Samodejna priprava prometnih poročil za obveščanje javnosti prek radijskih postaj

Ruben Ferreira,[1] Rok Kužel,[2] Timotej Knez,[1] Slavko Žitnik[1]

[1] Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Ljubljana, Slovenija
ruben@ferreira.si, slavko.zitnik@fri.uni-lj.si, timotej.knez@fri.uni-lj.si
[2] Radiotelevizija Slovenija, Ljubljana, Slovenija
rok.kuzel@rtvslo.si

V prispevku predstavljamo razvoj sistema za samodejno generiranje prometnih poročil za radijsko poročanje z uporabo velikih jezikovnih modelov. Sistem temelji na podatkih nacionalnega prometnega portala NAP, iz katerega redno pridobiva podatke o prometnih dogodkih in delih na cesti. Pridobljeni podatki se shranjujejo, normalizirajo in pripravijo za nadaljnjo klasifikacijo ter generiranje besedil. Poseben poudarek namenjamo primerjavi različnih pristopov k uporabi jezikovnih modelov, in sicer tako modelov v oblaku kot lokalno naučenih oziroma prilagojenih modelov. Primerjavo izvajamo za naloge klasifikacije prometnih dogodkov in generiranja prometnih poročil, pri čemer upoštevamo kakovost izhodov, hitrost odziva, stroške uporabe, robustnost in primernost za slovenski jezik. Za generiranje rednih prometnih poročil uporabljamo dvostopenjski pristop, pri katerem se prometni zapisi najprej pretvorijo v strukturirane semantične fragmente, nato pa v končno slovensko besedilo, primerno za radijsko branje. Prispevek obravnava podatkovni cevovod, obdelavo vhodnih podatkov, primerjavo modelov in vlogo človeka pri končni uredniški presoji.

Ključne besede:
veliki jezikovni modeli
generiranje prometnih poročil
prometni podatki
radijsko poročanje
umetna inteligenca

Univerzitetna založba Univerze v Mariboru
ISBN 978-961-299-188-3
DOI https://doi.org/10.18690/um.feri.7.2026.20

1 Uvod

Prometne informacije so pomemben del vsakodnevnega radijskega poročanja, saj poslušalcem omogočajo hitro prilagoditev poti, izogibanje zastojem, pravočasno zaznavanje zapor ter upoštevanje drugih izrednih razmer na cestnem omrežju. Pri radijskem poročanju je poleg vsebinske pravilnosti ključna tudi časovna ažurnost informacij. Prometna nesreča, zapora voznega pasu ali daljša čakalna doba na mejnem prehodu ima lahko za poslušalce neposredne posledice, zato mora biti informacija pravočasno zaznana, pravilno interpretirana in predstavljena v kratki, razumljivi ter govorno primerni obliki.

V obstoječih uredniških procesih priprava prometnih poročil pogosto temelji na ročnem pregledu več prometnih virov. Urednik oziroma radijski voditelj mora iz razpoložljivih podatkov prepoznati najpomembnejše dogodke, izločiti manj relevantne ali zastarele informacije, združiti vsebinsko sorodne zapise in jih preoblikovati v tekoče besedilo, primerno za radijsko branje. Takšen proces je časovno zahteven, hkrati pa je občutljiv na nepopolnost vhodnih podatkov, podvajanje informacij, različne oblike zapisov in subjektivno presojo pomembnosti posameznega dogodka. Dodaten izziv predstavljajo nujna prometna obvestila, ki morajo biti zaznana in posredovana hitreje kot redna prometna poročila.

Razvoj velikih jezikovnih modelov odpira možnost avtomatizacije priprave prometnih poročil. Jezikovni modeli lahko iz strukturiranih prometnih zapisov tvorijo naravno besedilo, združujejo sorodne dogodke in prilagajajo slog besedila izbranemu komunikacijskemu kanalu. Vendar pa uporaba takšnih modelov v produkcijskem uredniškem okolju zahteva več kot zgolj neposredno pošiljanje surovih podatkov v model. Prometni podatki so časovno občutljivi, pogosto vsebujejo ponovitve, neenotne zapise, zastarele dogodke ali informacije z različno stopnjo pomembnosti. Poleg tega mora sistem preprečevati izmišljanje dejstev, ohranjati sledljivost do izvornih zapisov in omogočati človeku končno uredniško presojo.

V prispevku predstavljamo razvoj sistema za samodejno generiranje prometnih poročil za potrebe radijskega poročanja na RTV Slovenija. Sistem temelji na podatkih nacionalnega prometnega portala NAP, ki zagotavlja sprotne informacije o prometnih dogodkih ter delih na cesti. Razvita rešitev podatke redno pridobiva, jih shranjuje v podatkovno bazo, normalizira in pripravlja kot vhod za klasifikacijo ter generiranje poročil. Pri tem ločujemo med rednim generiranjem prometnih poročil, ki se lahko izvede na zahtevo oziroma v določenih časovnih intervalih, in nujnimi prometnimi obvestili, ki morajo biti po zaznavi čim prej prikazana uporabniku.

Osrednji del rešitve predstavlja kombinacija klasične obdelave podatkov, lokalnega klasifikacijskega modela in velikih jezikovnih modelov. Lokalni klasifikacijski model se uporablja za prepoznavanje nujnih prometnih dogodkov, kar omogoča hitro odločanje brez odvisnosti od zunanjih storitev. Za tvorjenje končnih besedil pa uporabljamo dvostopenjski pristop z velikim jezikovnim modelom. V prvi stopnji se prometni zapisi pretvorijo v strukturirane semantične fragmente, v drugi stopnji pa se ti fragmenti uredijo, združijo in preoblikujejo v slovensko besedilo, primerno za radijsko branje. Takšen pristop zmanjšuje možnost izmišljanja in podvajanja informacij ter omogoča boljši nadzor nad tem, katere informacije so bile uporabljene pri generiranju poročila.

Prispevek se osredotoča predvsem na podatkovni del rešitve in pripravo vhodov za jezikovne modele. Najprej opišemo vrste podatkov, ki so bile uporabljene v projektu: zgodovinske podatke, pridobljene za analizo, testiranje in učenje modelov, ter sprotne podatke iz dostopne točke NAP. Nato predstavimo postopke obdelave, povezovanja zgodovinskih prometnih zapisov z arhiviranimi radijskimi poročili, shranjevanja surovih podatkov, normalizacije zapisov in priprave podatkov za klasifikacijo ter generiranje poročil. S tem pokažemo, da je kakovost izhodnega besedila v veliki meri odvisna od ustrezno zasnovanega podatkovnega cevovoda in ne le od izbire samega jezikovnega modela.

2 Pregled področja

Avtomatizacija priprave besedil v uredniških in informacijskih okoljih ni nov pojav. Že pred širšo uporabo velikih jezikovnih modelov so se v praksi uveljavili sistemi za podatkovno vodeno generiranje besedil, pri katerih je bil vhod praviloma strukturiran podatkovni vir, izhod pa kratek novinarski ali poročevalski prispevek. Takšni sistemi so bili posebej primerni za področja, kjer se vsebina pogosto ponavlja, podatki pa so jasno strukturirani, na primer pri finančnih rezultatih, športnih izidih, volitvah, vremenskih opozorilih ali lokalnih obvestilih. V tem obdobju so rešitve večinoma temeljile na pravilih, predlogah in klasičnih pristopih naravnega jezikovnega tvorjenja, vendar so že uvedle osnovni produkcijski vzorec: zajem podatkov, izbor relevantnih informacij, pretvorbo v besedilo in uredniški nadzor.

Med najbolj znanimi primeri zgodnje produkcijske uporabe je sodelovanje agencije Associated Press s podjetjem Automated Insights in družbo Zacks Investment Research [2]. Sistem je iz strukturiranih finančnih podatkov samodejno pripravljal kratka poročila o poslovnih rezultatih podjetij, s čimer je Associated Press povečal obseg pokrivanja četrtletnih rezultatov na več kot 3000 prispevkov na četrtletje. Podoben primer predstavlja sistem Heliograf [3], ki ga je razvil The Washington Post in ga uporabil za samodejno pripravo kratkih posodobitev pri poročanju o olimpijskih igrah v Riu leta 2016 ter pozneje tudi pri drugih podatkovno intenzivnih temah. Oba primera kažeta, da je avtomatizacija najuspešnejša tam, kjer je mogoče novinarsko nalogo razdeliti na jasno določene korake in kjer je glavni izziv hitra pretvorba strukturiranih podatkov v razumljivo besedilo.

Naslednji razvojni korak predstavljajo sistemi, ki poleg tvorjenja besedila vključujejo tudi zaznavanje, filtriranje oziroma razvrščanje dogodkov. Primer takšnega pristopa je Quakebot [4] časopisa Los Angeles Times, ki spremlja obvestila ameriške geološke službe USGS in ob zaznavi potresa, ki ustreza določenim kriterijem, pripravi osnutek članka. Pri tem objava ni popolnoma avtomatizirana, saj končno odločitev sprejme urednik. Podoben premik od samega pisanja k širši podpori uredniškemu procesu je viden pri Reutersovih sistemih. Reuters Tracer [5] je bil zasnovan za zaznavanje in klasifikacijo potencialno relevantnih dogodkov iz družbenih omrežij v realnem času, Reuters pa je pozneje razvil tudi sistem Lynx Insight [6], ki iz podatkovnih virov pripravlja kratke podatkovno utemeljene vsebine, opozorila in sprožilce za novinarje. Ti primeri so pomembni, ker združujejo samodejno obdelavo podatkov, prag pomembnosti in človeka kot končnega odločevalca, kar je neposredno primerljivo s sistemi za časovno občutljive dogodke, kot so prometne nesreče, zapore ali daljši zastoji.

Z razvojem velikih jezikovnih modelov se je težišče avtomatizacije deloma premaknilo od predlog in pravil k bolj prilagodljivemu tvorjenju jezika. Takšni modeli lahko bolje prilagajajo slog, združujejo več vhodnih informacij, krajšajo besedilo in ga oblikujejo glede na ciljni komunikacijski kanal. Hkrati pa uvajajo nova tveganja, predvsem možnost izmišljanja dejstev, izgubo sledljivosti do vhodnih podatkov in težje preverjanje, katere informacije so bile uporabljene pri tvorjenju končnega besedila. Zato se v produkcijskih okoljih pogosto uporabljajo nadzorovani pristopi, pri katerih so vhodni podatki predhodno očiščeni, strukturirani in omejeni, končni rezultat pa ostane predmet človeškega pregleda.

Najbližje področju obravnavanega sistema so rešitve za samodejno pripravo prometnih obvestil za radijsko oziroma medijsko uporabo. Bauer Media in finsko državno podjetje Fintraffic [7] sta predstavila sistem za lokalno usmerjena prometna obvestila na radiu, pri katerem se uporabljajo umetna inteligenca, prometni podatki v realnem času in samodejno generirani glasovi. Namen takšnega pristopa je poslušalcem posredovati bolj lokalno relevantna prometna obvestila, ki so vezana na dejanske prometne razmere in lokacijo. Ta primer je pomemben, ker avtomatizacijo postavlja neposredno v radijski kontekst, kjer mora biti besedilo kratko, govorno naravno, hitro razumljivo in primerno za takojšnjo uporabo v programu.

Še neposrednejši komercialni primer predstavlja rešitev INRIX AI Traffic Reporter [8], ki je namenjena avtomatizaciji prometnega poročanja za televizijo in radio. INRIX jo opisuje kot sistem, ki iz prometnih podatkov in analitike samodejno pripravlja prometna poročila ter jih pretvarja v avdio oziroma video izhode za medijsko uporabo. Pri tem ne gre zgolj za konceptualno rešitev, saj so bila poročila, pripravljena z uporabo umetne inteligence, v Združenem kraljestvu že uvedena v radijskem okolju. Po poročanju portala RadioToday [13] je Bauer Media v sodelovanju z družbo INRIX takšna poročila uvedla na omrežjih Greatest Hits Radio in Hits Radio, kjer

se večina prometnih informacij podaja z računalniško generiranim glasom. Takšne rešitve kažejo, da je avtomatizacija prometnega poročanja že prešla iz raziskovalnega okolja v produkcijsko uporabo. Hkrati pa ostajajo odprta vprašanja glede kakovosti vhodnih podatkov, lokalizacije, jezikovne prilagoditve, uredniškega nadzora in primernosti za posamezne nacionalne prometne vire.

Sistem, predstavljen v tem prispevku, sledi istemu produkcijskemu vzorcu, vendar ga prilagaja slovenskemu prostoru in specifičnim zahtevam radijskega prometnega poročanja. Vhodni podatki izvirajo iz nacionalnega prometnega portala NAP [1], sistem pa jih pred uporabo shrani, normalizira in pripravi za nadaljnjo obdelavo. Posebnost rešitve je kombinacija lokalnega klasifikacijskega modela za zaznavanje pomembnih oziroma nujnih prometnih dogodkov ter dvostopenjskega generiranja z velikim jezikovnim modelom. Pri tem se prometni zapisi najprej pretvorijo v strukturirane semantične fragmente, nato pa v končno slovensko besedilo, primerno za radijsko branje. V primerjavi z obstoječimi primeri je poudarek predvsem na slovenskem jeziku, uporabi nacionalnega prometnega vira, sledljivosti do izvornih prometnih zapisov in ohranjanju človeka kot končnega urednika oziroma uporabnika sistema.

3 Podatki in njihova priprava

Kakovost samodejno generiranega prometnega poročila je neposredno odvisna od kakovosti, ažurnosti in strukture vhodnih podatkov. Pri razvoju sistema smo zato posebno pozornost namenili podatkovnemu cevovodu, ki obsega pridobivanje prometnih podatkov, njihovo shranjevanje, obdelavo, normalizacijo, zaznavanje nujnih dogodkov in pripravo strukturiranih vhodov za jezikovne modele. Podatke, uporabljene v projektu, lahko razdelimo v dve glavni skupini: zgodovinske podatke in sprotne podatke iz nacionalnega prometnega portala NAP [1].

3.1. Zgodovinski podatki

Zgodovinski podatki so bili uporabljeni predvsem v razvojni fazi sistema, in sicer za razumevanje strukture prometnih informacij, analizo uredniškega procesa, testiranje različnih pristopov generiranja poročil ter pripravo učnih in evalvacijskih primerov. Pri tem smo imeli na voljo dva vira podatkov: prometne zapise v obliki Excel datotek ter arhivirana radijska prometna poročila v obliki RTF datotek.

Prometni zapisi iz Excel datotek so vsebovali besedilna polja v obliki HTML, razdeljena na vsebinske sklope, kot so opozorila, vreme, prireditve, delo na cesti in omejitve za tovorna vozila, kot je prikazano v tabeli 2. Ti zapisi so predstavljali arhivsko obliko prometnih informacij, ki so po vsebini primerljive s podatki, pridobljenimi prek programskega vmesnika NAP. Arhivirana radijska poročila pa so predstavljala ciljno obliko izhodnega besedila, saj kažejo, kako uredniki iz širšega nabora prometnih informacij izberejo najpomembnejše dogodke in jih preoblikujejo v kratko, govorno primerno poročilo.

Ker prometni zapisi in radijska poročila niso bili neposredno povezani z enoličnimi identifikatorji, smo pripravili postopek približnega povezovanja. Za vsako radijsko poročilo smo iz glave dokumenta izluščili naslov, datum, čas in program, nato pa poiskali prometne zapise, ki so nastali v časovnem oknu pred objavo poročila. Med kandidati smo nato izvedli besedilno primerjavo. Besedila smo predhodno očistili, pretvorili v male črke, odstranili ločila, kontrolne znake in slovenske zaustavitvene besede (angl. *stop words*), nato pa za razvrščanje uporabili predstavitev TF-IDF [14] in kosinusno podobnost [15].

S takšnim postopkom smo pridobili približne pare med vhodnimi prometnimi zapisi in končnimi radijskimi poročili. Ti pari so bili pomembni iz več razlogov. Prvič, omogočili so testiranje in prilagajanje različnih velikih jezikovnih modelov, saj smo lahko primerjali generirana poročila z dejanskimi uredniškimi poročili. Drugič, ker so bila pri zgodovinskih podatkih na voljo tudi ciljna besedila, je bilo mogoče del naloge obravnavati kot nadzorovano učenje oziroma kot evalvacijo modelov na podlagi znanih izhodov. Tretjič, podatki so omogočili pripravo primerov za učenje in preverjanje klasifikacijskih modelov, namenjenih ločevanju pomembnih oziroma nujnih prometnih dogodkov od manj pomembnih. Poleg tega so zgodovinski podatki služili za analizo vzorcev v prometnih informacijah in boljše razumevanje uredniških odločitev pri pripravi radijskih poročil.

Analiza zgodovinskih podatkov je pokazala, da se izvorni prometni zapisi in končna radijska poročila pomembno razlikujejo. Izvorni zapisi so pogosto daljši, bolj tehnični in vsebujejo več podrobnosti, medtem ko radijsko poročilo zahteva izbor najpomembnejših informacij, krajšanje, združevanje sorodnih dogodkov in prilagoditev jezika govornemu mediju. To je vplivalo na zasnovo sistema, predvsem na odločitev za strukturirano pripravo vhodnih podatkov pred končnim generiranjem besedila.

Tabela 1: Pregled zgodovinskih prometnih zapisov.

Obdobje	Št. zapisov	Povp. zapisov/dan
2022	55.001	150,7
2023	57.094	156,4
2024	58.278	159,2
Skupaj	170.373	155,4

V tabeli 1 »zapis« označuje eno vrstico v arhivski datoteki Excel oziroma časovno označen posnetek prometnih informacij, kot je prikazano v tabeli 2. Posamezen zapis lahko vsebuje več vsebinskih sklopov in več prometnih dogodkov, zato število zapisov ne predstavlja števila enoličnih prometnih dogodkov.

Tabela 2: Struktura enega zgodovinskega zapisa.

Polje v datoteki Excel	Pomen
LegacyId	Enolični identifikator zapisa v arhivskem izvozu
Datum	Čas nastanka oziroma posodobitve prometnega zapisa
Operater	Podatek o osebi oziroma viru, ki je pripravil ali uredil zapis.
Pomembno	Sklop za posebej izpostavljene oziroma nujne prometne informacije.
Nesreče	Informacije o prometnih nesrečah in njihovem vplivu na promet.
Vreme	Prometne informacije, povezane z vremenskimi razmerami.
Ovire	Informacije o ovirah na vozišču ali drugih motnjah v prometu.
DeloNaCesti	Informacije o delih na cesti, zaporah in spremenjeni prometni ureditvi.
Opozorila	Splošna opozorila, prometne napovedi, omejitve in druga obvestila.
MednarodneInformacije	Informacije, povezane s tujino, mejnimi prehodi ali mednarodnimi omejitvami.
Zastoji	Informacije o zastojih, kolonah in daljših potovalnih časih.
Splošno	Splošne prometne informacije, ki niso nujno vezane na en sam konkreten dogodek.

3.2. Sprotni podatki iz dostopnih točk NAP

V produkcijskem delu sistema se podatki pridobivajo iz sistema NAP. Sistem tehnično podpira tri vire: prometne dogodke, dela na cesti in prometna poročila. Za sprotno generiranje radijskih obvestil sta najpomembnejša vira prometni dogodki in dela na cesti, saj vsebujeta posamezne aktualne zapise o razmerah na cestah. Vir prometnih poročil je v sistemu prav tako predviden, vendar se uporablja predvsem za shranjevanje in primerjalno analizo, ne pa kot glavni vhod za generiranje posameznih nujnih obvestil.

Zapisi, pridobljeni iz dostopnih točk NAP, so strukturirani kot posamezni prometni dogodki oziroma zapisi o delih na cesti. Posamezen zapis praviloma vsebuje enolični identifikator, čas posodobitve, podatek o cesti oziroma

cestnem odseku, kategorijo ceste, vzrok dogodka, opis razmer in morebitno dodatno pojasnilo. To je pomembno, ker sistem teh podatkov ne obravnava zgolj kot prosta besedila, temveč kot strukturirane dogodke, ki jih je mogoče shraniti, primerjati, razvrščati in uporabiti kot vhod v nadaljnje modele.

Pri prehodu iz zgodovinskih podatkov na sprotne podatke NAP je bilo pomembno, da se sistem ne opira na podatke, ki so bili prisotni samo v arhivskih izvozih iz Excela. Modeli, razviti ali preverjeni na zgodovinskih podatkih, morajo namreč tudi v produkcijskem delovanju prejemati primerljivo strukturo vhodnih podatkov. Zato smo pri pripravi podatkov uporabili predvsem tiste vsebinske sklope iz zgodovinskih zapisov v Excelu, ki jih je mogoče neposredno ali vsaj delno povezati z vrednostmi polja »vzrok« v sprotnih podatkih NAP.

Primerjava je pokazala, da so neposredno primerljivi predvsem sklopi delo na cesti, zastoji, ovire, nesreče, vreme in omejitve za tovorna vozila. Ti sklopi so bili zato uporabljeni kot glavna osnova za pripravo primerljivih vhodnih podatkov. Sklop »Pomembno« je bil uporaben le delno, saj se v njem v zgodovinskih podatkih pojavljajo varnostno kritični dogodki, kot je vožnja v nasprotno smer, medtem ko se v podatkih NAP tak dogodek pojavi kot samostojna vrednost polja »vzrok«. Sklopi brez neposredne ali delne ustreznice, na primer splošne informacije, mednarodne novice in opozorila, niso bili uporabljeni kot glavna osnova za primerjavo oziroma učenje modelov.

Tabela 3: Primerljivost zgodovinskih podatkov s podatki, pridobljenimi iz dostopne točke.

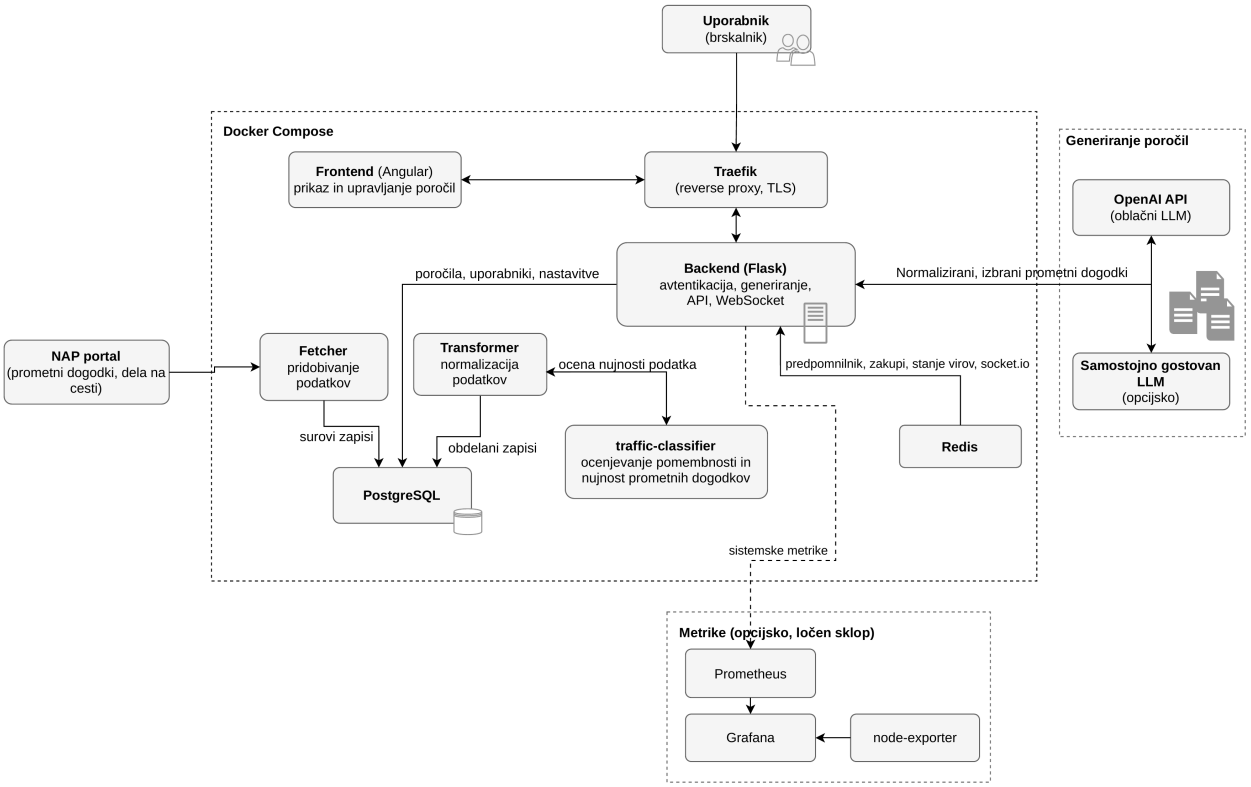
Vsebinski sklop v zgodovinskih podatkih iz Excela	Vrednost »vzrok« v podatkih NAP Primerljivost	Primerljivost
Delo na cesti	Delo	Primerljivo
Zastoji	Zastoj	Primerljivo
Ovire	Ovire	Primerljivo
Nesreče	Nesreča	Primerljivo
Vreme	Vreme	Primerljivo
Omejitve za tovorna vozila	Tovornjaki	Primerljivo
Pomembno	Vožnja v nasprotno smer	Delno primerljivo
Splošno	/	Ni primerljivo
Mednarodne novice	/	Ni primerljivo
Opozorila	/	Ni primerljivo

Pri uporabi sprotnih podatkov je bil pomemben izziv ločevanje nujnih oziroma pomembnih prometnih dogodkov od manj pomembnih. Dostopne točke NAP zagotavljajo aktualne prometne informacije, vendar dogodki, ki zahtevajo takojšnje radijsko obvestilo, niso posebej označeni kot nujni. Prometna nesreča, daljši zastoj, ovira na pomembni cesti ali vožnja v nasprotno smer se zato v viru pojavi kot običajen prometni dogodek. Zaradi tega smo v sistem vključili klasifikacijski model, ki na podlagi vsebine dogodka presodi, ali gre za pomemben dogodek, ki ga je treba nemudoma posredovati uporabniškemu vmesniku.

4 Zasnova in implementacija sistema

4.1. Arhitektura sistema

Sistem je zasnovan kot skupina med seboj ločenih storitev, ki tečejo v samostojnih vsebnikih Docker [16] in med seboj komunicirajo prek notranjega omrežja, kot je prikazano na sliki 1. Takšna zasnova omogoča neodvisno namestitev, ponovni zagon in skaliranje posameznih delov sistema, hkrati pa loči odgovornosti med pridobivanjem podatkov, njihovo obdelavo, klasifikacijo, generiranjem poročil in predstavitvijo uporabniku. V nadaljevanju so opisane posamezne storitve, kot so definirane v namestitveni konfiguraciji sistema.



Slika 1: Arhitektura sistema za generiranje prometnih poročil.

4.1.1. Reverse proxy (Traefik)

Traefik [17] predstavlja vstopno točko sistema. Poskrbi za zaključevanje povezav TLS (samodejno pridobivanje in obnavljanje potrdil prek storitve Let's Encrypt [18]) ter usmerjanje prihajajočih zahtev glede na domeno in pot: zahteve na "/api" in "/socket.io" preusmeri na zaledno aplikacijo, vse ostale pa na odjemalsko aplikacijo. S tem je edina komponenta, neposredno izpostavljena na javnem omrežju. Traefik na ločeni, izključno interni vstopni točki izpostavlja tudi svoje metrike prometa z uporabo sistema za spremljanje Prometheus [19], dostop do vmesnika Grafana [20] pa usmerja na podpoti "/monitoring" znotraj obstoječe domene, tako da spremljanje delovanja ne potrebuje ločene poddomene ali dodatnega potrdila TLS.

4.1.2. Zaledna aplikacija (angl. backend)

Zaledna aplikacija, razvita z ogrodjem Flask [21], izvaja poslovno logiko sistema in uporabnikom izpostavlja HTTP API ter komunikacijo v realnem času prek protokola WebSocket. Upravlja uporabnike, prometne zapise, generirana poročila, nastavitve sistema in povratne informacije urednikov.

Storitvena plast povezuje klasifikacijo dogodkov, dvostopenjsko generiranje poročil, časovno razporejanje in obveščanje uporabnikov. Redis [22] se uporablja za predpomnjenje rezultatov, koordinacijo razporejenih opravil

in razpošiljanje dogodkov med instancami, vsi klici jezikovnih modelov pa se beležijo zaradi sledljivosti in nadzora nad stroški. Podrobnosti izbire zapisov, predpomnjenja in generiranja so opisane v razdelku 4.4.

4.1.3. Fetcher in Transformer

Procesa Fetcher in Transformer sta zagnana iz iste izvorne slike kot zaledna aplikacija, vendar z drugačno vstopno točko, kar omogoča neodvisno skaliranje in ponovni zagon pridobivanja podatkov, ločeno od njihove obdelave.

Proces Fetcher redno dostopa do dostopnih točk portala NAP (prometni dogodki, dela na cesti, prometna poročila), preverja spremembe, da se izogne nepotrebnemu prenosu nespremenjenih podatkov, ter surove zapise shranjuje v podatkovno bazo. Poleg tega spremlja razpoložljivost posameznega vira in stanje objavlja prek Redisa in protokola WebSocket.

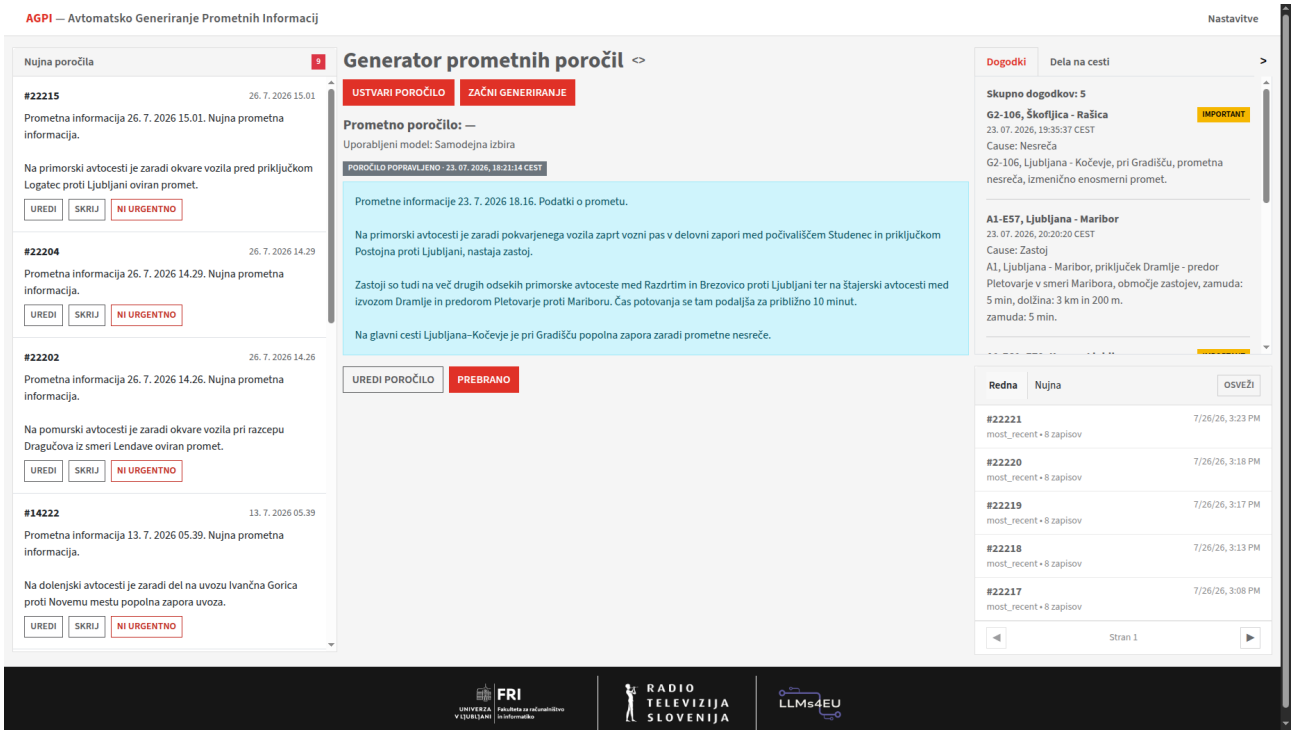
Proces Transformer surove zapise pretvarja in normalizira v obdelane tabele, kliče klasifikacijski model za oceno nujnosti posameznega dogodka ter zaznava dogodke, ki zahtevajo takojšnjo obravnavo in sprožitev nujnega poročila.

4.1.4. Klasifikator prometnih dogodkov

Klasifikator je samostojna storitev, razvita z ogrodjem FastAPI [23]. Uporablja naučeni model ModerniBERT, za slovenski jezik prilagojeno različico modela ModernBERT [24], proces Transformer pa jo kliče prek internega zahtevka REST, zaščitenega s ključem API. Storitev je namensko ločena od zaledne aplikacije, saj uporablja drugačen tehnološki sklad (Python z modelom PyTorch [25]) in ima drugačne strojne zahteve. Omogoča hitro lokalno klasifikacijo nujnosti dogodkov brez odvisnosti od zunanjega ponudnika; kot alternativa je na voljo tudi klasifikacija prek oblačnega jezikovnega modela (opisano v razdelku 4.3).

4.1.5. Odjemalska aplikacija (angl. frontend)

Odjemalska aplikacija je implementirana kot enostranska spletna aplikacija v ogrodju Angular [27]. Urednikom omogoča pregled prometnih zapisov, generiranje in urejanje poročil, pregled izvornih zapisov ter podajanje povratnih informacij. Nujna poročila, spremembe stanja virov in potek generiranja se posredujejo prek povezave WebSocket, zato ročno osveževanje strani ni potrebno. Administratorski del omogoča upravljanje dostopov ter konfiguracijo modelov in parametrov poročanja. Osnovni uporabniški vmesnik aplikacije je prikazan na sliki 2.



Slika 2: Uporabniški vmesnik.

Uporabniški vmesnik je razdeljen na tri glavne vsebinske dele. Na levi strani je prikazan seznam nujnih prometnih poročil, na katerem lahko urednik posamezno poročilo uredi, skrije ali označi kot napačno nujno poročilo. Osrednji del je namenjen ustvarjanju, prikazu in urejanju generiranega prometnega poročila ter spremljanju njegovega stanja. Na desni strani so prikazani izvorni prometni dogodki in dela na cesti, ki jih je mogoče uporabiti pri generiranju, pod njimi pa je seznam predhodno ustvarjenih rednih in nujnih poročil. Dostop do nastavitev aplikacije je omogočen v zgornjem desnem delu vmesnika.

4.1.6. Podatkovna in operativna infrastruktura

PostgreSQL [28] predstavlja trajno podatkovno shrambo za prometne zapise, poročila, uporabnike, nastavitve in metapodatke klicev jezikovnih modelov. Redis [22] se uporablja za predpomnjenje, koordinacijo opravil, stanje delovnih procesov in komunikacijo med instancami. Operativno spremljanje zagotavljata Prometheus [19] in Grafana [20], ki prikazujeta aplikacijske in sistemske metrike. Zunanji storitvi sistema sta portal NAP kot podatkovni vir in OpenAI API [26] kot oblačni ponudnik jezikovnih modelov.

4.2. Pridobivanje in shranjevanje podatkov

Proces Fetcher redno dostopa do treh dostopnih točk portala NAP – za prometne dogodke, dela na cesti in prometna poročila – do vsake v svojem, neodvisno nastavljenem časovnem intervalu. Dostop je avtenticiran prek OAuth2: fetcher najprej pridobi dostopni žeton z uporabniškim imenom in geslom, nato pa ga do izteka veljavnosti obnavlja z osvežitvenim žetonom, in sicer proaktivno tik pred iztekom ter reaktivno ob morebitnem odgovoru 401.

Preden fetcher prenese celoten odgovor, s pogojno zahtevo preveri, ali se je nabor podatkov od zadnjega poizvedovanja sploh spremenil. Če vir vrne 304 (ni sprememb), se noben zapis ne prepiše – namesto tega se za vrstice, ki so bile del zadnjega dejansko prenesenega posnetka, le pomakne časovna oznaka zadnje potrditve prisotnosti, s čimer sistem loči med »dogodek se od prejšnjega preverjanja ni spremenil« in »dogodek v tem posnetku ni bil več prisoten«, ne da bi za to moral znova zapisovati nespremenjeno vsebino.

Prejeti zapisi se shranjujejo v ločene surove tabele v PostgreSQL, ločeno za vsak vir, z enoličnim identifikatorjem dogodka, ki ga dodeli NAP. Znotraj enega prenesenega posnetka se zapisi z istim identifikatorjem pred zapisom v bazo združijo, sam zapis v bazo pa se izvede po tem identifikatorju: obstoječa vrstica se prepiše le, če je časovni žig spremembe iz izvornega sistema resnično novejši od že shranjenega oziroma če ta prej ni bil znan – s čimer poznejša, a vsebinsko zastarela dostava podatkov ne more prepisati novejšega stanja. Poleg žiga zadnje spremembe vsaka vrstica hrani tudi prvi trenutek, ko je bil zapis zaznan in zadnji trenutek, ko je bil še potrjen kot prisoten; ta dva časovna žiga sta osnova tako za filter zastarelosti vira kot za sprotni pogoj pri generiranju poročil (glej razdelek 4.4). Ob dejanski vsebinski spremembi se zapis dodatno označi kot neobdelan, kar zaledni storitvi »Transformer« pove, da ga mora znova pretvoriti in klasificirati. Vzporedno s shranjevanjem storitev »Fetcher« za vsak vir podatkov vodi tudi stanje razpoložljivosti glede na zaporedne uspehe in neuspehe poizvedb, ki ga objavlja prek Redisa in WebSocketa, ter piše lastni srčni utrip, s čimer je mogoče delovanje procesa pridobivanja podatkov spremljati ločeno od delovanja posameznega vira NAP.

4.3. Klasifikacija pomembnosti prometnih dogodkov

Cilj klasifikacije pomembnosti prometnih dogodkov je razločevanje med nujnimi in nenujnimi prometnimi dogodki, kar omogoča njihov prikaz v prednostnem delu uporabniškega vmesnika. Sprva smo za klasifikacijo uporabljali plačljivi model GPT. Zaradi klasifikacije velikega števila dogodkov je ta pristop zahteval veliko plačljivih klicev programskega vmesnika OpenAI. Za izboljšanje hitrosti in zmanjšanje stroškov smo zasnovali lasten namenski model, s katerim lahko klasificiramo dogodke. Da bi zagotovili večjo prilagodljivost in robustnost sistema, nujnosti nismo napovedovali neposredno, temveč smo opredelili več lastnosti dogodkov, na podlagi katerih je mogoče z uporabo pravil določiti, ali je dogodek nujen. Model prepoznava naslednje lastnosti: vožnja v napačno smer, zapora pasu zaradi nesreče, popolna zapora ceste zaradi nesreče, zapora ceste iz drugih razlogov, ovira na cesti, velika gneča (več kot 30 minut), varnostno tveganje, načrtovan dogodek in tip ceste. Preizkusili smo

več različnih modelov: diskriminativne modele SloBERTa, CroSloEngualBERT in ModerniBERT ter generativna modela GaMS-2B in GaMS-9B. Najboljše rezultate smo dosegli z modelom ModerniBERT, ki predstavlja različico modela ModernBERT, doučeno za slovenski jezik.

Za učenje namenskega klasifikacijskega modela smo pripravili uravnoteženo učno množico s skupno 58.000 primeri. Postopek je najprej obsegal zajem približno 15.000 dejanskih prometnih dogodkov iz preteklih poročil, ki smo jim vrednosti atributov samodejno določili z uporabo velikega jezikovnega modela GPT-5.2. Ker so bile določene kritične lastnosti (npr. vožnja v napačno smer) v realnih podatkih izjemno redke, smo z jezikovnim modelom sintetično ustvarili dodatne dogodke. Ustvarjanje smo nadaljevali, dokler za vsako vrednost posameznega atributa nismo zbrali vsaj 5.000 primerov, s čimer smo uspešno preprečili neenakomerno porazdelitev razredov in modelu omogočili stabilnejše učenje.

4.4. Generiranje prometnih poročil

4.4.1. Postopek generiranja prometnih poročil

Sistem loči med rednimi poročili, ki se generirajo na zahtevo urednika ali samodejno po nastavljenem urniku, zgodovinskimi poročili za izbran časovni trenutek, ročno sestavljenimi poročili iz izbranih zapisov ter nujnimi poročili, ki so vedno vezana na natanko en dogodek iz vira prometnih dogodkov. Za redna in nujna poročila velja dodatna, na zaledju uveljavljena zahteva po sprotnosti: v poštev pridejo le zapisi, ki so bili v zadnjem posnetku vira NAP zaznani v zadnjih 60 sekundah, s čimer se poročilo ne more sestaviti iz zapisov, ki jih je vmesni cikel pridobivanja podatkov že prehitel.

Kandidatni dogodki in dela na cesti se pred generiranjem izberejo in razvrstijo po administratorsko nastavljivi zgornji meji na posamezen vir ter po dveh dodatnih mehanizmih. Prvi je neobvezni filter zastarelosti vira: zapisi, katerih izvorni podatek je starejši od administratorsko nastavljenega praga v urah, se izločijo, saj lahko vir NAP dlje časa ne osveži dolgoročnih zapisov, kot so dela na cesti, ki pa niso pomembna za takojšnje radijsko poročanje. Drugi mehanizem je deterministična razvrstitev po pomembnosti: za dogodke se namesto preproste razvrstitve po kategoriji ceste uporablja razširjen vrstni red, ki najprej upošteva vzrok dogodka in šele nato skupino ceste (avtoceste/hitre ceste, regionalne ceste, lokalne ceste/javne poti) – vožnja v nasprotno smer je vedno na prvem mestu, sledijo nesreče na vseh cestah, nato zastoji, vreme, okvare in ovire, razvrščeni najprej po pomembnejši skupini cest, na koncu pa še dela na cesti in omejitve za tovorna vozila. Dela na cesti se razvrščajo po enostavnejši shemi, ki temelji neposredno na kategoriji ceste.

Poleg tega smo uvedli deterministični filter pomembnosti zastojev, ki pred generiranjem izloči zastoje pod administratorsko nastavljenim pragom: na avtocestah in hitrih cestah mora biti v zapisu navedena zamuda v minutah, ki dosega vsaj nastavljeni prag, na drugih cestah pa mora dolžina kolone v kilometrih dosegati vsaj nastavljeni prag.

Za nenujna poročila je pred klicem jezikovnega modela vključeno tudi predpomnjenje v Redisu, ki zmanjša odzivni čas in strošek klicev za vsebinsko enake zahteve. Vhodni zapisi in namig za glavo poročila se pred izračunom ključa normalizirajo: iz zapisov se odstranijo časovno spremenljiva polja (npr. čas zadnje zaznave), iz namiga za glavo pa datum in ura, tako da dva vsebinsko enaka zahtevka, ki se razlikujeta le v trenutku klica, ustvarita isti ključ; vrstni red zapisov se pri tem ohrani, saj neposredno določa oštevilčenje v prvi stopnji generiranja. Normaliziran zapis se skupaj z izbrano strategijo in različico shematike predpomnilnika kanonično serializira in zgosti v vsebinsko naslovljen ključ. Pod tem ključem se v Redisu z administratorsko nastavljivim časom veljavnosti shrani že izračunano poročilo; pri zadetku se preskočita obe stopnji klica jezikovnega modela in poročilo le dobi nov datum in uro glave.

Generiranje besedila poteka v dveh stopnjah z velikim jezikovnim modelom. V prvi stopnji model za vsak vhodni zapis neodvisno izdela en normaliziran element: strukturirana semantična polja (vrsta dogodka, razred ceste, oznake) ter kratke, za radio pripravljene slovenske besedilne fragmente (uvodna fraza, fraza vzroka, fraza lokacije, fraza smeri, fraza učinka in morebitna dodatna fraza, npr. za zamudo ali obvoz), ki jih druga stopnja sestavi v celotne povedi. Model v tej stopnji ne sme razvrščati, združevati ali primerjati zapisov med seboj, ne sme si

izmišljati vzroka, ki v vhodu ni izrecno naveden, in ne sme v izhod vključiti cestnih oznak, evropskih številk cest ali oznak odsekov oziroma kilometrskih oznak – vhodni zapisi so bili namreč že izbrani z deterministčnim filtriranjem na zaledju, zato prva stopnja ne sme dodatno uredniško presojati pomembnosti posameznega dogodka, temveč sme element označiti kot neuporaben le, če je strukturno neuporaben (npr. brez vsebinsko smiselnega podatka o prometu). Druga stopnja iz teh normaliziranih elementov sestavi končno besedilo: uredi vrstni red, združi elemente na isti cesti v en odstavek in za vsak vključen element ohrani sklic na izvorni zapis.

Klic vsake stopnje je zasnovan z rezervnim mehanizmom (angl. *fallback*) med ponudniki jezikovnih modelov: če je konfiguriran lasten (samostojno gostovan) model, se ta poskusi najprej, ob napaki pa se klic samodejno ponovi z oblačnim ponudnikom OpenAI, razen če je administrator izrecno izbral izključno lastni model. Odgovor vsake stopnje je strukturiran skladno z vnaprej opredeljeno shemo JSON, ki jo zaledje dodatno preveri in po potrebi popravi. Vsak klic jezikovnega modela – sistemsko sporočilo (in njegov zgoščeni odtis), uporabniški poziv, odgovor ter morebitna napaka – se zabeleži ločeno za vsako stopnjo, kar omogoča sledljivost, nadzor nad stroški in analizo napak.

Nujna poročila imajo lastno logiko zunaj zgoraj opisanega predpomnjenja: sprožitev, generiranje in oddaja so poenoteni v eni storitvi, ki prek protokola WebSocket naznani začetek in konec generiranja, po uspešnem zaključku izvorni zapis označi kot uporabljen (da se v naslednjih poročilih ne podvaja) ter pred oddajo izrecno preveri, da poročilo dejansko vsebuje natanko en sprožitveni dogodek in nobenega dela na cesti. Shranjevanje poročila, povezav do izvornih zapisov in zapisov o klicih jezikovnega modela poteka v eni sami podatkovni transakciji, tako da je poročilo bodisi v celoti shranjeno z vsemi sledljivimi povezavami bodisi sploh ne.

Vsako shranjeno poročilo poleg besedila in glave hrani tudi metapodatke, potrebne za poznejšo rekonstrukcijo in vpogled: strategijo generiranja, identifikatorje uporabljenih izvornih zapisov po viru ter posnetek vseh administratorsko nastavljivih parametrov generiranja, ki so veljali v trenutku nastanka poročila – ali je bil vključen filter zastarelosti vira in s kakšnim pragom, ali je bilo predpomnjenje omogočeno in s kakšnim časom veljavnosti, ter ali je bil vključen filter pomembnosti zastojev in z ustreznimi pragovi. Ta posnetek je potreben, ker so ustrezne nastavitve sistemsko globalne in se ob vsaki spremembi prepišejo, zato bi bila brez posnetka izgubljena informacija o tem, s kakšnimi nastavitvami je bilo generirano posamezno starejše poročilo.

4.4.2. Primer rednega prometnega poročila

Pri prikazanem primeru rednega poročila sta bila za generiranje izbrana dva prometna dogodka, ki se nanašata na različni cesti. Prva stopnja generiranja je za vsak vhodni zapis pripravila ločen strukturiran element z normaliziranimi podatki in besedilnimi fragmenti. Druga stopnja je iz obeh elementov oblikovala dve samostojni povedi. Ker se dogodka nanašata na različni cesti, ju sistem ni združil v isti odstavek.

Tabela 4: Primer izhodov prve in druge stopnje generiranja prometnega poročila.

Vhod	Stopnja 1 (izhod)	Stopnja 2 (izhod)
3958: "A4, Maribor - Gruškovje (Hrvaška), priključek Marjeta - priključek Zlatoličje v smeri Podlehnika, okvara vozila, oviran promet."	{ "source_ref": "3958", "road": "podravska avtocesta", "road_class": "avtocesta_hitra", "event_type": "drugo", "lead_phrase": "na podravski avtocesti", "cause_phrase": null, "location_phrase": "med priključkoma Marjeta in Zlatoličje", "direction_phrase": "proti Podlehniku", "effect_phrase": "oviran promet zaradi okvare vozila", "tags": ["okvara", "oviran_promet"] }	Na podravski avtocesti je med priključkoma Marjeta in Zlatoličje proti Podlehniku oviran promet zaradi okvare vozila.

3960: "G1-11, Šmarje - Koper, zastoj, dolžina: 2 km in 700 m."	{ "source_ref": "3960", "road": "Šmarje–Koper", "road_class": "regionalna_glavna", "event_type": "zastoj", "lead_phrase": "na glavni cesti Šmarje–Koper", "cause_phrase": null, "location_phrase": null, "direction_phrase": null, "effect_phrase": "zastoj", "tags": ["zastoj"] }	Na glavni cesti Šmarje–Koper je zastoj.

Končno poročilo, prikazano uredniku, je nato oblikovano na naslednji način:

Prometna informacija 19. 7. 2026 21.36. Podatki o prometu.

Na podravski avtocesti je med priključkoma Marjeta in Zlatoličje proti Podlehniku oviran promet zaradi okvare vozila.

Na glavni cesti Šmarje–Koper je zastoj.

4.4.3. Primer nujnega prometnega poročila

Pri nujnem poročilu je postopek generiranja vezan na en sam prometni dogodek, ki ga je klasifikator označil kot pomembnega. V prikazanem primeru je šlo za okvaro vozila na primorski avtocesti pred počivališčem Ravbarkomanda v smeri Ljubljane. Iz dogodka je prva stopnja pripravila normalizirane besedilne fragmente, druga stopnja pa jih je sestavila v končno poved nujnega poročila.

- Vhod: "A1, Koper - Ljubljana, pred počivališčem Ravbarkomanda v smeri Ljubljane, okvara vozila, oviran promet."
- Stopnja 1: {"road": "primorska avtocesta", "road_class": "avtocesta_hitra", "cause_phrase": "zaradi okvare vozila", "location_phrase": "pred počivališčem Ravbarkomanda", "direction_phrase": "proti Ljubljani", "effect_phrase": "oviran promet", "tags": ["okvara","oviran_promet"]}
- Stopnja 2: "Na primorski avtocesti je zaradi okvare vozila pred počivališčem Ravbarkomanda proti Ljubljani oviran promet."

Končno poročilo, prikazano uredniku, je nato oblikovano tako:

Prometna informacija 22. 7. 2026 18.43. Nujna prometna informacija.

Na primorski avtocesti je zaradi okvare vozila pred počivališčem Ravbarkomanda proti Ljubljani oviran promet.

4.5. Namestitev in operativni vidiki

Sistem se namešča kot skupina vsebnikov Docker, opisana v razdelku 4.1, ločeno za razvojno in produkcijsko okolje: razvojna konfiguracija izpostavi vrata za neposreden dostop in uporablja kodo iz delovnega imenika, produkcijska pa storitve izpostavi izključno prek storitve Traefik s samodejno pridobljenim potrdilom TLS. Vsaka storitev ima nastavljeno politiko samodejnega ponovnega zagona ter, kadar je smiselno, tudi preverjanje stanja (angl. *healthcheck*) na aplikacijski ravni. Spremembe podatkovne sheme se uveljavljajo prek migracijskega orodja Alembic [29], kar omogoča nadzorovan in ponovljiv prehod med različicami sheme brez ročnih posegov v bazo.

Del obvladovanja tveganj je vgrajen že v samo aplikacijo, neodvisno od infrastrukturnega spremljanja: dvostopenjsko generiranje poročil ima rezervni mehanizem med ponudniki jezikovnih modelov, predpomnjenje v Redisu je zasnovano tako, da vsaka napaka pri dostopu do Redisa generiranja poročila ne prekine, temveč ga le izvede brez predpomnilnika, vsak klic jezikovnega modela pa se zabeleži zaradi sledljivosti in naknadne analize napak. Prav tako se ob vsakem generiranju v metapodatkih poročila shrani posnetek takrat veljavnih nastavitev (filter zastarelosti vira, predpomnjenje, filter pomembnosti zastojev), kar omogoča naknadno razumevanje delovanja sistema, tudi če so se administratorske nastavitve medtem spremenile.

Za operativno spremljanje delovanja je vzpostavljen sklop orodij Prometheus, Grafana in node-exporter, ki v realnem času prikazuje zasedenost niti in bazena povezav do baze glede na njune konfigurirane zgornje meje, trajanje in prepustnost generiranja poročil ter osnovne metrike gostitelja. To upravljanje tveganj je trenutno pretežno opazovalno: nadzorna plošča je na voljo administratorju, ki jo mora sam pregledati, obveščanje o presežnih vrednostih pa ni samodejno – Prometheus ne izvaja pravil za alarme (angl. *alerting rules*), v namestitev pa ni vključena komponenta za obveščanje, prek katere bi bila anomalija (npr. izčrpan bazen povezav do baze ali neobičajno visok delež neuspešnih tvorjenj poročil) samodejno sporočena administratorju, namesto da jo ta opazi šele ob naslednjem pregledu nadzorne plošče.

Nekateri operativni vidiki sistema ostajajo predmet nadaljnjega dela. Trenutno se surovi in obdelani prometni zapisi v podatkovni bazi hranijo brez samodejnega arhiviranja ali brisanja, zato bo treba pri daljši produkcijski uporabi uvesti politiko hrambe podatkov. Prav tako je predvidena vzpostavitev rednega varnostnega kopiranja podatkovne baze in dokumentiranega postopka obnovitve po izpadu. Namestitev sistema za zdaj poteka ročno, zato bi nadaljnja avtomatizacija uvajanja omogočila bolj ponovljive in varnejše posodobitve. Dodatna možnost izboljšave je tudi centralizirano zbiranje dnevnikov posameznih storitev, kar bi olajšalo naknadno analizo napak in spremljanje delovanja sistema.

5 Vrednotenje sistema

Za generiranje prometnih poročil smo primerjali komercialni oblačni model GPT-5.1 in dve različici lokalnega modela GaMS3-12B-Instruct, prilagojenega za slovenščino (ena različica je bila učena na izvlečkih zgodovinskih poročil, druga pa na vhodno-izhodnih parih modela GPT). Strokovnjaki z RTV Slovenija so na podlagi evalvacijskega dokumenta z vzporednimi prikazi poročil ugotovili, da druga različica modela GaMS ustvarja jezikovno in slovnično najbolj kakovostna besedila. Kljub boljši jezikovni ustreznosti pa se je pri nadaljnjem delu lokalni model izkazal za manj prožnega. Zaradi strogega prilagajanja na učno množico je pri poskusih spreminjanja poziva (angl. *prompt*) občasno še vedno sledil prvotno naučenim vzorcem in s tem izgubil del splošnosti, medtem ko je bil model, dostopen prek programskega vmesnika OpenAI API, bistveno bolj prilagodljiv sprotnim spremembam pozivov.

Ko je aplikacijo v produkciji začelo uporabljati več uporabnikov (od 8. do 21. julija 2026 – to je najbolj reprezentativno obdobje, saj se je takrat začela intenzivna produkcijska uporaba), je povprečni dnevni strošek znašal približno 12,97 € na dan, kar ob mesečni ekstrapolaciji pomeni približno 389 € na mesec oziroma 4.670 € na leto. Za lokalno tvorjenje besedila smo že preizkusili model GaMS3-12B-Instruct (kvantiziran Q6_K), ki pri približno 6,56 bita na utež zavzame približno 10 GB pomnilnika za uteži [9]. Nekvantizirana različica bi predvidoma potrebovala približno 32 GB pomnilnika VRAM, kvantizirana pa pri skoraj enaki kakovosti besedila deluje na grafični kartici s 16 GB pomnilnika VRAM; v praksi uporabljamo kartico NVIDIA GeForce RTX 5070 Ti (16 GB, 300 W, priporočena maloprodajna cena približno 750 USD) [10]. Ob neprekinjenem 24-urnem delovanju in ceni električne energije za poslovne odjemalce približno 0,18 € na kWh (referenčna cena za negospodinjske odjemalce v prvem četrtletju 2026) [11], [12] bi tekoči strošek električne energije za kartico RTX 5070 Ti znašal približno 39 € na mesec oziroma 467 € na leto, kar je bistveno manj od mesečnega stroška klicev storitve OpenAI pri polni produkcijski uporabi. Lastna strojna oprema (približno 700–900 €) bi se ob trenutnem tempu rasti povrnila v manj kot dveh mesecih, pri čemer je treba upoštevati še dodaten čas in stroške vzdrževanja ter nadgradenj in odsotnost podpore, ki jo ponuja OpenAI.

Aplikacija trenutno omogoča lektorjem, da pri posameznem poročilu označijo besedilo kot ustrezno, uredijo generirano besedilo, označijo tip napake in dodajo prost komentar; ob ponovnem odprtju poročila se namesto izvirnega generiranega besedila prikaže zadnja lektorjeva popravljena različica, kar velja tudi za druge uporabnike, ki si isto poročilo ogledajo pozneje. Povezava med zbranimi povratnimi informacijami in dejanskim izboljševanjem poziva trenutno še ni samodejna, temveč zahteva ročni pregled in vključevanje ugotovitev v nadaljnji razvoj. V prihodnje bi bilo smiselno raziskati možnosti za delno ali popolnoma samodejno izvedbo tega procesa ter vzpostaviti sistematično povezavo med povratnimi informacijami uporabnikov in inženirstvom pozivov.

6 Zaključek

V prispevku smo predstavili zasnovo in implementacijo sistema za samodejno generiranje prometnih poročil za radijsko poročanje na RTV Slovenija. Sistem povezuje sprotne prometne podatke iz nacionalnega portala NAP, podatkovni cevovod za shranjevanje in normalizacijo zapisov, klasifikacijski model za zaznavanje pomembnih oziroma nujnih dogodkov ter dvostopenjsko generiranje besedila z velikimi jezikovnimi modeli. Pokazali smo, da kakovost končnega poročila ni odvisna samo od izbire jezikovnega modela, temveč predvsem od kakovosti, strukture, filtriranja in razvrščanja vhodnih podatkov.

Pomemben del razvoja je predstavljala povezava med zgodovinskimi in sprotnimi podatki. Zgodovinski prometni zapisi in arhivirana radijska poročila so omogočili razumevanje uredniškega procesa, pripravo evalvacijskih primerov in razvoj pristopov za klasifikacijo ter generiranje besedila. Hkrati se je pokazalo, da zgodovinskih podatkov ni mogoče uporabiti neposredno v celoti, saj morajo modeli v produkcijskem delovanju prejemati vhodne podatke, ki so primerljivi s podatki iz NAP dostopnih točk. Zato je bila posebna pozornost namenjena izboru tistih vsebinskih sklopov in atributov, ki jih je mogoče zanesljivo pridobiti tudi iz sprotnega vira.

Rešitev je bila preizkušena in se že uporablja tudi v dejanski uredniški praksi. Povratne informacije uredništva so pokazale, da sistem lahko uspešno podpira pripravo prometnih poročil, hkrati pa so razkrile pomen sprotnega prilagajanja pravil, pragov in pozivov jezikovnemu modelu. V resničnem obratovanju so se pokazali primeri, ki jih samo vrednotenje na zgodovinskih podatkih ne bi zadostno razkrilo, na primer obravnava vožnje v nasprotno smer, zastaranje nujnih obvestil, dvoumna krajevna imena, manjkajoči vzroki dogodkov in pragovi za vključevanje zastojev. Ti primeri potrjujejo, da mora biti sistem zasnovan kot nastavljiva in sledljiva uredniška podpora, ne kot popolnoma zaprt samodejni generator.

Primerjava med oblačnimi in lokalno prilagojenimi jezikovnimi modeli je pokazala tudi praktičen kompromis med kakovostjo besedila, stroški, odzivnostjo in neodvisnostjo od zunanjih ponudnikov. Oblačni modeli trenutno dosegajo boljšo slogovno kakovost, vendar njihova uporaba prinaša operativne stroške in odvisnost od zunanje storitve. Zato zbrani popravki, komentarji in uredniške oznake predstavljajo pomembno osnovo za nadaljnje prilagajanje lokalnega modela, s katerim bi bilo mogoče postopoma zmanjšati odvisnost od plačljivih oblačnih rešitev.

Na podlagi razvoja in pilotne uporabe lahko sklenemo, da je samodejno generiranje prometnih poročil izvedljivo in uporabno, kadar je podprto z zanesljivim podatkovnim cevovodom, deterministčnim filtriranjem, sledljivostjo do izvornih zapisov in možnostjo končne uredniške presoje. Sistem zato ni namenjen popolni zamenjavi urednika, temveč predvsem zmanjšanju ročnega dela, hitrejšemu zaznavanju pomembnih dogodkov in pripravi kakovostnega poročila. Nadaljnje delo vključuje dodatno prilagoditev lokalnega jezikovnega modela na zbranih produkcijskih podatkih, bolj sistematično vrednotenje kakovosti generiranih poročil ter nadgradnje uporabniškega vmesnika glede na potrebe poročanja.

Zahvala

Delo je nastalo v sodelovanju z RTV Slovenija ter v okviru projekta European Union GA 101198470 »Digital Europe Programme: LLMs4EU« in projekta Slovenian Artificial Intelligence Factory (SLAIF, 101254461), ki ga sofinancira Evropska unija (HORIZON-JU-EUROHPC-2025-AI-01-IBA). Izražena stališča in mnenja so izključno stališča avtorjev in ne odražajo nujno stališč Evropske unije ali EuroHPC JU.

Literatura

[1] https://nap.si/sl, Nacionalna točka dostopa, obiskano 21. 5. 2026.

[2] MEIR Nicole, »Automated earnings stories multiply«, *The Associated Press*, 15. 6. 2015, https://www.ap.org/the-definitive-source/announcements/automated-earnings-stories-multiply/, obiskano 4. 6. 2026.

[3] THE WASHINGTON POST, »The Washington Post experiments with automated storytelling to help power 2016 Rio Olympics coverage«, *The Washington Post*, 5. 8. 2016, https://www.washingtonpost.com/pr/wp/2016/08/05/the-washington-post-experiments-with-automated-storytelling-to-help-power-2016-rio-olympics-coverage/, obiskano 4. 6. 2026.

[4] LOS ANGELES TIMES, »What is the Quakebot and how does it work?«, *Los Angeles Times*, https://www.latimes.com/la-me-quakebot-faq-20190517-story.html, obiskano 4. 6. 2026.

[5] LIU Xiaomo, NOURBAKHSH Armineh, LI Quanzhi, SHAH Sameena, MARTIN Robert, DUPREY John, »Reuters Tracer: Toward Automated News Production Using Large Scale Social Media Data«, *2017 IEEE International Conference on Big Data*, IEEE, 2017, str. 1483–1493, doi: https://doi.org/10.1109/BigData.2017.8258082.

[6] REUTERS COMMUNICATIONS, »How AI helps power trusted news at Reuters«, *Reuters*, 12. 7. 2023, https://www.reuters.com/media-center/how-ai-helps-power-trusted-news-at-reuters/, obiskano 4. 6. 2026.

[7] FINTRAFFIC, »Bauer Media to start broadcasting targeted traffic bulletins produced locally using artificial intelligence on the radio«, *Fintraffic*, 8. 11. 2023, zadnjič spremenjeno 9. 1. 2026, https://www.fintraffic.fi/en/news/bauer-media-start-broadcasting-targeted-traffic-bulletins-produced-locally-using-artificial, obiskano 4. 6. 2026.

[8] INRIX, »INRIX Launches AI-Powered Traffic Reports for TV and Radio – Fully Replacing Human Broadcasters«, *INRIX*, 18. 11. 2025, https://inrix.com/press-releases/inrix-launches-ai-powered-traffic-reports-for-tv-and-radio-fully-replacing-human-broadcasters/, obiskano 4. 6. 2026.

[9] GGML-ORG, »llama.cpp – tools/quantize README (K-quants)«, GitHub, https://github.com/ggml-org/llama.cpp/blob/master/tools/quantize/README.md, obiskano 23. 7. 2026.

[10] NVIDIA, »GeForce RTX 5070 Ti«, NVIDIA, https://www.nvidia.com/en-us/geforce/graphics-cards/50-series/rtx-5070-family/, obiskano 23. 7. 2026.

[11] ENERGETIKA-PORTAL.SI, »Cene električne energije v prvem četrtletju 2026«, Energetika-portal.si, https://www.energetika-portal.si/nc/novica/n/cene-elektricne-energije-v-prvem-cetrtletju-2026/, obiskano 23. 7. 2026.

[12] GEN-I, »Nove cene tarif električne energije za male poslovne odjemalce«, GEN-I, https://gen-i.si/novice/nove-cene-tarif-elektricne-energije-za-male-poslovne-odjemalce/, obiskano 23. 7. 2026.

[13] RADIOTODAY, »AI-powered traffic reports have arrived at Bauer«, RadioToday, 14. 11. 2025, https://radiotoday.co.uk/2025/11/ai-powered-traffic-reports-have-arrived-at-bauer/, obiskano 23. 7. 2026.

[14] SALTON Gerard, BUCKLEY Christopher, »Term-weighting approaches in automatic text retrieval«, *Information Processing & Management*, let. 24, št. 5, 1988, str. 513–523, doi: https://doi.org/10.1016/0306-4573(88)90021-0.

[15] SALTON Gerard, WONG Anita, YANG Chung-Shu, »A Vector Space Model for Automatic Indexing«, *Communications of the ACM*, let. 18, št. 11, 1975, str. 613–620, doi: https://doi.org/10.1145/361219.361220.

[16] DOCKER, »What is Docker?«, *Docker Docs*, https://docs.docker.com/get-started/docker-overview/, obiskano 26. 7. 2026.

[17] TRAEFIK LABS, »Configuration Introduction«, *Traefik Documentation*, https://doc.traefik.io/traefik/getting-started/configuration-overview/, obiskano 26. 7. 2026.

[18] INTERNET SECURITY RESEARCH GROUP, »Documentation«, *Let's Encrypt*, https://letsencrypt.org/docs/, obiskano 26. 7. 2026.

[19] PROMETHEUS AUTHORS, »Overview«, *Prometheus Documentation*, https://prometheus.io/docs/introduction/overview/, obiskano 26. 7. 2026.

[20] GRAFANA LABS, »About Grafana«, *Grafana Documentation*, https://grafana.com/docs/grafana/latest/introduction/, obiskano 26. 7. 2026.

[21] PALLETS PROJECTS, »Welcome to Flask«, *Flask Documentation*, https://flask.palletsprojects.com/en/stable/, obiskano 26. 7. 2026.

[22] REDIS, »Redis Documentation«, *Redis Docs*, https://redis.io/docs/latest/, obiskano 26. 7. 2026.

[23] RAMÍREZ Sebastián, »FastAPI«, *FastAPI Documentation*, https://fastapi.tiangolo.com/, obiskano 26. 7. 2026.

[24] WARNER Benjamin et al., »Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference«, arXiv:2412.13663, 2024, https://arxiv.org/abs/2412.13663.

[25] PASZKE Adam et al., »PyTorch: An Imperative Style, High-Performance Deep Learning Library«, *Advances in Neural Information Processing Systems 32*, 2019, str. 8024–8035, https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[26] OPENAI, »OpenAI API Documentation«, https://developers.openai.com/api/docs/, obiskano 26. 7. 2026.

[27] GOOGLE, »What is Angular?«, *Angular Documentation*, https://angular.dev/overview, obiskano 26. 7. 2026.

[28] THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, »PostgreSQL 17 Documentation«, https://www.postgresql.org/docs/17/, obiskano 26. 7. 2026.

[29] BAYER Mike, »Alembic Documentation«, https://alembic.sqlalchemy.org/en/latest/, obiskano 26. 7. 2026.