Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/06BE26Stankovic

ANALIZA ALGORITAMA ZA VEKTORSKU REPREZENTACIJU TEKSTA

ALGORITMS FOR VECTOR REPRESENTATION OF TEXT

Milan Stanković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – Računarstvo i automatika

Kratak sadržaj – *Obrada teksta odnosno NLP (Natural language processing) predstavlja atraktivnu oblast u oblasti računarske inteligencije. U ovom radu će biti predstavljeno više načina za pretvaranje teksta u vektore. Postoje dve velike kategorije, prva se zasniva na statističkim metodama a druga na korišćenju neuronskih mreža. Prednost vektorske predstave je mogućnost da se takvi vektori porede ili dalje lakše koriste u nekoj od metoda mašinskog učenja. Vektorska reprezentacija održava deo semantike, tako da predstavlja pogodnu metodu za reprezentaciju i poređenje kompleksnijeg teksta.*

Ključne reči: *enkapsulacija, vektorsko predstavljanje reči, glove, word2vec, skipgram, cbow, duboko mašinsko učenje*

Abstract – *Natural language processing (NLP) is an attractive area of computing intelligence. This paper will introduce several ways to convert text to vectors. There are two major categories, the first based on statistical methods and the second using neural networks. The advantage of vector representation is the ability to compare or further use such vectors in one of the machine learning methods. Vector representation maintains some semantics, so it is a convenient method for representing and comparing more complex text.*

Keywords: *encapsulation, word vectors, glove, word2vec, skipgram, cbow, deeplearning*

1. UVOD

Postoji mnogo načina za predstavljanje teksta u dokumentu kao i njegovu analizu. Trenutno jedan od najpopularnijih i uspešnijih načina za reprezentaciju teksta je enkapsulacija. Cilj enkapsulacije je da se tekst pretvori u oblik koji mašina, odnosno neuronske mreže ili algoritmi za mašinsko učenje, mogu lakše da koriste. Tekst, za razliku od numeričkih podataka, nije moguće direktno analizirati gledajući samo ASCII vrednosti karaktera. Retko kada ima smisla gledati samo pojedinačne karaktere, jer bez da vidimo koji se nalaze ispred i iza njih ne možemo dobiti celu predstavu o tekstu. Većina algoritama za transformaciju teksta rade po principu pretvaranja teksta u vektorski oblik, češće višedimenzioni.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.

Ako ima potrebe samo za prebacivanje u vektorski oblik, osnovni algoritmi to rade time što svakoj jedinstvenoj reči dodeljuju numeričku vrednost, a onda se ceo tekst može predstaviti kao niz brojeva. Ovom osnovnom transformacijom sami brojevi samo zamenjuju reči, ali ne doprinose razumevanju te reči. Na osnovu dobijenog niza moguće je dobiti malo konteksta za sve reči koje se nalaze u tekstualnom korpusu. Čistom analizom frekvencije pojave nekih reči potencijalno možemo dobiti koje su reči značajne u tom tekstu. Međutim, u praksi se ovakvi jednostavni pristupi nisu pokazali kao previše korisni.

2. OSNOVNI ALGORITMI ZA ENKAPSULACIJU

Svakodnevni govor, ali i pisani tekstovi, se sastoje od dosta veznika, rečica, ali i uzrečica koje pisac koristi. Reč tog karaktera bi se pojavljivale kao najznačajnije, dok su suštinski retko kada bitne. Takvom analizom ne dobijamo gotovo nikakve korisne informacije. Moguće je izbaciti sve reči koje su kraće od zadate dužine ili, još bolje, da imamo korpus veznika i ostalih reči koji ne nose veliki semantički značaj. Prilikom analize teksta njih ne uzimati u obzir. Ovakvi pristupi nam, nažalost, rešavaju samo deo problema.

Drugi veliki problem kod ovakve predstave teksta je činjenica da reči koje su u množini ili u različitim padežima, vremenima, superlativi ili sinonimi bi bili svi predstavljeni kao različiti brojevi i, samim time, kao u potpunosti različite reči. Reči “knjiga“, „knjige“, „knjizi“ nikako ne bi bile povezane, iako se u suštini svode na istu stvar. Postoje algoritmi koji rade lematizaciju (odbijanje prefiksa i sufiksa) i stemovanje (svođenje reči na njen koren) ili oni koji porede reči na osnovu broja koraka koje je potrebno obaviti kako bi se jedna reč pretvorila u drugu. Ovakvi pristupi često mogu da rade dobro, čak i kada ima gramatičkih grešaka. Njihovom upotrebom se dobija dosta bolja predstava teksta, ali se gubi dosta konteksta i ne rešavaju se problemi žargona, sinonima.

Ako uzmemo u obzir reč “radio” ona može imati skroz različito značenje u “radio sam na tom projektu” naspram “uključio sam radio”. U prvom obliku je glagol koji predstavlja predikat u rečenici, dok je u drugom imenica koja prestavlja objekat. Rešenje ovih problema zahtevaju dosta kompleksnije algoritme. Kako je govor, a samim time i tekst, kompleksan sa dosta nijansi i razlika između jezika, kultura, govornih područja, još uvek nije napravljen savršen sistem za reprezentaciju i analizu teksta.

Češći način za prebacivanje teksta u neku numeričku reprezentaciju je da se svakoj reči dodeli vektor. Osnovni oblik ovakve transformacije, se naziva ***one-hot encoding***

(kodiranje). Kako bi se ono odradilo potrebno je znati broj jedinstvenih reči u tekstu koji se analizira. Recimo za rečenicu "Ja volim NLP i ja obožavam skijanje !". Dužina ***one-hot*** vektora bi bila šest. Prvo je potrebno odraditi osnovno procesiranje teksta. Reč "JA" se u rečenici nalazi u dva oblika "Ja" i „ja", kako bi se izbegle greške koje mogu uslediti zbog velikih i malih slova, najčešće se sva slova pretvaraju u velika. Uz to se otklanjaju i svi specijalni karakteri kao što su navodnici, tačke, upitnici… Ova konverzija povremeno može uvesti nove greške. Zatim, dodeljujemo svakoj reči vrednosti koje su nula ili jedan po principu da će svi vektori biti jedinstveni i da će svaki vektor imati samo jednu jedinicu i ostale vrednosti će biti nule. Na kraju se dobijaju vektori koji su prikazani u tabeli br. 1. Ovom tehnikom je za isti ulaz moguće dobiti različite vektore za iste reči. Na ovaj način imamo mogućnost da predstavimo bilo koju reč u tekstu kao vektor, ali taj vektor, nažalost, ne sadrži nikakvu semantiku.

JA	=	[	1,	0,	0,	0,	0,	0]
VOLIM	=	[	0,	1,	0,	0,	0,	0]
NLP	=	[	0,	0,	1,	0,	0,	0]
I	=	[	0,	0,	0,	1,	0,	0]
OBOZAVAM	=	[	0,	0,	0,	0,	1,	0]
SKIJANJE	=	[	0,	0,	0,	0,	0,	1]

Tabela 1. Prikaz one-hot kodiranja

Postoje i kompleksniji oblici pretvaranja reči u vektor, koji uzimaju u obzir kontekst, odnosno odnos reči jednih prema drugima. Jedan od načina da se odradi ovakva reprezentacija je da se napravi matrica uzajamnog pojavljivanja. Na slici 1. se može videti ovakva matrica na primeru rečenice "Ja volim NLP i ja obožavam skijanje !".
U slučaju da se u rečenici, odnosno tekstu, reči nalaze jedna do druge u tabeli na mestu preseka se vrednost postavlja na jedan. Osnovni primer ove tabele može imati samo vrednosti jedan ili nula. Bolji algoritmi popunjavaju tabelu tako što povećavaju vrednost za jedan, za svako ponavljanje susedstva. Zbog različite frekvencije reči, često se i drugi način predstave reči na kraju skalira na domen od [0..1] ili [-1..1]. Skaliranje pomaže pri daljim analizama time što smanjuje varijabilnost.

	JA	VOLIM	NLP	I	OBOZAVAM	SKIJANJE
JA	0	1	0	1	1	0
VOLIM	1	0	1	0	0	0
NLP	1	0	0	1	0	0
I	1	0	1	0	0	0
OBOZAVAM	1	0	0	0	0	1
SKIJANJE	0	0	0	0	1	0

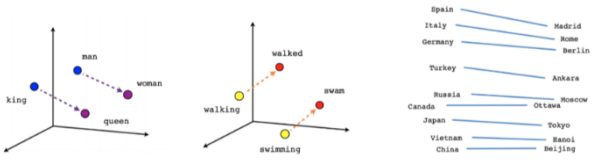
Slika 1. Primer matrice pojavljivanja

Na osnovu matrice moguće je kreirati vektore za svaku reč. Za primer prikazan iznad bi se kreirali vektori kao što je prikazano u tabeli ispod.
Za razliku od samog teksta, vektore je moguće porediti. Preko Euklidskog rastojanja moguće je odrediti koliko su dva vektora blizu jedan drugom.
Rastojanje možemo predstaviti kao sličnost dve reči, odnosno što su vektori bliži to su dve reči sličnije. Što je veći tekstualni korpus na osnovu koje se generišu ove matrice to je veća tačnost.

JA	=	[	0,	1,	0,	1,	1,	0]
VOLIM	=	[	1,	0,	1,	0,	0,	0]
NLP	=	[	1,	0,	0,	1,	0,	0]
I	=	[	1,	0,	1,	0,	0,	0]
OBOZAVAM	=	[	1,	0,	0,	0,	0,	1]
SKIJANJE	=	[	0,	0,	0,	0,	1,	0]

Tabela 2. Prikaz vektora

Postoje modeli zasnovani na neuronskim mrežama obučavaju se na ogromnom korpusu teksta kako bi što bolje reprezentovali reči u nadi da će se u njihovom vektorskom obliku nekako održati sama semantika i rešiti svi prethodno napomenuti problemi. Noviji probabilistički modeli, u određenim zadacima, su pokazali sličnu tačnost kao modeli zasnovani na neuronskim mrežama. Njihova velika prednost je da nemaju potrebu da se obučavaju.
Budući da su na kraju dobijeni vektori, osim poređenja, moguće je raditi i ostale matematičke operacije. Dobro obučeni modeli su pokazali da postoji održanje nekog vida konteksta i da je moguće izvući zanimljive zaključke. Na slici 2. se vidi da je rastojanje između reči muškarac (man) i žena (woman) isto kao rastojanje između reči kralj (king) i kraljica (queen), ovo održanje rastojanja između reči različitog roda se isticalo u mnogim primerima. Isto važi i za rastojanje između reči u različitim vremenima ili udaljenost između glavnog grada i države u kojoj se nalazi. Uz dodatne algoritme, moguće je čak vektorskim sabiranjem dve reči dobijati treću. Recimo reč ***apple*** (jabuka) i ***purple*** (ljubičasta) da daju reč ***plum*** (šljiva).



Slika 2. Poređenje održanje rastojanja

3. GLOVE MODELI

Radovi kao što je "GloVe: Global Vectors for Word Representaton"[1] prikazali su da je moguće reprezentovati reči kao n-dimenzione vektore koji sadrže neki vid semantike same reči. ***Glove*** algoritam je zasnovan na probabilističkim teorijama umesto na korišćenju ***LSTM***-a ili drugih neuronskih mreža, kao u ***word2vec*** i ostalim algoritmima za reprezentaciju reči sa vektorima. Korišćene su matrice međusobnih pojava gde vrednost u matrici predstavljaju koliko se puta jedna reč nalazi u kontekstu druge.

Kako bi se kreirao vektor za svaku reč se gleda u kojim se kontekstima javlja i dodeljuju se vrednost reči koje su blizu, sa tim da je uveden faktor opadanja na osnovu toga koliko je daleka jedna reč od druge.

Glove model ima procentualnu tačnost od 82% u kontekstu semantike. U testovima za sličnost reči je korišćena kosinusna sličnost za evaluaciju vektora, i dobijeni rezultat variraju u zavisnost skupa podataka za testiranje koji se koristi (za četiri od pet skupova je tačnost oko 70%, dok je za poslednji značajno manja sa 40%).

Ovaj model ima veliku prednost naspram algoritama zasnovanim na neuronskim mrežama iz razloga što se mora obučavati i često pokazuje veću tačnost naspram njih, ako je tekstualni korpus mali. Postoje već i unapred odrađeni ***Glove*** modeli koji su kreirani nad ogromnim korpusima teksta. Kada je pametnije koristiti pretreniran model i eventualno ga dotrenirati, a kada je potrebno kreirati nov model je pitanje koje muči većinu ljudi koji se bave analizom teksta.

Na ovo pitanje ne postoji univerzalan odgovor. Odgovor zavisi od korišćenog teksta. Što je tekst koji se treba analizirati sličniji tekstu koji je korišćen u kreiranju pretreniranog modela, to više ima smisla koristiti pretreniran model uz eventualne izmene ili dopune.

Sa druge strane, veoma zavisi i od veličine tekstualnog korpusa, odnosno komputacione moći računara na kojem je potrebno kreirati novi model.

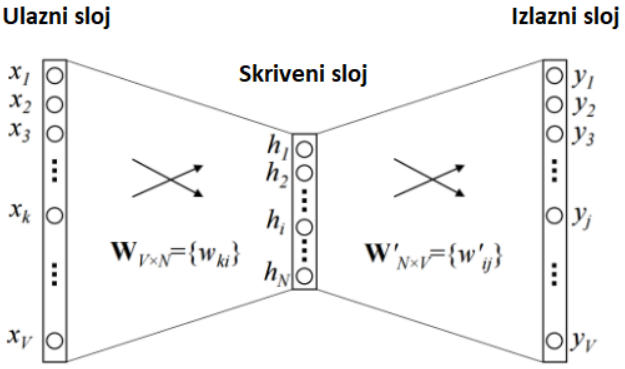
4. WORD2VEC MODEL

Word2Vec model (reč u vektor) je model zasnovan na neuronskim mrežama koji vrši transformaciju reči u njihov vektorski oblik. U prethodnim primerima je prikazano da je moguće od bilo kojeg teksta kreirati tekst, proces sam po sebi nije složen. Međutim održanje konteksta, odnosno suštine teksta je deo u kojem ***word2vec*** pokazuje dobre rezultate. On je trenutno jedan od najčešće upotrebljavanih metoda za ovaj posao, budući da u opštem slučaju daje najbolje rezultate bez previše obučavanja. Vektore je moguće dobiti pomoću dve metode, obe uključuju neuronske mreže : ***Skip Gram*** i ***Common Bag Of Words (CBOW***). Obe metode su sadržane u ***word2vec*** modelu, korisnik samostalno bira koju će verziju koristiti.

4.1 CBOW model

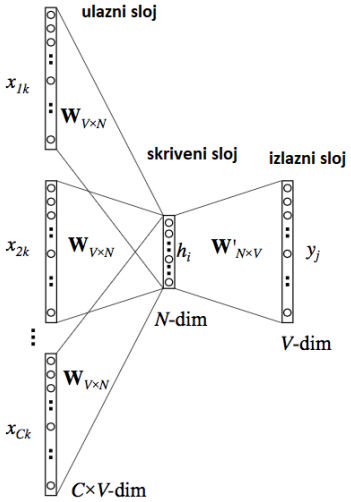
Ovaj model uzima u obzir kontekst svih reči kao ulaz i pokušava da predvidi reč koja najviše odgovara kontekstu. Ako je recimo unos u neuronsku mrežu reč “dobar“ želimo da nam se predvidi da je sledeća reč “dan”. Za reč “dan” se vrši ***one-hot*** kodiranje**.** Kao ulaz CBOW mreže šaljemo ***one-hot*** vektor, a kao izlaz očekujemo vektor reči “dobar”. Računa se greška i menjaju se težine neuronske mreže. U toku predviđanja ciljne reči dobijamo pravu vektorsku reprezentaciju ciljne reči. Suštinski skriveni sloj neuronske mreže sadrži vektorske reprezentacije reči. Tačnost zavisi od konteksta, odnosno od broja reči koje uzimamo pre ciljane reči.

Na slici 3. možemo videti ilustraciju CBOW mreže koja ima kontekst dužine jedan. Ulazna ili kontekstna reč je ***one-hot*** kodiran vektor veličine V. Skriveni sloj sadrži N neurona, a izlaz je opet vektor dužine V. Izlazi se dobijaju kao izlazne vrednosti softmax funkcije. Neuroni skrivenog sloja samo kopiraju zbir ulaza pomnožen sa težinama u sledeći sloj. Ne koristi se aktivaciona funkcija poput sigmoida, tanh ili ReLU. Jedina nelinearnost se dobija korišćenjem ***softmax***-a u izlaznom sloju. Umesto jedne ulazne reči za kontekst možemo koristiti njih više.



Slika 3. CBOW model sa kontekstom dužine jedan

Model prikazan na slici 4. sadrži reči iz konteksta C. Kada se ***Wvn*** koristi za izračunavanje vrednosti težina sakrivenih slojeva, uzima se prosek za svih ***C*** ulaznih kontekstnih reči.



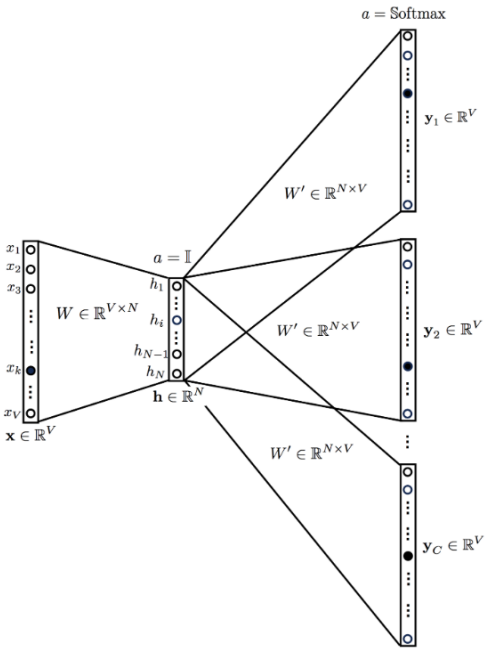
Slika 4. *CBOW mreža sa ulazom dužine C*

Upotrebom ove mreže možemo generisati vektorski prikaz reči korišćenjem konteksta. Postoji još jedan način da se učini isto. Moguće je koristiti ciljnu reč (reč čiju reprezentaciju tražimo), kako bi predvideli kontekst i tokom tog procesa proizvesti vektorske reprezentacije. Druga varijanta se zove ***Skip-Gram*** model.

4.2 Skip-Gram model

Skip-Gram model je, u suštini, samo obrnut CBOW model, slika 5. Izgleda kao invertovan višekontekstni CBOW model. U mrežu se unosi ciljna reč, a model daje C distribuciju verovatnoće. Za svaku poziciju u kontekstu dobijamo C verovatnoće podele V verovatnoća, po jednu za svaku reč. U oba slučaja mreža koristi propagaciju unazad za obučavanje.

Oba modela imaju svoje prednosti i mane. ***Skip-Gram*** model se pokazao kao bolji u slučaju da se radi sa malom količinom podataka, kao i u slučaju gde imamo retke reči. S druge strane, CBOV je brži i bolje se pokazao za predstavljanje češćih reči.



Slika 5. *Skip-Gram model*

U radu "Distributed Representations of Words and Phrases and their Compositionality" [2] su predstavljene osnove ***word2vec*** modela.
Oni su za obuku ***Skip-gram*** modela koristili veliki skup podataka koji se sastoji od različitih članaka. Kreiran je interni Google-ov skup podataka sa milijardu reči. Sve reči koje su se pojavile manje od pet puta su izbačene iz rečnika i kreiran je skup sa vokabularom veličine 692 hiljada reči. Trenirajući ***Skip-gram*** model na ovim skupom dobijena je tačnost od 72% kada su pogledu održanja konteksta.

5. ZAKLJUČAK

U ovom radu su predstavljene metode za enapsulaciju teksta, odnosno njegovo vektorsko predstavljanje. Trenutno ne postoji najbolje rešenje za ovaj problem ali se algoritmi razvijaju u dva pravca. Rešavaju se koristeći probabilističke modele kao i neuronske, najčešće rekurentne, mreže. Činjenica da word2vec kao i glove model zadržava deo semantike predstavlja veliku prednost pri analizi tekstova.

Zbog svog vektorskog oblika, vektori koji reprezentuju rečenice koje sadrže gramatičke i pravopisne greške se nalaze blizu vektora takvih rečenica sa ispravljenim greškama. Oba modela imaju svoje prednosti i mane tako da odabir zavisi prvenstveno od samog skupa podataka sa kojim se radi. Word2vec model daje bolje rezultate nad većim skupovima podataka pod uslovom da se omogući dovoljno vremena za treiranje modela.

Za dalje istraživanje, unapređivanje vektorizacije teksta upotebom novih transformer modela bi predstavljalo zanimljivu temu koja bi potencijalno dovela do značajno boljih rezultata.

6. LITERATURA

[1] Quoc Le, Tomas Mikolov, ``GloVe: Global Vectors for Word Representation", Computer Science Department, Stanford University, Stanford, CA 94305, 2014.

[2] Quoc Le, Tomas Mikolov, ``Distributed Representations of Sentences and Documents", Google Inc, 1600 Amphitheatre Parkway, Mountain View CA 94043, May 2014.

[3] Mikolov, Tomas & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

Kratka biografija:

Milan Stanković rođen je u Subotici 1995. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva Neuroevolucija u Java programskom jeziku odbranio je 2018.god.
kontakt: milan.s@uns.ac.rs