Univerzitet u Nišu, Elektronski fakultet
Katedra za računarstvo

Chatbot za generisanje web stranica
Izveštaj projekta

Predmet: Duboko učenje

Studenti: Nikola Stevanović 1281
Danilo Veljović 1120

Profesor: prof. dr Aleksandar Milosavljević

Niš, 2021

Sadržaj

1 O problemu

Ovaj projekat se bavi implementacijom prototipa četbot aplikacije koji dinamički generiše veb stranu, konkretno njene HTML elemente. Ideja projekta je da korisnik može da generiše web stranu sa proizvoljnim sadržajem tako što će četovati sa četbotom. Četbot bi trebalo da nudi I mogućnost ubacivanja osnovnih I složenijih (kombinacija osnovnih HTML) elemenata.

Trebalo bi da korisnik takođe može da elemente ubacuje na proizvoljna mesta I da im dodaje proizvoljan sadržaj, bilo da je to nekakav tekst ili neki drugi HTML element. Pošto se neki sadržaj može generisati dinamički, četbot može da ima I opciju da postavi pitanje kada nema dovoljno informacija u vezi nekog elementa.

Generisanje veb stranice je složen problem koji u sebi ima više potproblema. Jedan od problema je samo prepoznavanje elemenata koje korisnik želi da generiše, što može biti naročito složeno ako korisnik navede više elemenata u istoj rečenici.

Sledeći problem je detektovanje prostornih odrednica kao što su before, after, inside itd, kojima korisnik naglašava gde hoće da se određeni element nalazi. Takođe jedan od potproblema može biti, ako npr korisnik želi da generiše tabelu. Logično je da se postavi pitanje koliko veliku tabelu korisnik želi, a ne da se generiše tabela fiksne veličine. Sledeći potproblem je generisanje same veb strane posle dodavanja elemenata I prikaz tih elemenata.

2 Metode rešavanja problema

Kada se pristupalo rešavanju ovog problema odlučeno je da se koristi rule-based approach sa dubokim učenjem, odnosno da se napravi nekakva kombinacija između dva pristupa, kojim bi se dobili dobri rezultati čak I za male skupove podataka, kojima se baratalo prilikom razvoja ovog prototipa. Obično su mali skupovi podataka poprilično izazovni za rekurentne neuronske mreže, koje su bile alternativni pravac u rešavanju ovog problema, I iz tog razloga smo se odlučili za jedan ovakav pristup.

Ideja je bila da se koriste dva modela feed-forward mreža I bag-of-words. Pre toga je važno pročistiti ulaz I tokenizovati rečenicu. Nakon tokenizacije ulaza, I odbacivanja znakova interpukcije, bag-of-words pristupom se detektuju svi relevantni elementi koji su detektovani, zatim se pomoću dve mreže vraćaju odgovori odnosno klase koje su prepoznate. Na osnovu prostornih odrednica se zatim ti detektovani elementi ubacuju u postojeći sadržaj I sve se renderuje na real time prikaz.

2.1 Feed-forward mreže I bag of words

U aplikaciji se koriste dva modela feed-forward mreža. Jedan model se koristi za nalaženje odgovora na određene konstrukcije rečenične. Npr kada korisnik unese *I want a div*, potrebno je da se feed-forward modelom prepozna taj pattern I da se vrati odgovor na taj zahtev. Tako da odgovor na ovaj korisnikov unos bude "<BEFORE> <CONTENT><INSIDE> <AFTER>"

Nešto kasnije će biti reči o skupu podataka koji je korišćen I o njegovom formatiranju. Sama mreža se sastoji iz:

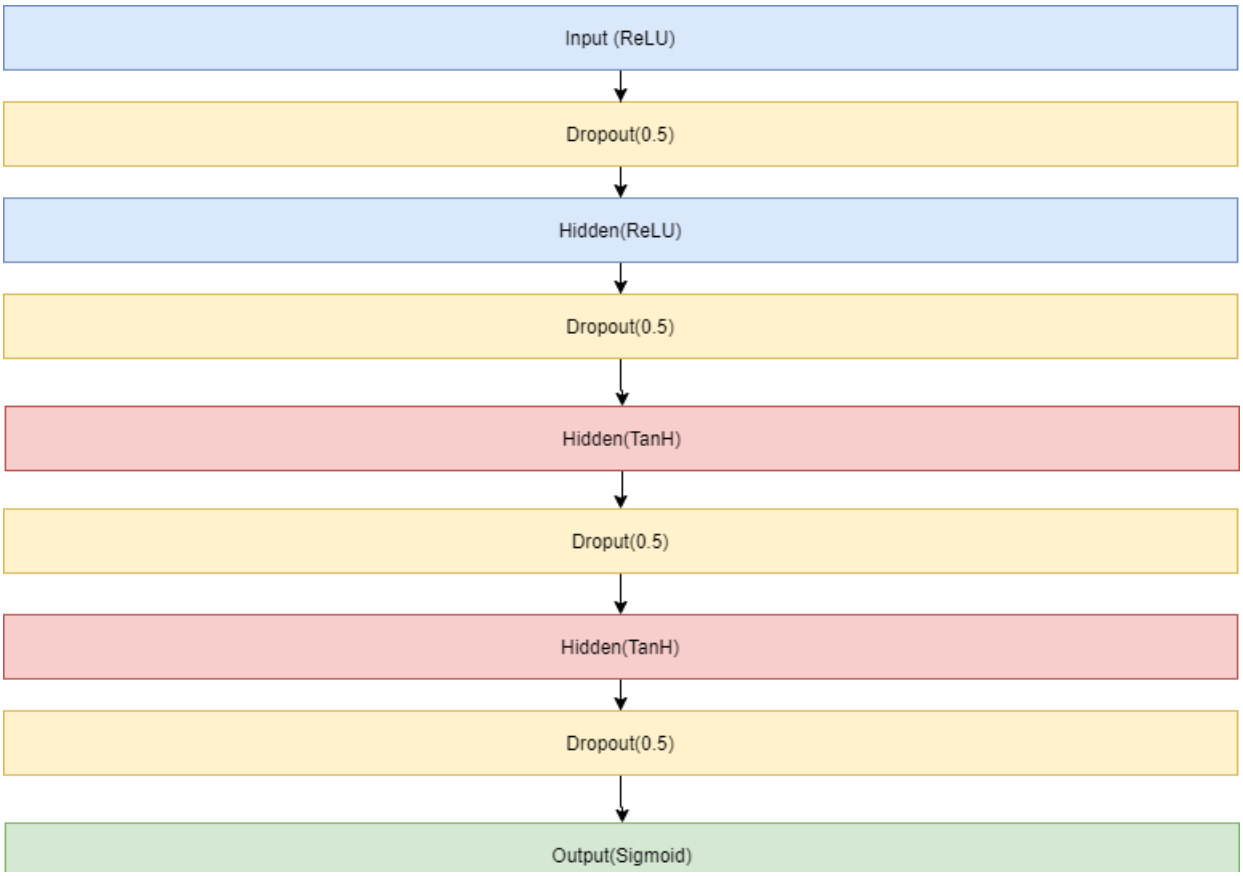
- Ulaznog sloja koji kao aktivacionu funkciju ima relu
- Jednog skrivenog sloja koji ima takodje relu aktivacionu funkciju
- Dva skrivena sloja koji imaju hiperbolicki tangens kao aktivacionu funkciju
- Izlazni sloj koji ima sigmoidnu funkciju kao aktivacionu

Između svakog od ovih slojeva se nalazi po jedan Dropout regularizacioni sloj koji ima vrednost 0.5.

Izlazni sloj ima sigmoidnu funkciju kao aktivacionu jer je potrebno da se radi multilabel klasifikacija, tj da se više elemenata prepozna u istoj rečenici. Primera radi, ako korisnik unese *I want a div and a header*, mreža mora da vrati HTML elemente za obe ove reči.

Ostale aktivacione funkcije su dobijene eksperimentalno. Obično nije praksa da se unutar mreže meštaju aktivacione funkcije, međutim u ovom slučaju to daje najbolje rezultate.

Mreža se trenira tako što se najpre bag-of-words pristupom detektuju sve relevantne reči (nazivi elemenata ili prostorne odrednice) I kreira se vektor od rečenice koja je prosleđena I svih mogućih relevantnih reči.

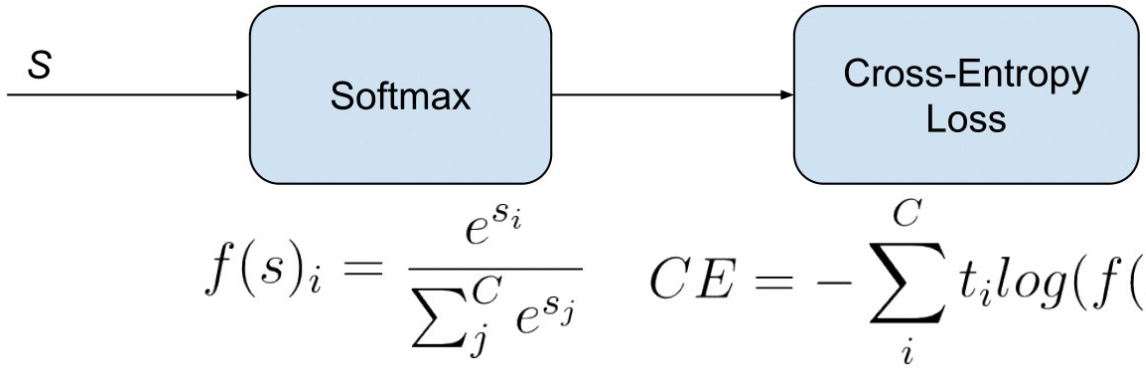


Tako formiran vektor se prosleđuje u mrežu I time se trenira.

Na slici 1 je dat prikaz feed forward modela koji je korišćen

2.2 Loss funkcije I metrika

Loss funkcija koja je korišćena je ***categorical cross entropy***. Ova loss funkcija se vrlo često koristi kod multilabel klasifikacije. To je Softmax aktivacija plus Cross-Entropy loss funkcija. Ovim se mreža trenira da vrati verovatnoće za N klasa po uzorku. Na slici 2 je dat prikaz categorical cross entropy loss funkcije.



Slika 2: Categorical cross entropy

Optmizaciona funkcija koja je korišćena je stohastic gradient descent, sa learning rate-om 0.01.

Metrika koja je praćena u toku treninga je accuracy, tj tačnost. Pratili smo koliko je u svakoj od epoha bila tačnost. U toku treniranja taj broj je varirao od 90-100%.

Model je bio treniran u 200 epoha, a bečevi kojima je treniran su bili veličine 5.

2.3 Tehnologije

Projekat je implementiran u Python programskom jeziku. Korišćena je biblioteka Tensorflow za duboko učenje. Za svođenje reči na njen koreni oblik korišćena je WordNetLemmatizer biblioteka. Za prikaz HTML koda na ekran korišćena je BeautifulSoup kao I cv2.

Projekat je implementiran kroz tri skripte. Prva skripta *training.py* u kojoj se nalazi model neuronske mreže koji se koristi za prepoznavanje/retrieval odgovarajućih odgovora. Njoj se kroz argument komandne linije prosleđuje putanja do trening skupa podataka, gde da sačuva sve korene reči koji se javljaju u tekstu, gde da sačuva sve klase koje treba da prepozna, gde da sačuva finalni istrenirani model.

U slučaju kad se trenira model za prepoznavanje prostornih odrednica, prosleđuje se putanja do **location.json** fajla gde se nalazi trening skup podataka.

U slučaju kad se trenira model za HTML elemenata, prosleđuje se putanja do **elements.json** fajla gde se nalazi trening skup podataka.

Sve reči iz trening skupa se stavljaju u pickle fajl-ove, kao I sve klase koje modeli treba da prepoznaju. Fajl gde se nalaze reči za prostorne odrednice je **loc_words.pkl**, a fajl gde se nalaze klase se **loc_classes.pkl**.

Fajl gde se nalaze reči za HTML elemente je **el_words.pkl**, a fajl gde se nalaze klase se el**_classes.pkl**.

Istrenirani model se čuva u fajlu čije se ime takođe prosleđuje kroz komandnu liniju. U slučaju modela za prepoznavanje prostornih odrednica to je fajl **loc_chatbotmodel.h5**. U slučaju modela za generisanje HTML elemenata to je fajl **el_chatbotmodel.h5**.

Skripta **chatbot.py** ima logiku za četovanje, generisanje I insertovanje elemenata na pravo mesto. Nema nikakve argumente komandne linije, ona samo učitava sve modele koji su sačuvani, klase I reči koje su korišćene. Najpre se tu prečisti ulaz, nađu se koreni reči, vidi se koji su elementi korišćeni u rečenici, nalaze se njihovi HTML kodovi, vidi se koji se presotorne odrednice koriste u rečenici, I na kraju se na osnovu svega toga vrši insertovanje elemenata u postojeći HTML kod I to se prikazuje pomoću BeautifulSoup biblioteke. Na slici 3 je dat prikaz rezultata kako izgledaju generisani elementi.

Skripta tableGenerator.py služi da pomogne kod dinamičkog generisanja tabele. Naime kada se prepozna da treba da se generiše tabela, postavljaju se pitanja da se odredi veličina tabele koja treba da se generiše I ova skripta onda pomaže kod generisanja te tabele.

Implementacija I kod su dostupni na github repozitorijumu(https://github.com/DaniloVeljovic/chatbot)

SLIKA NA KOJOJ SE VIDI KAKO IZGLEDA NEKI GENERISAN KOD

3 Skup podataka

Skup podataka koji je korišćen su sami autori generisali. Podaci su u json formatu. Na slici 4 je dat prikaz skupa podataka koji se koristi za treniranje modela za prepoznavanje prostornih odrednica.

```
{
 "intents": [
   {
    "tag": "before",
    "patterns": [
     "before",
     "prior to",
     "previous to",
     "ahead of",
     "in front of"
    ]
   },
   {
    "tag":"after",
    "patterns": [
     "after",
     "behind",
     "next to",
     "posterior to",
     "following",
     "beside"
    ]
   },
   {
    "tag": "inside",
    "patterns": [
     "inside",
     "in"
    ]
   }
 ]
}
```

Slika 4 – skup podataka za trening modela za detektovanje prostornih odrednica

Na slici 4 se jasno vidi format trening podataka. Tako se slično I učitava u training.py skripti. Učitava se lista intents-ova, svaki tag predstavlja klasu koju treba detektovati. U patterns delu se nalaze mogući sinonimi ili rečenice kako korisnik može da traži tu klasu/element. Npr ako korisnik negde u rečenici iskoristi reč before, ili prior to, model treba da detektuje klasu before koja odogovara tagu. U ovom slučaju nisu važni odgovori za tu klasu, samo je važna njena ispravna detekcija.

Na slici 5 je dat isecak za trening skup za


```
{
 "intents": [
{
 "tag": "grid",
 "patterns": [
  "i need a grid",
  "put a grid",
  "give me a grid",
  "add a grid"
 ],
 "responses": [
    "<BEFORE><div class=\"container\" id = ?>\n  <div class=\"row\" id = ?>\n    <div
class=\"col-sm-3\" id = ?> <CONTENT><INSIDE> </div>\n    <div class=\"col-sm-6\" id = ?>
<CONTENT><INSIDE> </div>\n    <div class=\"col-sm-3\" id = ?> <CONTENT><INSIDE>
</div>\n  </div>\n  <div class=\"row\" id = ?>\n    <div class=\"col-sm-3\" id = ?>
<CONTENT><INSIDE> </div>\n    <div class=\"col-sm-6\" id = ?> <CONTENT><INSIDE>
</div>\n    <div class=\"col-sm-3\" id = ?> <CONTENT><INSIDE> </div>\n  </div>\n  <div
class=\"row\" id = ?>\n    <div class=\"col-sm-3\" id = ?> <CONTENT><INSIDE> </div>\n    <div
class=\"col-sm-6\" id = ?> <CONTENT><INSIDE> </div>\n    <div class=\"col-sm-3\" id = ?>
<CONTENT><INSIDE> </div>\n  </div>\n</div><AFTER>"
 ]
},
   {
    "tag": "div",
    "patterns": [
     "i need a div",
     "put a div",
     "give me a div",
     "add a div"
    ],
    "responses": [
     "<BEFORE><div id = ?> <CONTENT><INSIDE> </div><AFTER>"
    ]
   },
   {
    "tag": "span",
    "patterns": [
     "i need a span",
     "put a span",
     "give me a span",
     "add a span"
    ],
    "responses": [
     "<BEFORE><span id = ?> <CONTENT><INSIDE> </span><AFTER>"
    ]
   },
```


Slika 5 – skup podataka za trening modela za detektovanje prostornih odrednica

Na slici 5 se vidi da se trening podaci nalaze u intents nizu. Vrednost tag elementa je vrednost klase koja je označena. Patterns ukazuje na sve moguće patterne kojim se može naglasiti da se radi o toj klasi. Jedina razlika u odnosu na trening skup vezan za detektovanje prostornih odrednica je postojanje responses polja. Ovaj niz sadrži sve moguće odgovore koji treba da se generišu kada je ta klasa prepoznata.

Primera radi, ako je detektovana klasa div potrebno je generisati "<BEFORE> <CONTENT><INSIDE> <AFTER>".

Ono što je takođe jedna novina što responses polje poseduje I specifične tagove poput <AFTER>, <BEFORE>, <CONTENT>, <INSIDE>.

Ovi tagovi kasnije pomažu kada se elementi insertuju u odnosu na ovaj element da se odredi tačno mesto tog novog elementa. Npr ako ovaj div ubacimo negde u HTML kod, on će biti ubačen sa ovim tagovima. Nakon toga ako hoćemo da neki element ubacimo pre ovog diva, potrebno je da se iskoristi ključna reč BEFORE I da se navede id ovog diva I na osnovu ovih tagova koje div ima, novi element će biti ubačen.

Ključna reč after funkcioniše na sličan način. Ključne reči CONTENT I INSIDE se jedino razlikuju iako deluju slično. Inside tag služi da se jedan element ubaci unutar drugog elementa. Content služi da se jednom elementu doda određeni tekstualni sadržaj.

Kada se renderuje web strana, ovi prostorni tagovi se brišu.

Takođe vidi se da pored jednostavnijih elemenata kao što su div I span, aplikacija može da generiše I nešto složenije elemente poput ovog grid elementa prikazanog na slici 4.

4 Rezultati

Neke od metrika tokom treniranja modela date su na slici 6.

```
Epoch 190/200
7/7 [==============================] - 0s 1ms/step - loss: 0.1554 -
accuracy: 0.9684 - precision: 0.2423 - recall: 1.0000 - auc: 0.9894
Epoch 191/200
7/7 [==============================] - 0s 1ms/step - loss: 0.2830 -
accuracy: 0.9440 - precision: 0.2609 - recall: 1.0000 - auc: 0.9790
Epoch 192/200
7/7 [==============================] - 0s 1ms/step - loss: 0.0238 -
accuracy: 1.0000 - precision: 0.2404 - recall: 1.0000 - auc: 0.9909
Epoch 193/200
7/7 [==============================] - 0s 1ms/step - loss: 0.0723 -
accuracy: 0.9768 - precision: 0.2442 - recall: 1.0000 - auc: 0.9967
Epoch 194/200
7/7 [==============================] - 0s 1ms/step - loss: 0.1657 -
accuracy: 0.9880 - precision: 0.2647 - recall: 0.9880 - auc: 0.9897
Epoch 195/200
7/7 [==============================] - 0s 1ms/step - loss: 0.1146 -
accuracy: 0.9452 - precision: 0.2596 - recall: 1.0000 - auc: 0.9846
Epoch 196/200
7/7 [==============================] - 0s 1ms/step - loss: 0.0365 -
accuracy: 1.0000 - precision: 0.2582 - recall: 1.0000 - auc: 0.9958
Epoch 197/200
7/7 [==============================] - 0s 1ms/step - loss: 0.2421 -
accuracy: 0.9768 - precision: 0.2554 - recall: 1.0000 - auc: 0.9811
Epoch 198/200
7/7 [==============================] - 0s 1ms/step - loss: 0.2258 -
accuracy: 0.8704 - precision: 0.2592 - recall: 1.0000 - auc: 0.9873
Epoch 199/200
7/7 [==============================] - 0s 2ms/step - loss: 0.0739 -
accuracy: 0.9802 - precision: 0.2567 - recall: 1.0000 - auc: 0.9898
Epoch 200/200
7/7 [==============================] - 0s 1ms/step - loss: 0.0836 -
accuracy: 0.9922 - precision: 0.2699 - recall: 1.0000 - auc: 0.9876
```

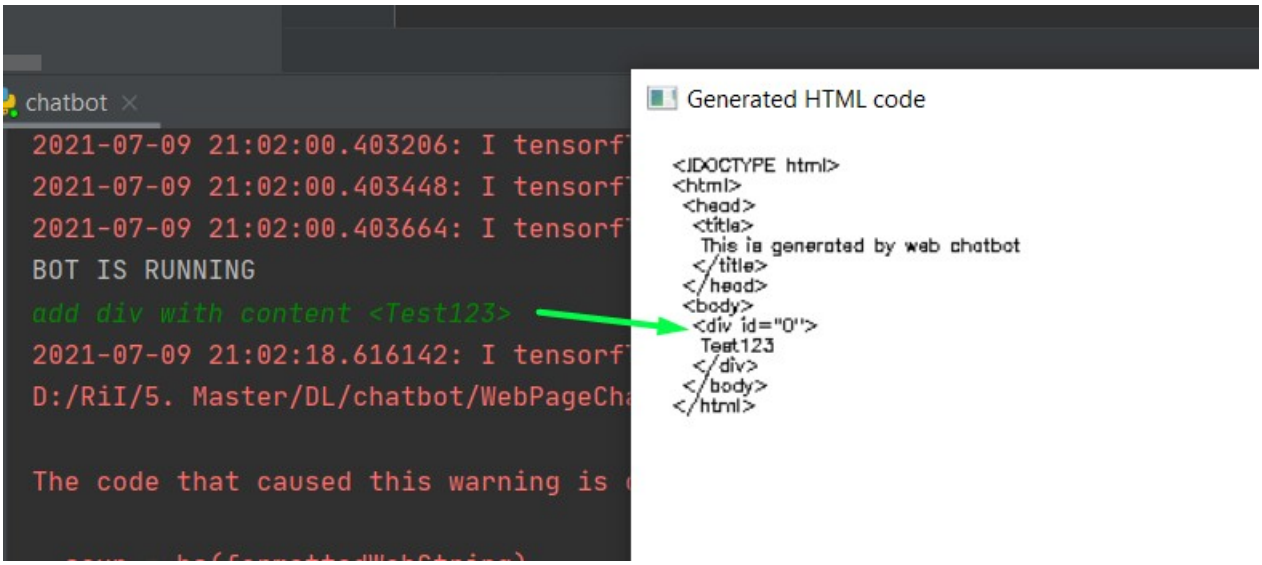
Slika 6 – Metrike tokom treniranja modela

Nakon treninga modela za prepoznavanje HTML elemenata neke metrike su date na slici 7.

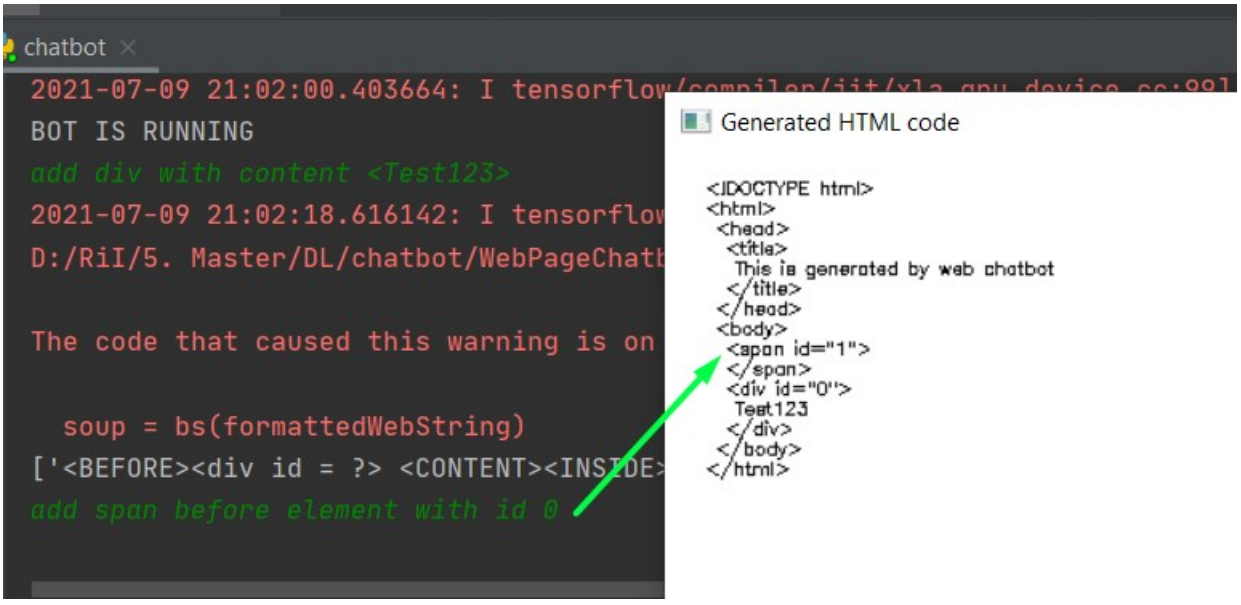
```
Evaluate on test data
1/1 [==============================] - 0s 422ms/step - loss: 2.9896e-04 - accuracy: 1.0000 - precision: 0.2353 - recall: 1.0000 - auc: 1.0000
test loss, test acc: [0.00029895527404733, 1.0, 0.23529411852359772, 1.0, 1.0]
```

Slika 7 – evaluacija modela za prepoznavanje HTML elemenata

Na slici 8 prikazan je rezultat izvršenja naredbe *add div with content Test123*.

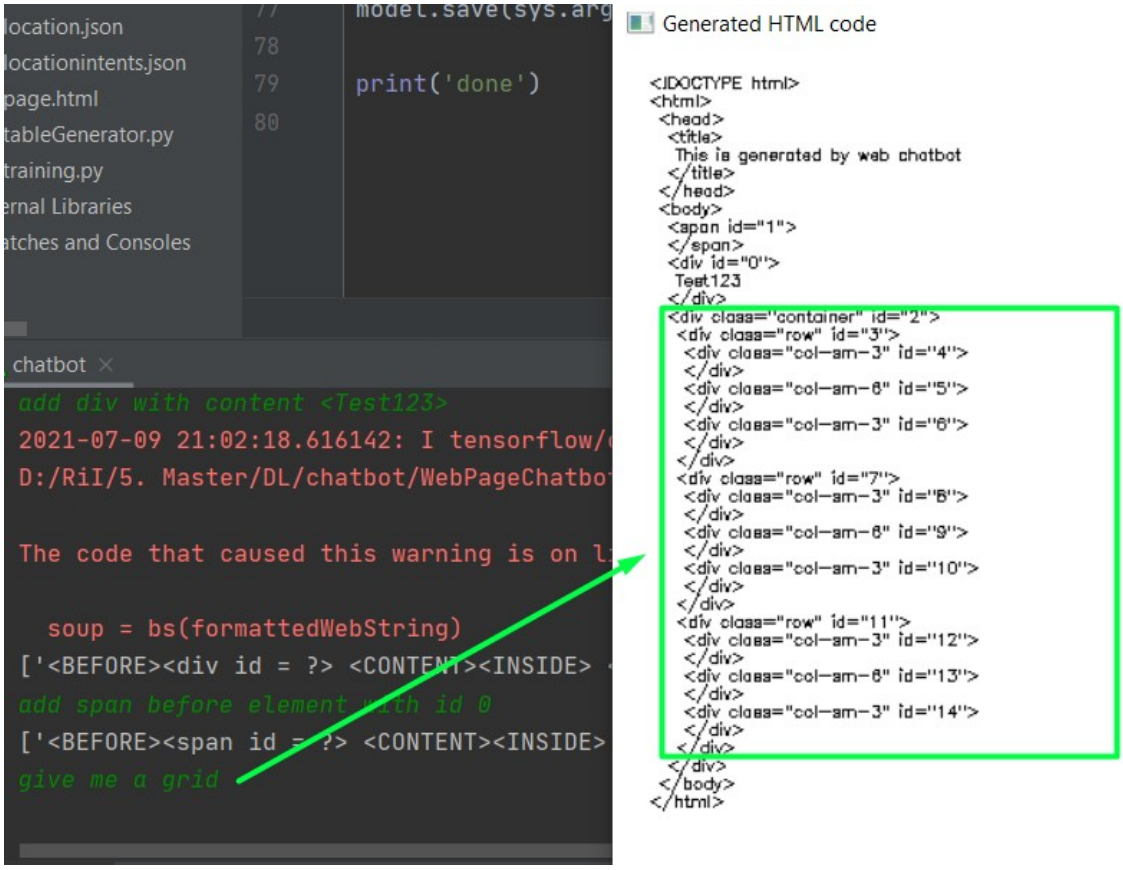


Slika 8 - Naredba *add div with content Test123*



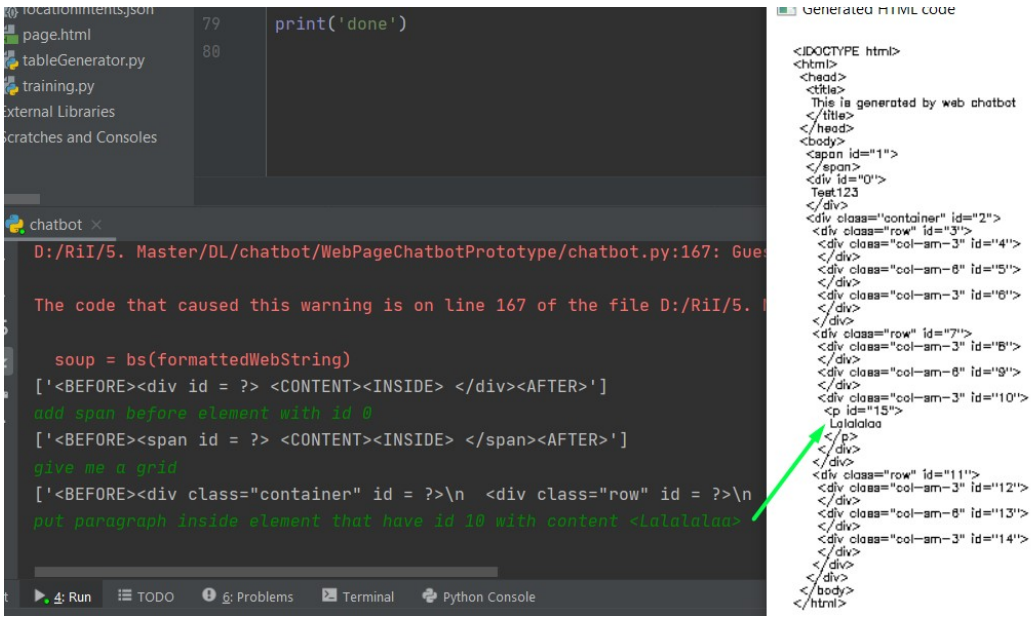
Slika 9 – Dodavanje spana u postojeći sadržaj

Na slici 9 prikazan je rezultat izvršenja naredbe *add span before element with id 0*. Ova naredba sledi naredbi sa slike 8.



Slika 10 – Dodavanje u postojeći sadržaj grid elementa

Na slici 10 je dat rezultat izvrsenja naredbe add me a grid u postojeći sadržaj.



Slika 11 – dodavanje paragrafa unutar elementa koji ima id 10

Na slici 11 dat je rezultat izvršenja naredbe koja treba u već postojeći sadržaj, doda paragraf sa sadržajem lalalala unutar elementa koji ima id 10.

5 Zaključak

Četbotovi su danas vrlo popularan alat koji omogućava automatizaciju mnogih delova preduzeća. Četbotovi su vrlo fleksibilni sa ulazom I izlazom, pa tako imamo I text to speech, text to text, speech to text I speech to speech četbotove.

Generalni princip razvoja četbotova danas je usmeren ka RNN. Ovaj pristup je jako problematičan za male skupove podataka, I ne daje dobre rezultate. Uz to, ovaj pristup ima problem sa "ličnošću" pa tako ako se negde u trening skupu podataka kaže "Ja sam iz Novog Sada" a negde kasnije "Ja sam iz Niša", kada se četbotu postavi pitanje, dobijaće se naizmenični odgovori.

Četbotovi takođe mogu da imaju razne funkcionalnosti I po tome su jako fleksibilni. Jedna od primena je upravo I ova, za genereisanje web strana. Na osnovu korisničkog unosa napravljen je sistem koji kombinacijom dubokog učenja I rules based pristupa efikasno generiše web elemente. Takođe ima mogućnost da postavlja pitanja, da elemente ubacuje jedan u drugi, da dodaje tekstualni sadržaj elementima I da ubacuje elemente jedan u odnosu na drugi.

Mešavina ova dva pristupa daje jako dobre rezultate za jako male skupove podataka što je ovde bio slučaj. Međutim I ovaj sistem ima dosta mogućnosti za unapređivanje.

Jedna od mogućnosti kojom može da se dalje razvija ovaj sistem je da se ubaci mašina stanja. Naime da se proces generisanja elementa razbije na stanja, I da četbot na osnovu unosa prepoznaje u kom se stanju nalazi, I kada skupi dovoljno informacija o elementu koji treba da generiše, odnosno kada dođe u finalno stanje, tek tada se generiše element. Ovim bi generisanje elemenata bilo mnogo fleksibilnije, I ovim bi se uvela pitanja u sistem, jer bi četbot na osnovu stanja u kom se nalazi morao da postavi pitanje o tome šta mu sve fali I kako da to korisnik unese.

Još jedna od mogućnosti proširenja sistema je da se umesto regularnih feed forward mreža koriste rekurentne mreže za retrieval. Takođe, veliki izazov bi bio korišćenje LSTM-ova za end-to-end approach generisanja elemenata. U ovom slučaju bi bilo izazovno pribavljanje dovoljne količine podataka, struktiranja tih podataka kao I generisanje dobre arhitekture mreže koja bi dala adekvatne rezultate.

Literatura

[1] – Yousha Beningio et al, Deep Learning, 2015
[2] – Jutjub video https://www.youtube.com/watch?v=8lG6qRIdSA0
[3] – Britz D., Implementing a Neural Network from Scratch, 2015
http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/
[4] – Nalwan A., Building Jarvis, the Generative Chatbot with an Attitude, 2021
https://towardsdatascience.com/building-jarvis-the-generative-chatbot-with-an-attitude-a833f709f46c
[5] – Jutjub video https://www.youtube.com/watch?v=wypVcNIH6D4
[6] – Codeacademy, Generative Chatbots https://www.codecademy.com/learn/deep-learning-and-generative-chatbots/modules/generative-chatbots/cheatsheet