Rešenje

Organizacija rada

Problem	Teorijske osnove	Rešenje	Evaluacija rešenja	Rezultati/Diskusija

Cilj rešenja

- Pojednostavljeni slučajevi korišćenja, bez definisanja koraka procesa
  - Koje potrebe tvoj softver ispunjava?
  - Kakav ulaz se očekuje, kakav izlaz se dobija?

Opis strukture rešenja

- *Pipeline* u ML-u
- Moduli (*feature* u kontekstu *package by feature* pristupa paketiranja) u SW inženjerstvu

Detalji svake celine u strukturi rešenja

- Rad prelazi na detalje svake celine sistema tek nakon opisa cilja i strukture sistema

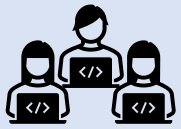
Rešenje

Cilj rešenja

III. REŠENJE

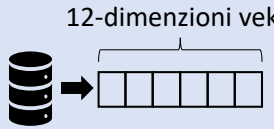
Slika 1 prikazuje postupak kojim smo analizirali kako redovnost angažovanja na pripremnim aktivnostima za čas utiče na uspeh učenika na završnom ispitu.

Opis strukture rešenja



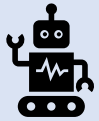
Interakcija 110 studenata sa platformom za učenje; uspeh na ispitu

Prikupljanje podataka



Procena ponašanja koja ukazuju na redovnost studentovog angažovanja

Ekstrakcija obeležja



Treniranje regresionih modela i analiza važnosti obeležja

Predikcija uspeha na završnom ispitu

Analiza važnosti obeležja

Utvrđivanje faktora značajnih za uspeh na završnom ispitu

Detalji svake celine u strukturi rešenja

A. Prikupljanje podataka

B. Ekstrakcija obeležja

C. Utvrđivanje faktora značajnih za uspeh na ispitu

Nivo detalja

- Nemoguće je opisati sve detalje sistema
  - Opišite ga na apstraktnom nivou – nemojte prikazivati isečke koda
  - Posvetite pažnju bitnim funkcionalnostima

Ne mešajte rešenje sa teorijskim osnovama

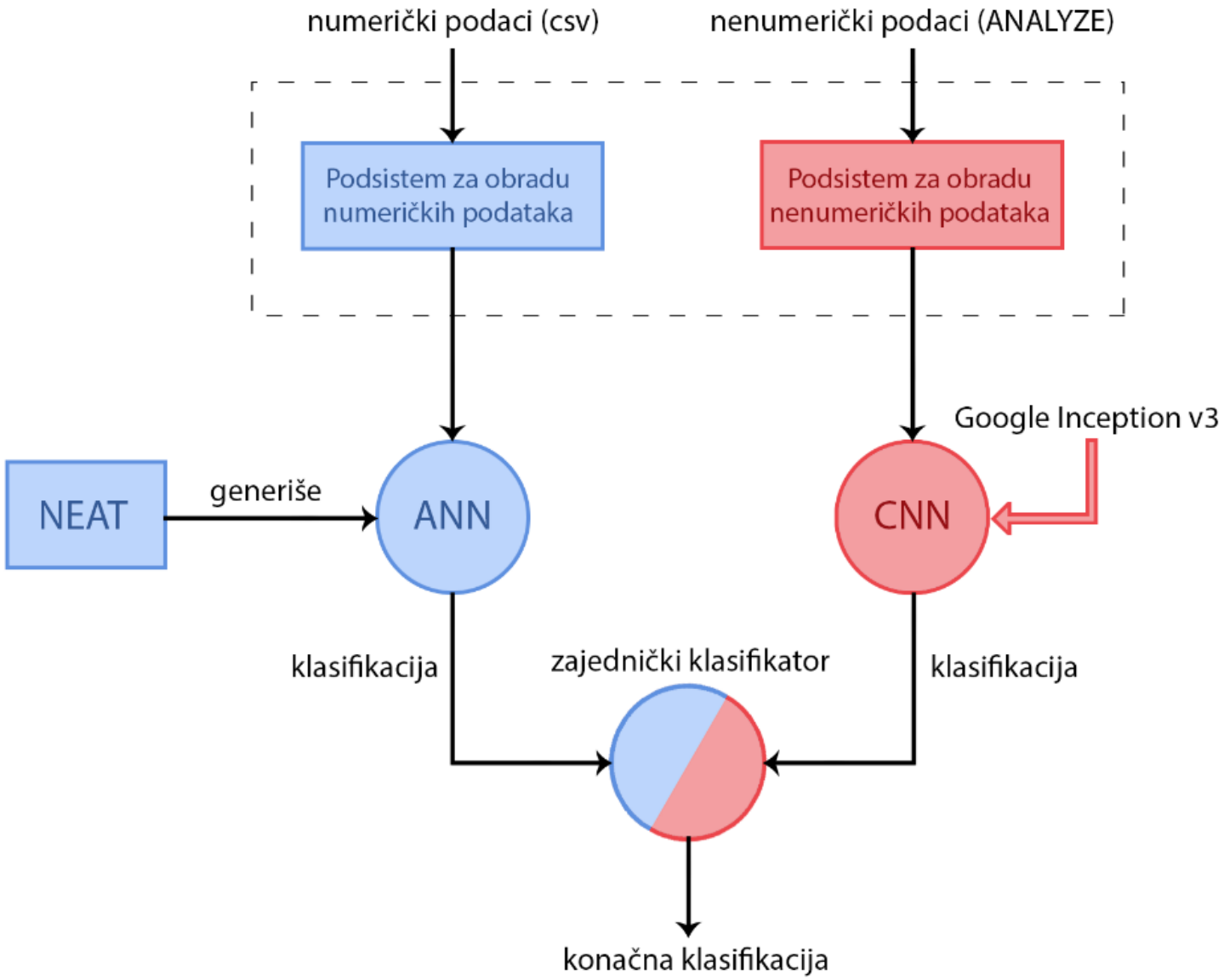
- Ne opisujete metode koje niste vi osmislili (ali ih primenjujete kao deo svog rešenja)

- Na primer, predstavljate CNN za klasifikaciju slika
  - Ne opisujete osnove CNN
    - generalna arhitektura, slojevi, kako se CNN trenira,…
  - Opišite kako ste vi primenili CNN za rešavanje problema
    - Koje ste podatke koristili, kako ste ih pretprocesirali, augmentacija skupa podataka,…
    - Koju **konkretnu** arhitekturu CNN ste koristili (i zašto tu)
      - Pomoću koje funkcije gubitka se optimizuje model
      - Koji optimizacioni algoritam koristite
    - Koje ste kombinacije model/pretprocesiranje isprobali

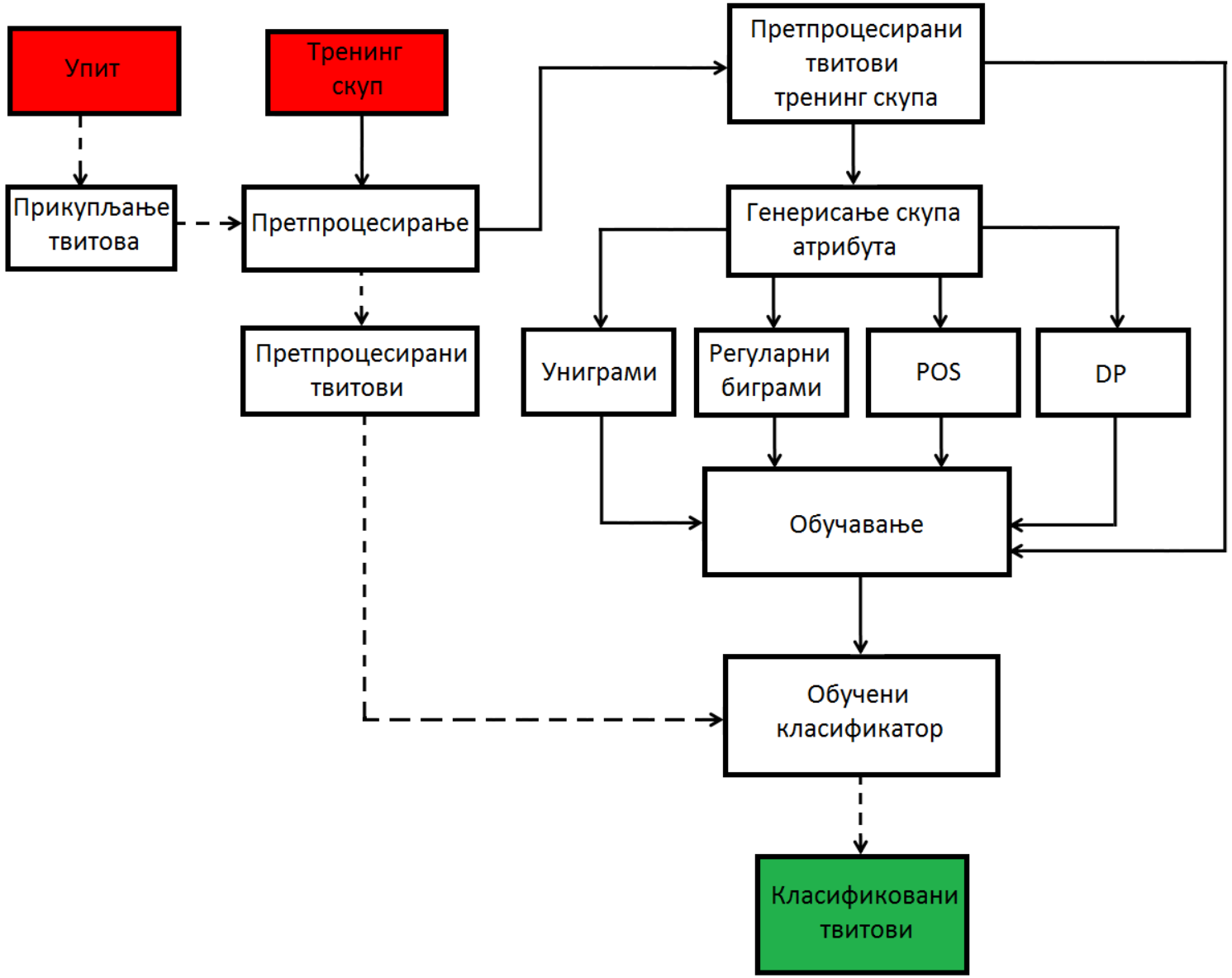
Skup podataka

- Da li je skup podataka javan
- Kako su podaci prikupljani i anotirani
- Koje klase postoje i koliki je njihov udeo
- Broj instanci
- Eventualni problemi koje treba rešiti
  - Duplikati, nedostajuće vrednosti, šum/loš kvalitet, varijacija u jednoj kategoriji, sličnost primera različitih kategorija,…

Primer opisa strukture rešenja



Primer opisa strukture rešenja



Primer opisa strukture rešenja

Ulazna slika (A) se prosleđuje

U-Net modelu (B)

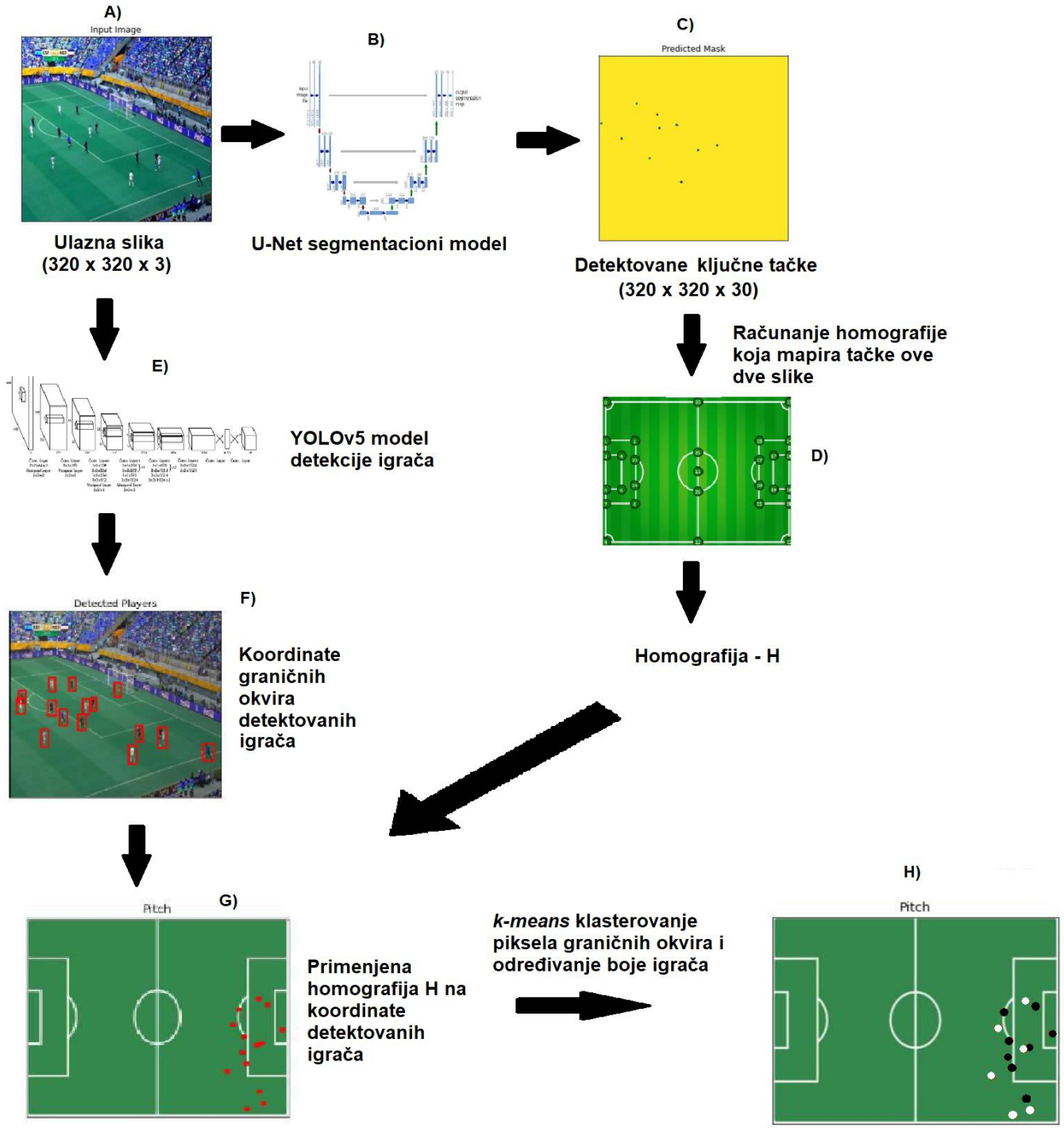
koji za izlaz daje matricu koja predstavlja detektovane tačke koje su vizualizovane na slici (C).

Mapiranjem tačaka na njihove ciljne koordinate sa slike (D), dobijamo homografiju H.

Na sliku (A) se primenjuje YOLO (E) model za detekciju igrača (F).

Primenom homografije na igrače dobijamo sliku (G).

K-means klasterovanjem piksela dobijamo boje igrača za konačni prikaz (H).



Primeri grešaka

- Redosled izlaganja:
  - Opis kako se prva faza evaluira
  - Opis kako se druga faza evaluira
  - Opis šta treba uraditi pre prve i druge faze (pretprocesiranje)
  - Opis šta je zaključeno u pretprocesiranju i kako je to rešeno

I. KREIRANJE *CHATBOT-A* KORIŠĆENJEM WORD2VEC BIBLIOTEKE I REKURENTNIH NEURONSKIH MREŽA

Naš pristup razvoju *chatbot* aplikacije se može podeliti u dve faze. Prvu fazu koja zahteva brz odziv (*high recall*) i drugu fazu visoke preciznosti (*high precision*). Pre prve faze smo odradili predprocesiranje našeg skupa podataka u kom smo došli do zaključka da postoji dosta duplih pitanja i istih pitanja sa različitim odgovorima. Dupla pitanja smo obrisali, a pitanja sa različitim odgovorima smo konkatenirali u jedan jedinstven odgovor.

- Pitanja za autora:
  - Šta je cilj prve faze
  - Šta je cilj druge faze
  - Kako izgleda skup podataka?

Primeri grešaka

- Bilo bi bolje da naslov poglavlja izražava cilj faze
- Opis postupka, bez da nam je jasno šta je cilj postupka

A. Faza 1: High recall

Prvi korak u razvoju našeg modela bio je vektorizacija reči, odnosno transformacija reči u numeričke vektore. Za ovu svrhu smo isprobali različite metode. Prvo smo probali *CountVectorizer* koji nam transformiše reči u matricu ponavljanja. Model je dosta jednostavan za naš problem nije davao dobre rezultate jer ne uzima u obzir semantiku reči. Drugo stvar koju smo probali je *TFIDF Vectorizer* on je naprednije od prvog pristupa jer pored broja ponavljanja reči uzima u obzir i važnost reči u našem dokumentu. Kao što smo naglasili ni jedan od ova dva pristupa se nije dobro pokazao kod nas, verovatno jer oba pristupa koriste *bag-of-words*, koji ignoriše redosled reči u tekstu.

Primeri grešaka

- Redosled:
  - 3 skupa podataka – 1. objašnjen pre potpoglavlja, druga dva se sada definišu
- Mešanje „Teorijskih osnova“ i „Rešenja“:
  - Opis šta su lematizacija i steming

Zbog gore navedenih nedostataka prva dva pristupa, odlučili smo se za Word2Vec, koji uzima u obzir kontekst između dve reči. Word2Vec smo isprobali smo na 3 različita skupa podataka. Prvi je bio naš osnovni koji smo sredili pre ulaska u fazu 1, dok drugi i treći smo dobili lematizacijom i shematizacijom. Lematizacija je pretvaranje reči u njihov osnovni oblik, dok je stematizacija proces koji vraća reč na njen korenski oblik. Ni jedan od ova dva napredna pristupa nije doneo poboljšanje u rezultatima, tako da smo odlučili da radimo sa prvobitnim skupom podataka.

Primeri grešaka

- Tek u ovom pasusu saznajemo cilj faze 1
  - Mada, zašto baš 100?
- Ostalo je nejasno šta je ulaz
  - Da li je u pitanju rečenica?
  - Ako jeste, kako se od nje formira vektor?
  - Koje je dužine taj vektor (i kako je ta dužina određena)?
- Dijagram je nejasan

Kada smo završili vektorizaciju, isprobali smo 3 različite metrike sličnosti da bi pronašli slična pitanja. Isprobali smo *Manhattan* razdaljinu, kosinusnu razdaljinu i Euklidsku razdaljinu. Nakon što smo istestirali sva 3 pristupa, došli smo do zaključka da najmanju grešku pravi kosinusna razdaljina i odlučili smo nju da koristimo (Slika 2). Kao krajnji rezultat faze 1 dobijamo 100 pitanja koja su najsličnija unosu korisnika. Ovim smo postigli da kad krenemo u drugu fazu, naša rekurentna neuronska mreža neće prolaziti kroz ceo skup podataka, nego samo kroz ovih 100 najsličnijih pitanja.

Primeri grešaka

- Sadržaj koji ne pripada poglavlju *Faza 2: High precision*

Poslednji korak razvijanje naše *chatbot* aplikacije. Odlučili smo da napravimo *web* aplikaciju kako bi mogli lakše da testiramo naš *chatbot*. Serversku stranu naše aplikacije smo napravili koristeći *Flask framework*, dok smo klijentsku stranu naše *web* aplikacije napravili koristeći *React JS* biblioteku.

- Sadržaj koji ne pripada poglavlju „Rešenje“

C. Alternativni pristupi

Takođe smo razmatrali i druge načine za rešavanje ovog problema. Jedan od njih je bio korišćenje Transformer modela, kao sto jer *BERT* (*Bidirectional Encoder Representations from Transformers*). Ovakvi modeli su se pokazali dosta efikasnim u obradi prirodnog jezika, međutim ovo su veoma kompleksni modeli koji zahtevaju velike količine podataka i računarske snage koja je potrebna za treniranje.