnost, ali i dalje im je vrijednost odziva manja od 0.05. *T5* rezultira nulom za vrijednost odziva, što implicira da nije označio niti jednu anomaliju te je iz tog razloga i F1 rezultat jednak nuli.



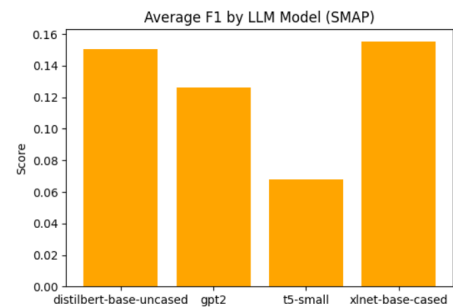
(a) Stupčasti dijagram točnosti velikih jezičnih modela na skupu *SMAP*



(b) Stupčasti dijagram preciznosti velikih jezičnih modela na skupu *SMAP*

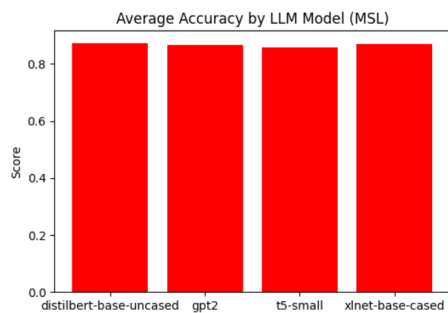


(c) Stupčasti dijagram odziva velikih jezičnih modela na skupu *SMAP*

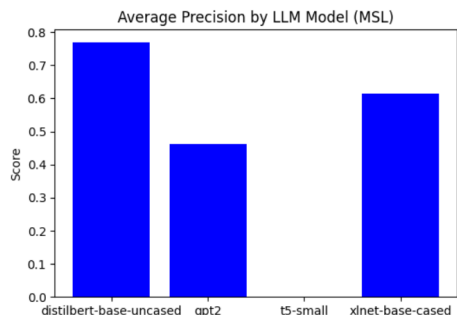


(d) Stupčasti dijagram F1-rezultata velikih jezičnih modela na skupu *SMAP*

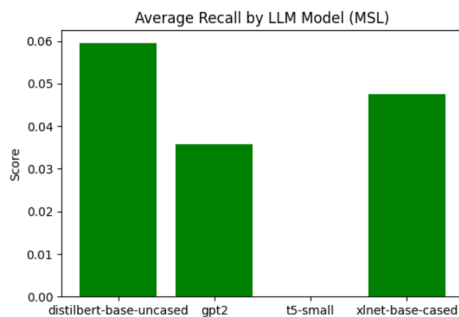
Slika 5.3. Evaluacijske metrike velikih jezičnih modela na skupu *SMAP*



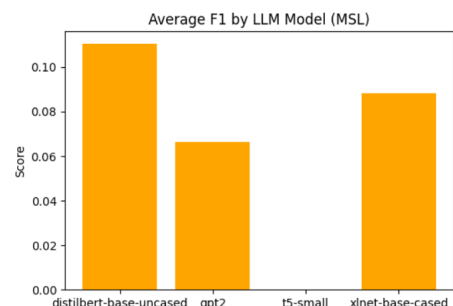
(a) Stupčasti dijagram točnosti velikih jezičnih modela na skupu *MSL*



(b) Stupčasti dijagram preciznosti velikih jezičnih modela na skupu *MSL*



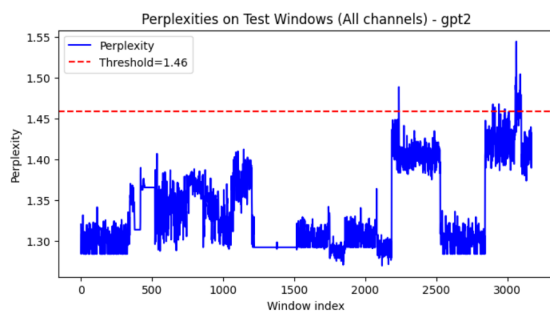
(c) Stupčasti dijagram odziva velikih jezičnih modela na skupu *MSL*



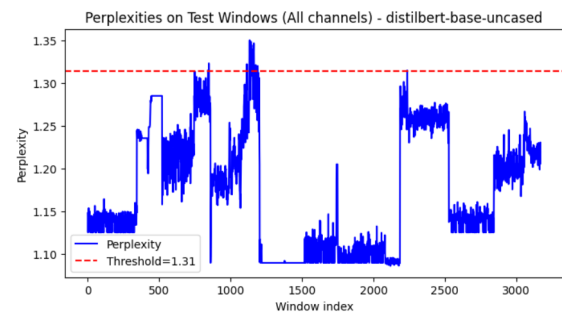
(d) Stupčasti dijagram F1-rezultata velikih jezičnih modela na skupu *MSL*

Slika 5.4. Evaluacijske metrike velikih jezičnih modela na skupu *MSL*

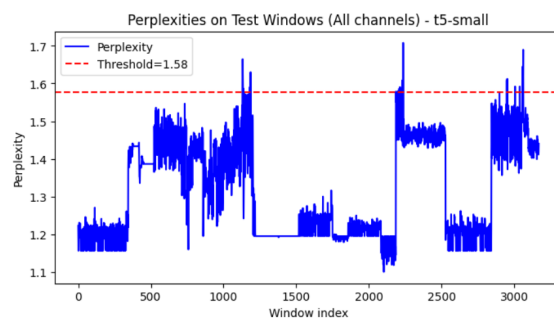
Na slikama 5.3. i 5.4. vidljivo je da metode temeljene na velikim jezičnim modelima često postižu iznimno visoku preciznost uz nekoliko lažno pozitivnih rezultata, suprotno od mnogih klasičnih metoda, koje mogu podlijegati većem broju lažnih upozorenja. Dok obje vrste pristupa imaju problema s otkrivanjem velike količine stvarnih anomalija, rezultirajući niskim vrijednostima odziva, razlozi njihovih pogrešaka znatno se razlikuju. Detaljnija analiza dobivenih rezultata bit će opisana u nastavku ovog poglavlja. Osim prikaza stupčastih dijagrama, na slikama 5.5. i 5.6. generirani su i grafovi perpleksnosti za svaki korišteni model.



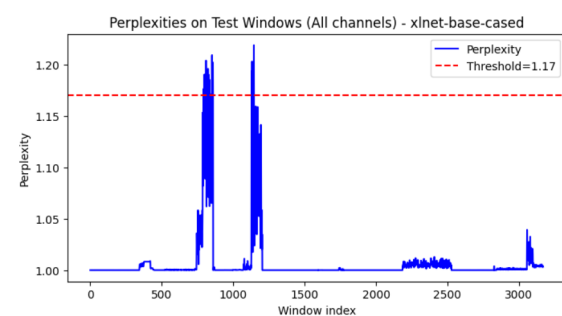
(a) Perpleksnost modela *GPT-2* na skupu *SMAP*



(b) Perpleksnost modela *DistilBERT* na skupu *SMAP*

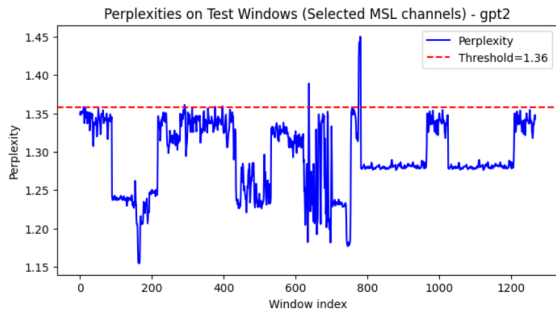


(c) Perpleksnost modela *T5* na skupu *SMAP*

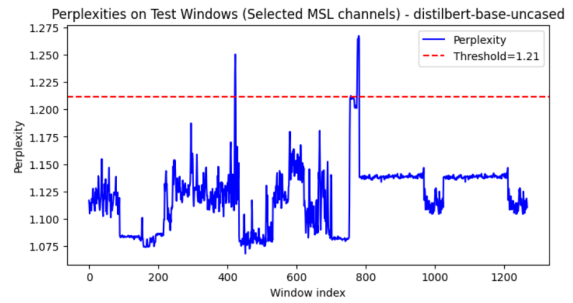


(d) Perpleksnost modela *XLNet* na skupu *SMAP*

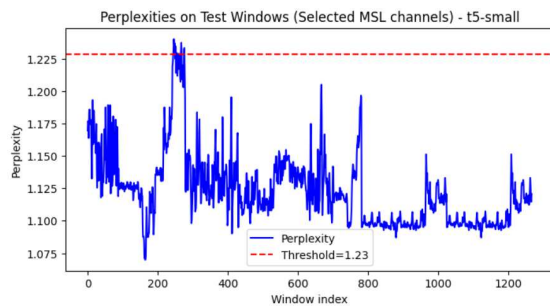
Slika 5.5. Vrijednosti perpleksnosti za sve prozore iz skupa za testiranje *SMAP*



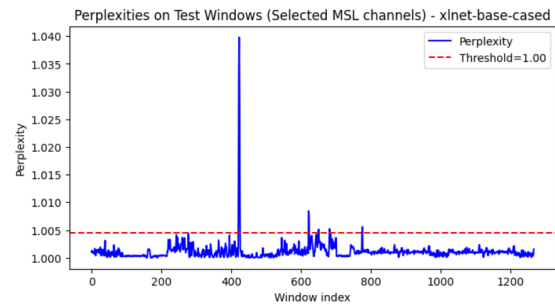
(a) Perpleksnost modela *GPT-2* na skupu *MSL*



(b) Perpleksnost modela *DistilBERT* na skupu *MSL*



(c) Perpleksnost modela *T5* na skupu *MSL*



(d) Perpleksnost modela *XLNet* na skupu *MSL*

Slika 5.6. Vrijednosti perpleksnosti za sve prozore iz skupa za testiranje *MSL*

Samo nekoliko pojedinih prozora prelazi postavljeni prag, što objašnjava visoke preciznosti, ali ograničenu pokrivenost stvarnih anomalija. Povremeni oštri šiljci na grafovima upućuju na prozore koji se osjetno razlikuju od naučenih normalnih obrazaca, što uzrokuje oznaku odstupanja. Konzervativni pristup velikih jezičnih modela razlog je stabilnih vrijednosti perpleksnosti, osim ako pripadni numerički podaci znatno odstupaju, u skladu s pragom od 99. percentila. Postavljeni prag daje manji broj anomalija, ali pouzdanije rezultate. Ovi dijagrami stoga vizualno potvrđuju kako funkcionira detekcija stršećih vrijednosti temeljena na velikim jezičnim modelima. Model održava relativno nisku perpleksnost za uobičajene prozore, a u slučaju neočekivanih podataka proizvodi visoke vrijednosti perpleksnosti te se pripadni podaci označavaju kao anomalije.

5.3. Analiza performansi

Ovaj pododjeljak govori o usporedbi velikih jezičnih modela i klasičnih metoda otkrivanja anomalija u kontekstu načina detekcije, količine računalnih sredstava i potrebe za preciznom prilagodbom koja je specifična za pojedinu domenu, u svrhu postizanja

boljih rezultata.

5.3.1. Usporedba velikih jezičnih modela i klasičnih metoda

Rezultati iz prethodnog potpoglavlja naglašavaju jasan kontrast između algoritama koji se temelje na velikim jezičnim modelima i tradicionalnih algoritama detekcije anomalija. Iako oba pristupa imaju poteškoća u slučaju iznimno rijetkih anomalija, pokazuju vrlo različite obrasce pogrešaka i karakteristike izvedbe.

Metode temeljene na velikim jezičnim modelima obično označavaju manji broj anomalija, čime se postiže veća preciznost i manje lažnih upozorenja. Često se oslanjaju na nagle promjene perpleksnosti kako bi prepoznali odstupanja, što ih čini iznimno konzervativnima. Takav pristup odgovara primjenama gdje su lažno pozitivni rezultati vrlo skupi (primjerice, dugotrajne ručne provjere). Međutim, ključne stršeće vrijednosti mogu ostati neotkrivene ako reprezentacija numeričkih podataka u tekstualnom obliku ne rezultira snažnim razlikama u perpleksnosti.

S druge strane, tradicionalne metode češće detektiraju stršeće vrijednosti, koristeći heuristiku temeljenu na kontaminaciji ili gustoći. Iz tog razloga, one mogu uočiti odstupanja koje veliki jezični modeli zanemaruju, ali uz rizik povećanja lažno pozitivnih rezultata i dobivanja niže preciznosti. Jednostavnije su za implementaciju, ali mogu se pokazati lošijima ako su anomalije vrlo složene i neupadljive ili ugrađene u visokodimenzionalni šum.

Oba pristupa imaju sveukupno nizak F1-rezultat zbog iznimno niske stope anomalija i neizbježnih kompromisa (preciznost u odnosu na pokrivenost). Veliki jezični modeli zadržavaju visoku preciznost, ali često propuštaju više anomalija, dok klasične metode variraju u pokrivenosti, ali mogu pogrešno označiti previše normalnih točaka.

Bitno je naglasiti da oba načina dodatno zahtijevaju vrlo složenu preciznu prilagodbu s ciljem dobivanja boljih rezultata. Veliki jezični modeli mogu imati koristi od specijalizirane numeričke tokenizacije, namještanja praga perpleksnosti ili povećanja količine podataka prilagođenih specifičnoj domeni. Klasični pristupi zahtijevaju podešavanje parametara poput kontaminacije, broja susjeda i izbora bazne funkcije, ovisno o distribuciji analiziranih podataka. Precizna prilagodba je iznimno kompleksna i ovisi o samoj

domeni te često uključuje iterativno istraživanje i spoznaju o tome kako se stršeće vrijednosti manifestiraju u pojedinom skupu podataka.

5.3.2. Računalni zahtjevi

Vrijeme izvođenja klasičnih metoda iznosi 10 do 15 sekundi po pojedinom skupu podataka. Također, one mogu učinkovito raditi na standardnom *CPU* procesoru, bez potrebe za posebnim hardverom. Moguće je zaključiti da su tradicionalne metode vrlo prigodne za implementaciju velikih skupova podataka u stvarnom vremenu, pritom zahtijevajući malo računalnih sredstava.

Trajanje precizne prilagodbe velikih jezičnih modela na numeričkim prozorima iznosi 25 do 30 minuta po analiziranom skupu podataka, čak i uz korištenje *GPU* procesora. Zaključivanje velikih jezičnih modela također je sporije od klasičnih metoda, zbog zasebnog izračuna perpleksnosti za svaki dio podijeljenog niza. Dakle, veliki jezični modeli zahtijevaju snažniji hardver i izvanmrežni način rada, kada je visoka cijena treniranja prihvatljiva. Također, potrebno je detaljno prilagoditi hiperparametre modela (stopa učenja, duljina niza, strategija tokenizacije) kako bi se optimizirala brzina i izvedba implementiranih algoritama.

5.4. Ključne spoznaje

U ovom se pododjeljku razmatraju glavna otkrića, opisuju prednosti i nedostaci svakog pristupa te zaključuje o mogućnostima praktične primjene i smjernicama budućih istraživanja. Naposljetku, i metode temeljene na velikim jezičnim modelima i klasične metode koriste određene kompromise prilikom zaključivanja, budući da niti jedan pristup nema sposobnost jednostavnog prepoznavanja anomalija (*plug-and-play* princip), zbog izuzetno male učestalosti prisutnih odstupanja u odabranom skupu podataka. Oba načina zahtijevaju pažljivo postavljanje praga i dubinsko poznavanje domene kako bi se postigla ravnoteža između točnosti otkrivanja, lažnih upozorenja i računalnih resursa.

Tradicionalni algoritmi za otkrivanje anomalija oslanjaju se na izravnu numeričku heuristiku ili heuristiku temeljenu na gustoći podataka. S druge strane, metode temeljene na velikim jezičnim modelima koriste nagle promjene perpleksnosti izvedene iz

tekstualnog prikaza numeričkih podataka. Veliki jezični modeli ističu se u preciznosti zbog označavanja manjeg broja anomalija i smanjivanjem količine lažnih upozorenja, dok klasične metode mogu otkriti različite obrasce nauštrb većeg broja lažno pozitivnih primjera, pogotovo ako pragovi nisu pažljivo postavljeni. U slučaju iznimno rijetkih odstupanja, oba pristupa nailaze na prepreke. Male promjene u vrijednostima praga mogu značajno promijeniti ravnotežu između propuštenih anomalija i lažnih upozorenja. Iz tog razloga, jako je bitno precizno prilagoditi sve parametre.

Prednosti velikih jezičnih modela:

- **Visoka preciznost:** minimiziraju lažno pozitivne rezultate označavanjem anomalija samo kada perpleksnost prijeđe postavljeni prag.
- **Uočavanje složenih obrazaca:** transformatorske arhitekture mogu prepoznati zamršene vremenske odnose ili interakcije značajki, pod uvjetom da kodiranje numeričkih podataka u tekstualni oblik naglašava te obrasce na odgovarajući način.

Nedostaci velikih jezičnih modela:

- **Računalni zahtjevi:** precizna prilagodba može trajati 25 do 30 minuta po skupu podataka, a zaključak o perpleksnosti također zahtijeva veliku količinu potrebnih resursa.
- **Problemi s numeričkom tokenizacijom:** brojčane vrijednosti možda neće uvijek proizvesti različite tokene, što otežava otkrivanje graničnih anomalija u slučaju kada tokenizirani podaci ne odražavaju male numeričke razlike.

Općenito, obje vrste pristupa imaju jasne prednosti i ograničenja zbog niske stope anomalija. Iako veliki jezični modeli mogu ponuditi veću preciznost, njihovi računalni zahtjevi i potreba za pažljivom preciznom prilagodbom ograničava mogućnost *plug-and-play* korištenja, koje je često najpoželjnije. Klasični pristupi jednostavniji su za implementaciju, ali mogu imati više lažno pozitivnih primjera. Odabir optimalnog pristupa uključuje kompromise između dostupnosti resursa, prihvatljivog rizika od lažnih upozorenja i potrebnog stupnja precizne prilagodbe za specifičnu domenu.

6. Zaključak

Ovaj diplomski rad istražuje učinkovitost korištenja velikih jezičnih modela u otkrivanju anomalija, uspoređujući dobivene vrijednosti s tradicionalnim pristupima. Rezultati potvrđuju da su obje paradigme uspješne pod određenim uvjetima, ali obavezno zahtijevaju detaljnu preciznu prilagodbu, osobito u slučajevima s izuzetno niskom stopom prisutnih anomalija. Dokazano je da se pretvorbom numeričkih podataka u tekstualni oblik može omogućiti velikim jezičnim modelima prepoznavanje složenih obrazaca, koji nisu vidljivi klasičnim metodama, ali uz cijenu velikih računalnih zahtjeva. Nadalje, istaknuta je važnost ispravnog postavljanja praga i dovoljne informiranosti o analiziranoj domeni, u svrhu ublažavanja lažnih upozorenja. Naposljetku, optimalno rješenje uključivalo bi kombinaciju oba pristupa. Veliki jezični modeli mogu dati visoku preciznost u slučajevima obilnih podataka i dostupnih računalnih sredstava, dok su klasične metode prikladnije u scenarijima nedovoljne količine raspoloživih resursa, kada je potrebna analiza podataka u stvarnom vremenu i stvaranje lažno pozitivnih rezultata ne predstavlja problem.

Moguća poboljšanja dosadašnje implementacije uključuju numeričke strategije tokenizacije, izdvajanje značajki specifičnih za pojedinu domenu, korištenje lanca misli (engl. *chain-of-thought*), preciznu prilagodbu uputa (engl. *instruction fine-tuning*), napredne preinake u određivanju praga i otkrivanje anomalija u stvarnom vremenu tijekom prijenosa podataka. Jedan od takvih primjera bio bi korištenje hibridnih modela, kombiniranjem ugrađivanja izvedenim iz velikih jezičnih modela s klasičnim metodama poput faktora lokalnog odstupanja ili jednoklasnog stroja potpornih vektora u ugradbenom prostoru, s ciljem prepoznavanja složenih obrazaca bez značajnog povećanja u vremenu zaključivanja. Drugi primjer mogao bi biti usmjeren na numeričku tokenizaciju, u kojoj su specifični tokenizatori za pojedinu domenu prilagođeni brojčanim podacima te se na taj način pokušavaju ublažiti problemi graničnih anomalija koje se ne presli-

kavaju u različite znakovne nizove. Aktivno učenje ili polu-nadzirane metode također imaju mogućnost dodatnog poboljšanja izvedbe modela, iterativnim ažuriranjem praga ili kontaminacijskog parametra, paralelno s dolaskom novih podataka. Na taj način mogla bi se postići prilagodba sustava u stvarnom vremenu. Dodatno, takav učinak ostvariv je i pomoću *online* prilagodbe sustava kroz prijenos različitih načina računanja perpleksnosti ili dinamičkog ažuriranja praga. Opisanim procesom omogućio bi se kontinuirani nadzor aplikacija u stvarnom vremenu, osiguravajući točnost i brzinu otkrivanja anomalija.

U konačnici, niti jedno rješenje ne predvodi u svakom scenariju, posebice u slučaju iznimno niske stope stršćih vrijednosti. Poznavanje domene ključno je za ispravno provođenje predobrade podataka, optimizaciju parametara i postavljanje praga. Sustavnim istraživanjem klasičnih pristupa i metoda temeljenih na velikim jezičnim modelima te potencijalnim kombiniranjem njihovih obilježja u hibridnim ili polu-nadziranim arhitekturama, podatkovni znanstvenici mogli bi se približiti stvaranju robusnog, učinkovitog i adaptivnog procesnog toka za otkrivanje anomalija u vremenskim serijama.

Literatura

- [1] D. Effrosynidis, *Time Series Analysis with Theory, Plots, and Code Part 1*, pristupljeno 8. veljače 2025. [Mrežno]. Adresa: <https://medium.com/towards-data-science/time-series-analysis-with-theory-plots-and-code-part-1-dd3ea417d8c4>
- [2] G. Cunningham, R. C. Bunescu, i D. Juedes, *Towards Autoformalization of Mathematics and Code Correctness: Experiments with Elementary Proofs*, pristupljeno 16. prosinca 2024. [Mrežno]. Adresa: <https://arxiv.org/pdf/2301.02195>
- [3] M. Jin, H. Tang, C. Zhang, Q. Yu, C. Liu, S. Zhu, Y. Zhang, i M. Du, *Time Series Forecasting with LLMs: Understanding and Enhancing Model Capabilities*, pristupljeno 3. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2402.10835v1>
- [4] X. Zhang, R. R. Chowdhury, R. K. Gupta, i J. Shang, *Large Language Models for Time Series: A Survey*, pristupljeno 20. prosinca 2024. [Mrežno]. Adresa: <https://www.ijcai.org/proceedings/2024/0921.pdf>
- [5] J. Morales-Garcia, A. Llanes, F. Arcas-Tunez, i F. Terroso-Saenz, *Developing Time Series Forecasting Models with Generative Large Language Models*, pristupljeno 21. prosinca 2024. [Mrežno]. Adresa: <https://dl.acm.org/doi/pdf/10.1145/3663485>
- [6] S. D. Yang, Z. A. Ali, i B. M. Wong, *FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion*, pristupljeno 19. siječnja 2025. [Mrežno]. Adresa: https://www.researchgate.net/publication/373352176_FLUID-GPT_Fast_Learning_to_Understand_and_Investigate_Dynamics_with_a_Generative_Pre-Trained_Transformer_Efficient_Predictions_of_Particle_Trajectories_and_Erosion
- [7] V. Sanh, L. Debut, J. Chaumond, i T. Wolf, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, pristupljeno 28. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/1910.01108>
- [8] H. Adel, A. Dahou, A. Mabrouk, M. A. Elaziz, M. Kayed, I. M. El-Henawy,

- S. Alshathri, i A. A. Ali, *Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm*, pristupljeno 9. veljače 2025. [Mrežno]. Adresa: https://www.researchgate.net/publication/358239462_Improving_Crisis_Events_Detection_Using_DistilBERT_with_Hunger_Games_Search_Algorithm
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, i I. Polosukhin, *Attention Is All You Need*, pristupljeno 4. prosinca 2024. [Mrežno]. Adresa: <https://arxiv.org/pdf/1706.03762>
- [10] A. Zhang, Z. Lipton, M. Li, i A. J. Smola, *Large-Scale Pretraining with Transformers*, pristupljeno 9. veljače 2025. [Mrežno]. Adresa: https://d2l.ai/chapter_attention-mechanisms-and-transformers/large-pretraining-transformers.html
- [11] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, i Q. V. Le, *XLNet: Generalized Autoregressive Pretraining for Language Understanding*, pristupljeno 21. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/1906.08237>
- [12] P. Pandey, *Outlier Detection using Isolation Forests*, pristupljeno 9. veljače 2025. [Mrežno]. Adresa: <https://machinelearninggeek.com/outlier-detection-using-isolation-forests/>
- [13] Scikit-learn, *Outlier detection with Local Outlier Factor (LOF)*, pristupljeno 9. veljače 2025. [Mrežno]. Adresa: https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html
- [14] A. Retico, I. Gori, A. Giuliano, F. Muratori, i S. Calderoni, *One-Class Support Vector Machines Identify the Language and Default Mode Regions As Common Patterns of Structural Alterations in Young Children with Autism Spectrum Disorders*, pristupljeno 9. veljače 2025. [Mrežno]. Adresa: <https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2016.00306/full>
- [15] Y. Regaya, F. Fadli, i A. Amira, *Point-Denoise: Unsupervised outlier detection for 3D point clouds enhancement*, pristupljeno 9. veljače 2025. [Mrežno]. Adresa: https://www.researchgate.net/publication/352017898_Point-Denoise_Unsupervised_outlier_detection_for_3D_point_clouds_enhancement
- [16] Kaggle, *Time Series Anomaly Detection*, pristupljeno 5. listopada 2024. [Mrežno]. Adresa: <https://www.kaggle.com/code/joshuaswords/time-series-anomaly-detection/notebook>
- [17] S. Alnegheimish, L. Nguyen, L. Berti-Equille, i K. Veeramachaneni, *Large language*

- models can be zero-shot anomaly detectors for time series?*, pristupljeno 5. listopada 2024. [Mrežno]. Adresa: <https://arxiv.org/pdf/2405.14755>
- [18] GitHub, *Using Large Language Models for Time Series Anomaly Detection*, pristupljeno 11. listopada 2024. [Mrežno]. Adresa: <https://github.com/sintel-dev/sigllm>
- [19] A. Lavin i S. Ahmad, *Evaluating Real-time Anomaly Detection Algorithms - the Numenta Anomaly Benchmark*, pristupljeno 11. listopada 2024. [Mrežno]. Adresa: <https://arxiv.org/pdf/1510.03336v4>
- [20] GitHub, *Public datasets for time series anomaly detection*, pristupljeno 13. listopada 2024. [Mrežno]. Adresa: <https://github.com/elisejiuqizhang/TS-AD-Datasets>
- [21] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, i H. Qiao, *Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications*, pristupljeno 21. listopada 2024. [Mrežno]. Adresa: <https://arxiv.org/pdf/1802.03903>
- [22] Kaggle, *Benchmark: Labeled Anomaly Detection TS*, pristupljeno 4. studenog 2024. [Mrežno]. Adresa: <https://www.kaggle.com/datasets/caesarlupum/benchmark-labeled-anomaly-detection-ts/data>
- [23] —, *Seasonal Anomaly Detection - Streaming Data*, pristupljeno 4. studenog 2024. [Mrežno]. Adresa: <https://www.kaggle.com/code/leomauro/seasonal-anomaly-detection-streaming-data>
- [24] R. J. Hyndman i G. Athanasopoulos, *Forecasting: principles and practice, 2nd edition*, pristupljeno 20. studenog 2024. [Mrežno]. Adresa: <https://otexts.com/fpp2/>
- [25] S. Modeling i Forecasting, *How To Isolate Trend, Seasonality And Noise From A Time Series*, pristupljeno 21. studenog 2024. [Mrežno]. Adresa: <https://timeseriesreasoning.com/contents/time-series-decomposition/>
- [26] AWS, *What is Anomaly Detection?*, pristupljeno 23. studenog 2024. [Mrežno]. Adresa: <https://aws.amazon.com/what-is/anomaly-detection/>
- [27] Tableau, *Time Series Analysis: Definition, Types, Techniques, and When It's Used*, pristupljeno 23. studenog 2024. [Mrežno]. Adresa: <https://www.tableau.com/analytics/what-is-time-series-analysis>
- [28] Anomalo, *Data Anomaly: What Is It, Common Types and How to Identify Them*, pristupljeno 23. studenog 2024. [Mrežno]. Adresa: <https://www.anomalo.com/>

- blog/data-anomaly-what-is-it-common-types-and-how-to-identify-them/
- [29] R. Shokrzad, *6 Pivotal Anomaly Detection Methods: From Foundations to 2023's Best Practices*, pristupljeno 1. prosinca 2024. [Mrežno]. Adresa: <https://medium.com/@reza.shokrzad/6-pivotal-anomaly-detection-methods-from-foundations-to-2023s-best-practices-5f037b530ae6>
- [30] M. Munir, M. A. Chattha, A. Dengel, i S. Ahmed, *A Comparative Analysis of Traditional and Deep Learning-based Anomaly Detection Methods for Streaming Data*, pristupljeno 2. prosinca 2024. [Mrežno]. Adresa: https://www-live.dfki.de/fileadmin/user_upload/import/10754_comparative_anomaly_detection_ICMLA.pdf
- [31] DataHeroes, *The Top Anomaly Detection Techniques You Need to Know*, pristupljeno 4. prosinca 2024. [Mrežno]. Adresa: <https://dataheroes.ai/blog/anomaly-detection-techniques-you-need-to-know/>
- [32] V. Shutenko, *AI Anomaly Detection: Best Tools And Use Cases*, pristupljeno 4. prosinca 2024. [Mrežno]. Adresa: <https://www.techmagic.co/blog/ai-anomaly-detection>
- [33] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, i Z. Sui, *A Survey on In-context Learning*, pristupljeno 15. prosinca 2024. [Mrežno]. Adresa: <https://arxiv.org/pdf/2301.00234>
- [34] P. Team, *How LLMs and Data Analytics Work Together*, pristupljeno 17. prosinca 2024. [Mrežno]. Adresa: <https://www.pecan.ai/blog/llm-data-analytics-work-together/>
- [35] M. Noguer i Alonso i R. Pereira Franklin, *Large Language Models for Financial Time Series Forecasting*, pristupljeno 20. prosinca 2024. [Mrežno]. Adresa: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4988022
- [36] Y. Jiang, Z. Pan, X. Zhang, S. Garg, A. Schneider, Y. Nevmyvaka, i D. Song, *Empowering Time Series Analysis with Large Language Models: A Survey*, pristupljeno 3. siječnja 2025. [Mrežno]. Adresa: <https://www.ijcai.org/proceedings/2024/0895.pdf>
- [37] S. Alnegheimish, L. Nguyen, L. Berti-Equille, i K. Veeramachaneni, *Can Large Language Models be Anomaly Detectors for Time Series?*, pristupljeno 3. siječnja 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/10722786>

- [38] W. Dang, B. Zhou, W. Zhang, i S. Hu, *Time Series Anomaly Detection Based on Language Model*, pristupljeno 6. siječnja 2025. [Mrežno]. Adresa: <https://dl.acm.org/doi/10.1145/3396851.3402925>
- [39] C. Liu, S. He, Q. Zhou, S. Li, i W. Meng, *Large Language Model Guided Knowledge Distillation for Time Series Anomaly Detection*, pristupljeno 7. siječnja 2025. [Mrežno]. Adresa: <https://www.ijcai.org/proceedings/2024/0239.pdf>
- [40] M. Horak, S. Chandrasekaran, i G. Tobar, *NLP Based Anomaly Detection for Categorical Time Series*, pristupljeno 7. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2204.10483>
- [41] D. Wei, W. Sun, X. Zou, D. Ma, H. Xu, P. Chen, C. Yang, M. Chen, i H. Li, *An anomaly detection model for multivariate time series with anomaly perception*, pristupljeno 10. siječnja 2025. [Mrežno]. Adresa: <https://peerj.com/articles/cs-2172/#>
- [42] A. Kalinowski i Y. An, *A Survey of Embedding Space Alignment Methods for Language and Knowledge Graphs*, pristupljeno 11. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2010.13688>
- [43] Kaggle, *NASA Anomaly Detection Dataset SMAP and MSL*, pristupljeno 11. siječnja 2025. [Mrežno]. Adresa: <https://www.kaggle.com/datasets/patrickfleith/nasa-anomaly-detection-dataset-smap-msl>
- [44] R. Lowe, *OpenAI's GPT-2: the model, the hype, and the controversy*, pristupljeno 14. siječnja 2025. [Mrežno]. Adresa: <https://medium.com/towards-data-science/openais-gpt-2-the-model-the-hype-and-the-controversy-1109f4bfd5e8>
- [45] G. AI, *T5 (language model)*, pristupljeno 21. siječnja 2025. [Mrežno]. Adresa: [https://en.wikipedia.org/wiki/T5_\(language_model\)](https://en.wikipedia.org/wiki/T5_(language_model))
- [46] L. Kozma i J. Voderholzer, *Theoretical Analysis of Byte-Pair Encoding*, pristupljeno 21. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2411.08671>
- [47] V. Rebernak, *Otkrivanje anomalija i stršćih vrijednosti u podacima*, pristupljeno 24. siječnja 2025. [Mrežno]. Adresa: <https://repozitorij.fer.unizg.hr/islandora/object/fer%3A10562>
- [48] S. Nobrega, *Next-Level Anomaly Detection: Unlocking Neural Insights*, pristupljeno 25. siječnja 2025. [Mrežno]. Adresa: <https://medium.com/towards-data-science/an-exploration-of-model-state-data-in-anomaly-detection-e6860cbca160>
- [49] U. I. M. L. Repository, *Steel Industry Energy Consumption*, pristupljeno 26. siječnja

2025. [Mrežno]. Adresa: <https://archive.ics.uci.edu/dataset/851/steel+industry+energy+consumption>
- [50] Investing.com, *Nasdaq 100 Historical Data*, pristupljeno 26. siječnja 2025. [Mrežno]. Adresa: <https://www.investing.com/indices/nq-100-historical-data>
- [51] Boltzmannbrain, *Numenta Anomaly Benchmark (NAB)*, pristupljeno 26. siječnja 2025. [Mrežno]. Adresa: <https://www.kaggle.com/datasets/boltzmannbrain/nab/data>
- [52] H. Face, *The AI community building the future*, pristupljeno 27. siječnja 2025. [Mrežno]. Adresa: <https://huggingface.co/>
- [53] V. Ekambaram, A. Jati, P. Dayama, S. Mukherjee, N. H. Nguyen, W. M. Gifford, C. Reddy, i J. Kalagnanam, *Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series*, pristupljeno 27. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2401.03955>
- [54] Y. Li, A. Korhonen, i I. Vulić, *On Bilingual Lexicon Induction with Large Language Models*, pristupljeno 29. siječnja 2025. [Mrežno]. Adresa: <https://aclanthology.org/2023.emnlp-main.595.pdf>
- [55] E. Schwitzgebel, D. Schwitzgebel, i A. Strasser, *Creating a large language model of a philosopher*, pristupljeno 29. siječnja 2025. [Mrežno]. Adresa: <https://onlinelibrary.wiley.com/doi/epdf/10.1111/mila.12466>
- [56] J. J. Yao, M. Aggarwal, R. D. Lopez, i N. Surena, *Large Language Models in Orthopaedics*, pristupljeno 29. siječnja 2025. [Mrežno]. Adresa: https://journals.lww.com/jbjsjournal/abstract/2024/08070/large_language_models_in_orthopaedics_.7.aspx
- [57] Y. Wu, A. Q. Jiang, W. Li, M. N. Rabe, C. Staats, M. Jamnik, i C. Szegedy, *Autoformalization with Large Language Models*, pristupljeno 29. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2205.12615>
- [58] C. Veres, *Large Language Models are not Models of Natural Language: they are Corpus Models*, pristupljeno 29. siječnja 2025. [Mrežno]. Adresa: <https://arxiv.org/pdf/2112.07055>
- [59] G. Dimopoulos, P. Barlet-Ros, C. Dovrolis, i I. Leontiadis, *Detecting network performance anomalies with contextual anomaly detection*, pristupljeno 30. siječnja 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/8078404>

- [60] M. Mobilio, M. Orru, O. Riganelli, A. Tundo, i L. Mariani, *Anomaly Detection As-a-Service*, pristupljeno 30. siječnja 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/8990365>
- [61] C. Chang, *Anomaly Detection Characterization*, pristupljeno 30. siječnja 2025. [Mrežno]. Adresa: https://link.springer.com/chapter/10.1007/978-1-4419-6187-7_14
- [62] V. Chandola, A. Banerjee, i V. Kumar, *Anomaly detection: A survey*, pristupljeno 30. siječnja 2025. [Mrežno]. Adresa: <https://dl.acm.org/doi/10.1145/1541880.1541882>
- [63] J. V. Dueholm, M. Siemon, R. T. Ionescu, T. B. Moeslund, i K. Nasrollahi, *Harborfront Anomaly Detection*, pristupljeno 30. siječnja 2025. [Mrežno]. Adresa: <https://link.springer.com/article/10.1007/s11063-024-11696-9>
- [64] C. Francq, C. Hurlin, S. Laurent, i J.-M. Zakoian, *Time Series for QFFE: Special Issue of the Journal of Time Series Analysis*, pristupljeno 1. veljače 2025. [Mrežno]. Adresa: <https://onlinelibrary.wiley.com/doi/epdf/10.1111/jtsa.12814>
- [65] A. I. McLeod, *Hyperbolic Decay Time Series*, pristupljeno 1. veljače 2025. [Mrežno]. Adresa: <https://onlinelibrary.wiley.com/doi/10.1111/1467-9892.00104>
- [66] J. Kong i R. Lund, *Seasonal count time series*, pristupljeno 1. veljače 2025. [Mrežno]. Adresa: <https://onlinelibrary.wiley.com/doi/epdf/10.1111/jtsa.12651>
- [67] C. Bandt i F. Shiha, *Order Patterns in Time Series*, pristupljeno 1. veljače 2025. [Mrežno]. Adresa: <https://onlinelibrary.wiley.com/doi/10.1111/j.1467-9892.2007.00528.x>
- [68] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. Nesnas, i M. Pavone, *Semantic anomaly detection with large language models*, pristupljeno 3. veljače 2025. [Mrežno]. Adresa: <https://link.springer.com/article/10.1007/s10514-023-10132-6>
- [69] T. Zhang, D. Cai, L. Qiu, i X. Leng, *Research on Anomaly Detection Methodology Combining Large Language Models*, pristupljeno 3. veljače 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/10653263>
- [70] J. Mbuya, D. Pfoser, i A. Anastasopoulos, *Trajectory Anomaly Detection with Language Models*, pristupljeno 3. veljače 2025. [Mrežno]. Adresa: <https://dl.acm.org/doi/10.1145/3678717.3691257>
- [71] K. Nasrollahi i A. Ntelopoulos, *CALLM: Cascading Eutoencoder and Large Language Model for Video Anomaly Detection*, pristupljeno 3. veljače 2025.

- [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/10755883>
- [72] Z. Gu, B. Zhu, G. Zhu, Y. Chen, M. Tang, i J. Wang, *AnomalyGPT: Detecting Industrial Anomalies Using Large Vision-Language Models*, pristupljeno 3. veljače 2025. [Mrežno]. Adresa: <https://ojs.aaai.org/index.php/AAAI/article/view/27963>
- [73] S. Schmidl, P. Weing, i T. Papenbrock, *Anomaly detection in time series*, pristupljeno 4. veljače 2025. [Mrežno]. Adresa: <https://dl.acm.org/doi/10.14778/3538598.3538602>
- [74] M. Leng, X. Lai, T. Guo-Iv, i X. Xu, *Time series representation for anomaly detection*, pristupljeno 4. veljače 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/5234775>
- [75] S. Karthik, H. V. Supreetha, i S. Sandhya, *Detection of Anomalies in Time Series Data*, pristupljeno 4. veljače 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/9683715>
- [76] O. Chaffard, M. Cavazza, i H. Prendinger, *Enhancing Large Language Models for Bitcoin Time Series Forecasting*, pristupljeno 5. veljače 2025. [Mrežno]. Adresa: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5052085
- [77] P. Boniol, E. Sylligardos, J. Paparrizos, P. Trahanias, i T. Palpanas, *ADecimo: Model Selection for Time Series Anomaly Detection*, pristupljeno 6. veljače 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/10597942>
- [78] H. Qiu, Y. Liu, N. Subrahmanya, i W. Li, *Granger Causality for Time-Series Anomaly Detection*, pristupljeno 6. veljače 2025. [Mrežno]. Adresa: <https://ieeexplore.ieee.org/document/6413806>
- [79] G. Colaboratory, *Colab is a hosted Jupyter Notebook service*, pristupljeno 7. veljače 2025. [Mrežno]. Adresa: <https://colab.google/>
- [80] Canva, *Home - Canva*, pristupljeno 7. veljače 2025. [Mrežno]. Adresa: <https://www.canva.com/>
- [81] Wikipedia, *Soil Moisture Active Passive*, pristupljeno 7. veljače 2025. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Soil_Moisture_Active_Passive
- [82] NASA, *Homepage NASA*, pristupljeno 7. veljače 2025. [Mrežno]. Adresa: <https://www.nasa.gov/>
- [83] Wikipedia, *Mars Science Laboratory*, pristupljeno 7. veljače 2025. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Mars_Science_Laboratory

Sažetak

Korištenje jezičnih modela za otkrivanje anomalija u vremenskim serijama

Vid Rebernak

Ovaj diplomski rad predstavlja komparativnu analizu pristupa temeljenih na velikim jezičnim modelima i klasičnih metoda u procesu otkrivanja anomalija u vremenskim serijama. Istraživanje procjenjuje četiri tradicionalne metode otkrivanja stršćih vrijednosti u odnosu na četiri velika jezična modela različitih arhitektura. Proučavanje koristi dosljedan okvir za pripremu podataka i implementaciju kliznih prozora, pretvaranje numeričkih očitavanja u tekstualni oblik za analizu perpleksnosti kod velikih jezičnih modela te primjenu kontaminacijskog parametra ili praga temeljenog na percentilu kod klasičnih metoda. Dobiveni rezultati naglašavaju značajne razlike u načinu detekcije odstupanja. Klasične metode pokazuju širu pokrivenost, ali i više lažnih upozorenja, dok veliki jezični modeli rezultiraju znatno većom preciznošću, ali manje identificiranih anomalija. Unatoč tome što oba pristupa pokazuju niske F1-rezultate zbog iznimno rijetkih stršćih vrijednosti, svaki od njih donosi određene prednosti: jednostavnost parametara za tradicionalne metode i napredno kontekstualno modeliranje za velike jezične modele. U radu se raspravlja o ključnim izazovima, uključujući preciznu prilagodbu jezičnih modela koja zahtijeva veliku količinu resursa i postavljanje praga ovisno o specifičnom području primjene. Također, naglašava se i važnost znanja o konkretnoj domeni za oba pristupa. Evaluacija upućuje na potencijalno korištenje hibridnih ili adaptivnih strategija kao obećavajućih prijedloga za buduća istraživanja, povezujući brzinu klasičnih tehnika s nijansiranim prepoznavanjem obrazaca koje nude veliki jezični modeli.

Ključne riječi: otkrivanje anomalija, vremenske serije, veliki jezični modeli, tradicionalni pristupi, komparativna analiza, složeni obrasci, perpleksnost, precizna prilagodba, određivanje praga

Abstract

Using Language Models for Anomaly Detection in Time Series

Vid Rebernak

This thesis presents a comparative analysis of approaches based on large language models and classical methods in the process of detecting anomalies in time series. The research evaluates four traditional outlier detection methods in relation to four large language models with different architectures. Consistent framework for data preparation and implementation of sliding windows, converting numeric readings into textual form for perplexity analysis in large language models and the application of contamination parameter or percentile-based threshold in classical methods were used while trying to establish a successful pipeline. Obtained results emphasize significant differences in the way both methods detect anomalies. Classical methods show wider coverage, but also more false alarms, while large language models result in significantly higher precision, but fewer outliers detected. Despite both approaches showing low F1-scores due to extremely rare anomalies, each of them brings certain advantages to the process: simplicity of parameters in traditional methods and advanced contextual modeling for large language models. The paper discusses key challenges, including precise fine-tuning of language models, which needs intensive resources, and domain-dependent thresholding. Also, the importance of subject-matter expertise for both approaches is highlighted. The conclusion suggests the potential use of hybrid or adaptive strategies as promising proposals for future research, connecting the speed of classical techniques with the nuanced pattern recognition offered by large language models.

Keywords: Anomaly Detection, Time Series, Large Language Models, Classical Methods, Comparative Analysis, Complex Patterns, Perplexity, Fine-Tuning, Thresholding