Chat bot korisničke podrške

Milan Jokanović
jokanovic.r224.2021@uns.ac.rs

Aleksandar Nikolić
aleksandar.nikolic021@uns.ac.rs

Apstrakt–Kvalitet *customer support*-a predstavlja jednu od najbitnijih aspekata zadovoljstva klijenata sa nekom kompanijom. Ovaj kvalitet se ogleda u dve osobine, brzinom kojom kompanija odgovara na potrebe korisnika i koliko je taj odgovor korisniku koristan. U ovom radu je predložen model za kreiranje *chatbot*-a koji bi mogao da automatizuje proces davanja odgovora *customer support*-a na pitanja korisnika, a da se pritom ne izgubi njihov značaj. Model je kreiran korišćenjem rekurentnih neuronskih mreža i enkoder dekoder arhitekture. Skup podataka koji se koristi je dobijen iz *online* skupa podataka *tweet*-ova iz kog su izvučeni parovi korisničkih pitanja i kompanijskih odgovora. U ovom radu obučavani su sledeći modeli: GRU, LSTM, GRU sa GloVe-om i LSTM sa GloVe-om. Najbolje se pokazao GRU model, ali ni on nije dao zadovoljavajuće rezultate.

Ključne reči– *customer support*, *chatbot*, *tweet*, GRU, LSTM, GloVe

1. Uvod

Klijent kada ima određeni problem vezan za uslugu kompanije, pokušava da ostvari komunikaciju sa korisničkom podrškom kompanije od interesa kako bi se problem otklonio. Idealno bi ta komunikacija trebala da bude uspostavljena trenutno i da se problem otkloni u što kraćem roku. Međutim, kako je ograničen broj ljudi korisničke podrške na raspolaganju u nekom trenutku, može se desiti da klijenti čekaju satima pre nego što dobiju bilo kakvu povratnu informaciju. Ovo dovodi do nezadovoljstva klijenata, što može da prouzrokuje njihovo napuštanje korišćenja usluga kompanije i/ili pad kompanijskog imidža u javnosti, što vremenom prerasta u lošije poslovanje.

Ovaj rad za cilj ima da kreira automatizovanu korisničku podršku uz pomoć koje rešava gore navedene probleme i to na sledeći način. Automatizovana korisnička podrška u vidu *chatbot*-a je dostupna klijentu u svakom trenutku i uz odgovarajuće skaliranje može da mu pošalje povratnu informaciju za svega nekoliko sekundi.

Kako je zamisao rada da se kreira *chatbot* koji bi mogao da daje odgovor korisnicima samo za pitanja napisana na engleskom jeziku, jedan od problema sa kojima se on sreće jeste izdvajanje samo onih *tweet*-ova koji su na engleskom. Takodje kako su modelu potrebni parovi korsničkih pitanja i kompanijskih odgovora, potrebno je pronaći odgovarajuće *tweet*-ove iz kojih bi mogli da generišemo ove parove. Po završetku generisanja parova potrebno je izvršiti predprocesiranje njihovog sadržaja uklanjanjem ili zamenom emotikona i

linkova koji bi proizveli šumove u modelu. Sređeni parovi se na kraju prosledjuju enkoder dekoder modelu za treniranje, gde enkoder deo prima korsničko pitanje, a dekoder prima kompanijski odgovor i rezultat enkoder modela.

U daljem nastavku rada detaljno je opisan skup podataka, metode i model korišćeni za rešavanje problema. U narednom poglavlju biće reči o radovima koji su se bavili sličnim problemima. U poglavalju tri detaljno su objašnjene metodologije korišćene za rešavanje problema. Zatim poglavlje četiri detaljno objašnjava skup podataka koji se koristi, a poglavlje pet objašnjava samu implementaciju modela. Poglavlje šest se bavi eksperimentom i problemima koji su pronađeni u okviru eksperimenta. Poslednje sedmo poglavlje daje zaključak na ceo rad.

2. Srodni radovi

U radu [7], Xu, Anbang, et al predstavljeno je rešenje za implementaciju *chatbot*-a za korisničke usluge na socijalnim medijima. Za treniranje modela korišćene su tehnike *deep learning*-a, koji je treniran na skoro milion *Twitter* razgovora. Njihov sistem je zabeležio da preko četrdeset posto korisničkih zahteva su bili emocionalnog karaktera, dok je ostatak bio informativnog tipa. U ovom radu model su evaluirali korišćenjem automatske BLEU tehnike i ljudskom evaluacijom i zaključili su da je *deep learning* tehnika pokazala bolje rezultate u odnosu na druge tehnike.

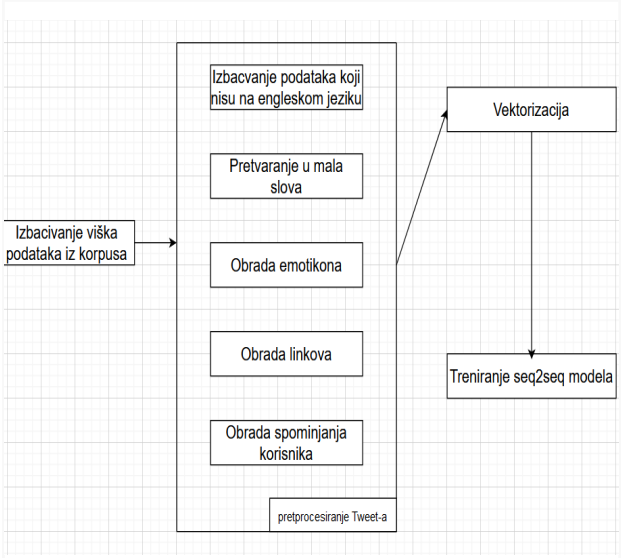
U radu [8], Peters, Florian je implementiran *chatbot* za potrebe korišćenja korisničke podrške kompanije GAMING1. Test podaci koji su korišćeni za treniranje sačinjeni su od tri jezika Engleski, Holandski i Francuski. Model se prvo trenirao na klasifikaciju da li će moći samostalno da odgovori ili je potrebna pomoć čoveka. Zatim u slučaju da je odlučeno da model može sam da reši problem on je dalje treniran na određen broj klasa problema, gde za svaku klasu model daje predefinisan odgovor.

U radu [9], Vinyals, Oriol, and Quoc Le, predstavljen je model *chatbot*-a korišćenjem seq2seq modela koji radi tako što predviđa sledeću rečenicu u razgovoru na osnovu predhodne rečenice ili rečenica. Model je treniran na dva skupa podataka gde je prvo bio domenski skup podataka za IT pomoć i tu se model pokazao poprilično uspešno i uspevao je da nađe rešenje za tehničke probleme iz razgovora. Drugi skup podataka je dobijen uzimanjem konverzacija iz filmova koje nisu bile vezane ni za kakav specifičan domen. Ovaj drugi skup podataka se pokazao lošije, ali je još uvek uspevao uspešno da razume temu razgovora.

3. Metodologije

U ovoj sekciji je objašnjeno kojim metodologijama i na koji način će se vršiti obrada *Tweet*-a od njegovog pribavljanja do transformacije u njegovu vektorsku predstavu koja je pogodna za prosleđivanje modelu mašinskog učenja. Detaljno su objašnjeni modeli i

struktura koja će se trenirati. Ovaj proces je ilustrovan na slici 1.



Slika 1. Put *Tweet*-a u rešenju

3.1. Pretprocesiranje

Pre nego što *Tweet* postane pogodan za prosleđivanje modelu mašinskog učenja, a i kako bi povećali učinkovitost modela, neophodno ga je najpre obraditi na odgovarajući način. U ovom radu ta obrada podrazumeva sledeće:

3.1.1. Izbacivanje svih *Tweet*-ova koji nisu na engleskom jeziku:

Kako želimo da model prepoznaje i radi jedino na engleskom jeziku, neizbacivanje *Tweet*-ova koji nisu na engleskom bi samo stvaralo pometnju i degradiralo performanse modela (pošto bi učio reči i strukture koje ne treba).

3.1.2. Prebacivanje teksta *Tweet*-a u mala slova:

Veličina slova (mala i velika slova) može da nosi određeni sentiment (npr. rečenica napisana velikim slovima može da znači da autor "viče"), te bi u tom slučaju iste reči, napisane malim i velikim slovima, trebalo različito tretirati. Međutim, analizom skupa podataka je ustanovljeno da je broj takvih rečenica izuzetno mali, te da one u ovom slučaju ne pradstavljaju faktor koji bi mogao da utiče na performanse modela. Stoga je odlučeno da se sve rečenice pretvore u mala slova (eng. *lower case*).

3.1.3. Obrada emotikona:

Emotikoni su zamenjeni njihovim sentimentom, odnosno rečima koje predstavljaju njihov sentiment. Ovaj korak je urađen zato što nije planirano da model uči specijalne kodove emotikona iz razloga što postoji mnogo različitih emotikona koji se ponavljaju mali broj puta. Stoga je procenjeno da će model imati veći benefit od reči koje određuju sentiment emotikona (reči koje ga predstavljaju). Druga opcija za obradu emotikona je da se oni u potpunosti izbace, ali s obzirom da se pojavljuju relativno često i mogu da doprinesu boljem razumevanju rečenica, ova opcija nije primenjena. Treba naglasiti da ovaj pristup nije u stanju da generiše emotikone na izlazu.

3.1.4. Obrada linkova:

Ovaj korak podrazumeva pronalaženje svih linkova i njihovu zamenu sa kjlučnom reči "URL_POSITION". Linkovi se obrađuju na ovaj način kako bi se modelu predstavili u samo jednom obliku. Ovo je bitno kako ne bi "gušili" model sa velikim brojem pravih linkova, koji suštinski ništa ne znače. Odnosno na

ovaj način model može da nauči kako da odredi poziciju linka, a posao nekog drugog sistema je onda da na tu poziciju stavi konkretan link.

U narednim podnaslovima će biti diskutovano nekoliko tehnika pretprocesiranja koje se često koriste prilikom problema koji podrazumevaju obradu prirodnog jezika, a u ovom radu nisu korišćenje.

3.1.5. Izbacivanje stop reči:

Pod stop rečima se podrazumevaju reči koje se najčešće javljaju u jeziku, a ne predstavljaju nikakvu informaciju od značaja. Ovaj korak se često primenjuje kod klasifikacije teksta iz razloga što stop reči najčešće ne utiču na klasu, a povećavaju obim reči na koje model treba da pazi, samim tim ga čineći kompleksnijim nego što je neophodno. U ovom radu, s obzirom da se radi o generativnom modelu (model na osnovu ulaza treba da generiše izlaz), ovaj korak nije primenjen. Primena ovog koraka na generativnim modelima bi rezultovala u odgovorima (izlazima iz modela) čija struktura bi bila čudna, odnosno rečenice ne bi bile prirodne u smislu da bi im falili članovi, veznici i slično.

3.1.6. Steming (eng. *Stemming*):

Stemming je tehnika koja služi da se smanji broj dimenzija (reči) u problemima obrade prirodnog teksta. Funkcioniše tako što reči zamenjuje njihovim korenskim oblikom. Ovo rezultuje ne samo manjem broju jedinstvenih reči u korpusu, već i tome da se reči koje imaju isti koren posmatraju na isti način. Primena ovog procesa u generativnim modelima bi dovela do toga da model generiše izlaze koji u sebi sadrže korene reči, što bi moglo da dovede do gubitka značenja rečenice i zabune onoga koji to čita. Stoga ova tehnika nije korišćena u ovom radu. Primer *Stemming*-a je prikazan u tabeli 1.

3.1.7. Lematizacija (eng. Lemmatization):

Lematizacija je proces svođenja reči na njihove leme (eng. *lemma*). Lema reči predstavlja njen osnovni ili rečnički oblik [1]. Ovaj proces je sofisticiraniji od *stemming*-a iz razloga što *stemming* samo odseca kraj reči praveći koren i on recimo ne bi mogao da se izbori sa nepravilnim glagolima u engleskom jeziku tj. postoji šansa da bih ih posmatrao kao dve različite reči (korena), dok je lematizacija za to sposobna tj. bila bi u stanju da zaključi da se radi o jednoj lemi. Ovaj proces nije primenjen u ovom radu iz sličnih razloga kao i *stemming* , odnosno u slučaju njegove primene kod generativnih modela, dobili bi izlaze koji se sastoje od lema i samim tim gramatički netačnu rečenicu. Primer lematizacije je prikazan u tabeli 1.

Početna reč	information	feet	changing	ran
Stemming	inform	feet	chang	ran
Lemmatization	information	foot	change	run

Tabela 1. Primer i poređenje steminga i lematizacije

3.2. Tokenizacija i *word embeddings*

Nakon pretprocesiranja neophodno je *Tweet* odnosno reči pretvoriti u oblik pogodan za modele mašinskog učenja. To podrazumeva transformaciju iz teksta u numerički oblik.

U ovom radu je najpre tokenizacijom napravljen rečnik 10000 najčešće ponavljanih reči skupa podataka i svakoj reči je dodat jedinstveni indeks. Potom se na osnovu tog rečnika skup podataka transformisao tako da su sve reči zamenjene njihovim odgovarajućim indeksima. Ovako transformisan skup podataka je spreman za obučavanje.
Word embedding predstavlja vektorski prostor proizvoljnog broja dimenzija, koji sadrži pozicije reči. Poenta ovakve konstrukcije je da se semantički slične reči nalaze blizu jedna drugoj u vektorskom prostoru. U ovom radu je vršen eksperiment sa pretreniranim *word embedding* modelom *GloVe* i *word embedding* modelom treniranim nad skupom podataka.

3.2.1. GloVe

GloVe predstavlja pretrenirani *word embedding* model koji sadrži vektorsku reprezentaciju reči. Dostupan je u više verzija, odnosno više različitih dimenzionalnosti koje koristi da predstavi reč. Potekao je sa Stanford univerziteta [2]. U ovom radu se koristi i poredi sa *embedding*-om treniranim specifično nad skupom podataka iz ovog rada.

3.3. Modeli mašinskog učenja

U ovoj podsekciji su ukratko objašnjeni modeli mašinskog učenja koji se koriste u ovom radu.

3.3.1. LSTM [4]

LSTM (eng. *Long short term memory*) je poseban tip rekurentne neuronske mreže (u daljem tekstu RNN), koja ima izmenjenu arhitekturu samog neurona. LSTM neuron se sastoji od sledećih elemenata: stanje ćelije (eng. *cell state)*, kapije zaborava (eng. *forget gate*), kapije ulaza (eng. *input gate)* i kapije izlaza (eng. *output gate*). Ulaz u LSTM neuron je skriveno stanje pređašnjeg neurona, *Cell state* pređašnjeg neurona i vektor na ulazu (koji u tom trenutku treba da se obrađuje).

Cell state služi za tok informacija unutar mreže. Ove informacije se u svakoj iteraciji ažuriraju *gate*-ovima i igraju ulogu u kreiranju narednog skrivenog stanja. *Cell state* se može nazvati i dugoročnom memorijom.

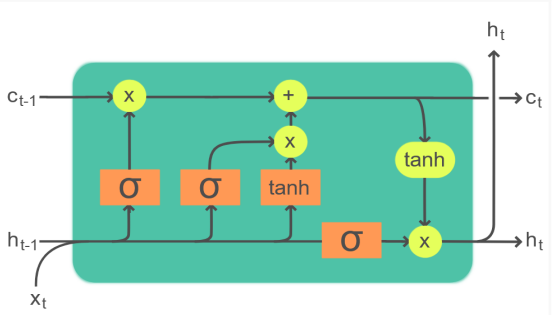
Forget gate je struktura koja uzima prethodno skriveno stanje i trenutni ulaz, spaja ih i propušta kroz

sigmoid funkciju (koja vraća vrednosti između 0 i 1). Rezultati se potom množe sa vrednostima iz *cell state*-a i ulaze u njega. Po prirodi množenja što je vrednost iz *forget gate*-a bliža nuli, to će se ona više zaboraviti pri spajanju sa *cell state*-om.

Input gate je struktura koja uzima prethodno skriveno stanje i trenutni ulaz, te ih propušta kroz sigmoid funkciju i tanh funkciju. Rezultat propuštanja kroz sigmoid funkciju ima za cilj da odluči koje su vrednosti bitne da se ažuriraju. Propuštanje kroz tahn funkciju (koja vraća vrednosti između -1 i 1) ima za cilj da svede ulaz na na opseg [-1,1]. Potom se rezultati ove dve operacije množe i rezultat dodaje na *cell state*.

Output gate uzima vrednosti iz *cell state-a* propuštene kroz tahn funkciju, kombinovanu vrednost pređašnjeg stanja i trenutnog ulaza propuštenu kroz sigmoid funkciju. Množeći rezultate dobije se skriveno stanje koje se prosleđuje narednoj iteraciji.

Ovakva arhitektura LSTM neurona omogućava dugoročno pamćenje, odnosno filtriranje nebitnih, a pamćenje bitnih podataka. Arhitektura neurona LSTM mreže je prikazana na slici 2.



Slika 2. Neuron LSTM mreže

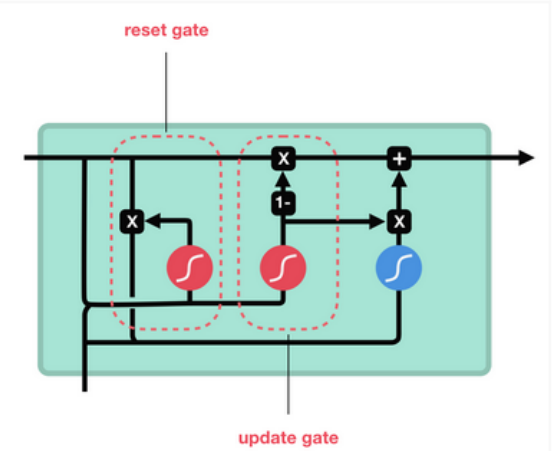
3.3.2. GRU (*Gated recurrent unit*) [5]

GRU predstavlja modifikaciju LSTM-a. Sadrži dva *gate*-a: *update gate* i *reset gate.*

Update gate određuje koliko prethodno stanje utiče na trenutno stanje.

Reset gate određuje koliko novi ulaz utiče na trenutno stanje.

GRU nema *output gate.* Arhitektura neurona GRU modela je prikazana na slici 3.



Slika 3. Neuron GRU mreže

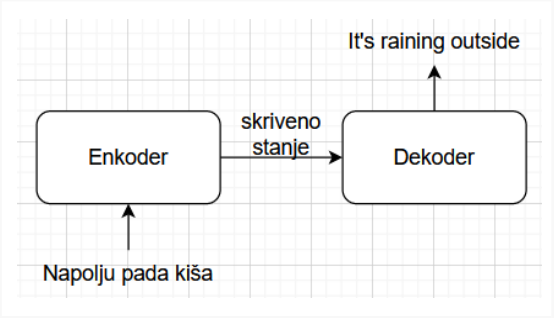
3.3.3. Seq2Seq [6]

Seq2Seq (puna naziv eng. *sequence to sequence*) model je posebna klasa arhitektura rekurentnih neuronskih mreža. Koristi se kada je potrebno da iz sekvence objekata (npr. reči) generiše sekvencu objekata.

Najčešća izvedba ove klase arhitekture je pomoću enkoder dekoder (eng. *encoder decoder*) modela. Ovaj model je prikazan na slici 4.

Enkoder deo modela je RNN koja ima za cilj da prepozna najbitnije podatke sa ulaza i sažme ih. Izlaz enkoder modela se odbacuje, a dekoder modelu se prosleđuje poslednje skriveno stanje.

Dekoder deo modela je RNN koja ima za cilj da na osnovu ulaza (poslednjeg skrivenog stanja enkodera) počne da generiše izlaznu sekvencu. Na izlaznu sekvencu ne utiče samo poslednje skriveno stanje enkodera, već i prethodno generisani delovi sekvence dekodera.



Slika 4. Ideja rada enkoder dekoder modela

3.4. BLEU (eng. *bilingual evaluation understudy*)

BLEU je algoritam za evaluaciju mašinski generisanog teksta. Ovaj algoritam računa rezultat gledajući generisani tekst i referentni tekst, odnosno rezultat je direktno srazmeran broju reči koje se pojavljuju istovremeno i u generisanom i u referentnom skupu. Rezultat BLEU algoritma se uvek kreće u opsegu [0,1], gde nula znači da nema poklapanja uopšte, a jedinica da su referentni i generisani tekst identični [3].

4. Skup podataka

Kako je naš problem, problem mašinskog učenja, krucijalnu ulogu za uspeh našeg rešenja igraju podaci. U ovoj sekciji ćemo razmotriti skup podataka na koji se oslanjamo.

U ovom radu se kao skup podataka (eng. *dataset*) koriste parovi *tweet*-ova (*tweet* i odgovor na njega), koji sadrže komunikaciju između klijenta i kompanije, a pripadaju domenu korisničke podrške. Skup podataka je dostupan na sledećem linku (https://www.kaggle.com/thoughtvector/customer-support-on-twitter?select=sample.csv). Ovaj skup podataka sadrži nešto više od 3 miliona *tweet*-ova u domenu korisničke podrške raznih kompanija. U tabeli 2 su prikazane kolone skupa podataka i dato njihovo objašnjenje. Zbog obima samog skupa podataka i ograničenosti dostupnih računarskih resursa, u radu će biti posmatrane instanca skupa podataka manjeg obima (od oko 20 hiljada pitanje-odgovor parova). Ova instanca sadrži po hiljadu pitanje-odgovor parova dvadeset najzastupljenijih kompanija iz originalnog skupa podataka. Ideja je da vidimo kako se model ponaša učeći iz više izvora podataka, odnosno više različitih kompanija.

Naziv kolone	Opis kolone
tweet_id	jedinstveni identifikator *tweet*-a
author_id	ID autora (moguće da neće biti korišćeno)
inbound	Da li je *tweet* upućen kompaniji ili ne
created_at	Vreme kad je *tweet* nastao (potencijalno će se koristi za analizu)
text	Tekstualni sadržaj *tweet*-a
response_tweet_id	ID *tweet-a* koji je odgovorio na posmatrani *tweet*
in_response_to_tweet_id	ID *tweet*-a čiji je odgovor trenutni *tweet*

Tabela 2. Kolone u skupu podataka

5. Implementacija

Implementacija rada je izvršena u programskom jeziku *Python.*

U procesu pretprocesiranja se za detekciju jezika koristila biblioteka *langdetect* [10].

Za učitavanje/manipulisanje i rad sa podacima se koristila biblioteka *pandas* [12]. Za rad sa neuronskim mrežama i tokenizaciju je korišćen *keras* [13]. Za zamenu emotikona korišćen je rečnik emotikona koji se nalazi u repozitorijumu, a koji sadrži njihove kodove i semantičko značenje. Pored ovih biblioteka u manjoj meri je korišćena biblioteka *scikit learn* [14] i *numpy* [11], kao i ugrađene python funkcije.

6. Eksperiment

Za sprovođenje eksperimenata je potrebno istrenirati modele, metodama predloženim u metodologijama, a zatim izvršiti evaluaciju istreniranih modela. O ovome i potencijalnim problemima pisaćemo u nastavku sekcije.

6.1. Rezultati

U ovom radu su modeli evaluirani na dva načina: ručno i pomoću BLEU algoritma. Prilikom ručne evaluacije se posmatra da li generisani izlaz pruža odgovor na postavljeno pitanje, kao i koliko je generisani odgovor sličan pravom odgovoru. Rezulati ručne evaluacije su prikazani u tabeli 3. Kao što se može videti iz tabele, GRU daje odgovor na pitanje u 45% slučajeva (u 30% odgovor i liči na očekivani), dok LSTM daje odgovor u 44% slučajeva (u 26% odgovor liči na očekivani). Iz ovoga se vidi da oba modela rade dosta slično, s tim što GRU u ovom slučaju ima neznatnu prednost.

BLEU je primenjen na način da je referentni korpus sadržao samo pravi odgovor, odnosno u ovoj izvedbi BLEU očekuje da generisani odgovor bude u potpunosti jednak sa pravim odgovorom. Ovo je proizvelo izuzetno

loše rezultate pri evaluaciji rešenje, više o ovome u poglavlju problemi.

6.2. Problemi

Kao što je već rečeno BLEU je proizvodio jako loše rezultate. Glavni razlog za ovo jeste u tome što nije posedovao više od jedne referentne rečenice i to je dovelo do toga da rečenica koja po svim karakteristikama predstavlja ispravno generisan odgovor dobije rezultat nula. Ovo je prikazano na slici 5.

Jedan od najvećih problema sa kojima se sreće model jeste *overfit*-ovanja tako da model uvek vraća uzak skup jednakih odgovora, a u najgorem slučaju uvek vraća jedan identičan odgovor. Ovaj problem sa najviše ogleda u modelima koji koriste GloVe embedding, dok modeli koji koriste embedding koji se trenirao na skupu podataka uspevaju da prevaziđu ovaj problem. Iz ovog razloga se rezultati Glove modela se ne razmatraju prilikom evaluacije.

Ručno analizom GRU modela pokazalo se dao on ispravno prepoznaje korisnička pitanja koja su emotivnog karaktera, ali došlo je do *overfit*-ovanja odgovora na ovakva pitanja. Prevelika količina emotivnih pitanja negativnog karaktera je dovela do toga da model nauči da za sva emotivna pitanja daje odgovore u obliku izvinjenja bez obzira na to da li je samo pitanje negativnog ili pozitivnog karaktera.

```
Question:  Im full. Thank you @ChipotleTweets  __EMOJI__face_blowing_a_kiss
 __EMOJI__face_blowing_a_kiss __EMOJI__face_blowing_a_kiss __EMOJI__raising_hands __EMOJI__medium_skin_tone

Expected:  <SOS> @119926 We've got you, fam. -Becky <EOS>

Given:
we love you guys too becky
```

Slika 5. Ispravno generisan odgovor sa nula BLEU rezultatom

	Ukupno instanci	Odgovara na pitanje	Slični pravom odgovoru	Totalno pogrešne
LSTM	200	88	52	112
GRU	200	90	60	110

Tabela 3. Rezultati ručne evaluacijeLSTM i GRU modelom sa treniranim embeddingom.

7. Zaključak

U ovom radu je implementiran *chatbot* u domenu korisničke podrške. Sama implementacija uključuje tri koraka: pretprocesiranje, vektorizaciju i treniranje modela.

Pretprocesiranje uključuje niz koraka za prilagođavanje ulaznog skupa podatka kako bi se iz njega izvukao maksimum. Potom se vrši tokenizacija i transformisanje skupa podataka u oblik pogodan za prosleđivanje modelu mašinskog učenja. Konačno se primenjuje *seq2seq* arhitektura sa enkoder dekoder izvedbom koja se trenira da od ulaznog teksta (pitanja) generiše odgovor.

Za implementaciju se koristilo python okruženje. Pomoću biblioteka *keras, langdetect, pandas, numpy, nltk i scikit* moguće je na jednostavan način impelementirati sve potrebne funkcije izloženih metodologija.

Na kraju pojašnjavamo sam eksperiment i prikazujemo dobijene rezultate. Zaključujemo da BLEU daje izuzetno loše rezultate u izvedbi u kojoj smo ga koristili. Daleko bolje rezultate dobijamo ručnom evaluacijom, gde smo ocenjivali da li rečenice daju odgovor koji je sličan ciljnom odgovoru. Kao i da li generisan izlaz pruža odgovor na postavljeno pitanje.

U daljem radu se predlažu tri pristupa za koje smatra da mogu da poboljšaju performanse modela.

Prvi pristup je da se izvrši klasterovanje skupa podataka kako bi se dobile klase pitanja (a samim tim i klase odgovora koje su povezane sa pitanjem). Potom umesto generisanja samog odgovora, odabrati odgovor iz skupa podataka koji pripada prepoznatoj klasi i ima najveću verovatnoću da da odgovor na pitanje (pronalaženje najsličnijeg pitanja iz odabrane klase i uzimanje njegovog odgovora).

Drugi pristup ima za cilj da poboljša BLEU rezultat trenutnog modela. Kako je u trenutnoj implementaciji ciljani odgovor jedini referentni odgovor za BLEU, dobijaju se jako loši rezultati. Ako bi se primenilo klasterovanje nad skupom podataka i dobile klase pitanje-odgovor. Mogla bi se cela jedna klasa odgovora iskoristiti kao referentni odgovori za BLEU. Ovo ne samo da bi moglo poboljšalti BLEU rezultat već bi i ovaj rezultat nosio veću težinu, s obzirom da bi mogao bolje da oceni suštinu (intent) odgovora.

Treći pristup je proširenje skupa podataka, odnosno korišćenja sva 3 miliona dostupnih podataka iz skupa. Ovo nije bilo moguće da se primeni usled nedostatka računarskih resursa.

8. Reference

[1] *Collins English Dictionary*, entry for "lemmatise"

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[3] Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. J. (2002). *BLEU: a method for automatic evaluation of machine translation* (PDF). ACL-2002: 40th Annual meeting of the Association for Computational Linguistics. pp. 311–318.

[4] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". *Neural Computation*. **9** (8): 1735–1780.
[5] Cho, Kyunghyun; van Merrienboer, Bart; Bahdanau, DZmitry; Bengio, Yoshua (2014). "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches".
[6] Sutskever, Ilya; Vinyals, Oriol; Le, Quoc Viet (2014). "Sequence to sequence learning with neural networks".
[7] Xu, Anbang, et al. "A new chatbot for customer service on social media." *Proceedings of the 2017 CHI conference on human factors in computing systems*. 2017.
[8] Peters, Florian. "Master thesis: Design and implementation of a chatbot in the context of customer support." (2018).
[9] Vinyals, Oriol, and Quoc Le. "A neural conversational model." *arXiv preprint arXiv:1506.05869* (2015).
[10] https://pypi.org/project/langdetect/
[11] https://numpy.org/doc/
[12] https://pandas.pydata.org/
[13] https://keras.io/
[14] https://scikit-learn.org/stable/