Hierarchical Attention Network za klasifikaciju dokumenata

Sadržaj

---

1. Uvod

1.1 Opis problema

Klasifikacija dokumenata predstavlja jedan od ključnih problema u oblasti obrade prirodnog jezika (NLP). Cilj je automatski dodeliti dokumente unapred definisanim kategorijama na osnovu njihovog sadržaja. Efikasna klasifikacija dokumenata ima primenu u mnogim domenima, uključujući analizu sentimenta korisničkih recenzija, filtriranje sadržaja, sistem za preporuke i automatsko indeksiranje informacija.

Ova arhitektura pokazuje da modeliranjem hijerarhijske strukture dokumenata mogu da se postignu značajna poboljšanja u pogledu njihove klasifikacije. Tradicionalni modeli poput bag-of-words ili klasičnih RNN/LSTM modela često zanemaruju ovu hijerarhiju i međusobnu važnost reči i rečenica.

1.2 Pregled literature

Yang i njegovi saradnici (2016) su u svom radu *Hierarchical Attention Networks for Document Classification* predstavili model koji uzima u obzir hijerarhijsku strukturu dokumenata. HAN takodje uzima u obzir kontekstne zavisnosti reči i rečenica jednih od drugih. HAN model je pokazao značajno poboljšanje u odnosu na tradicionalne RNN i CNN pristupe za klasifikaciju

dokumenata kao i u odnosu na linearne i SVM modele (mada su oni već prevaziđeni od strane modela koji koriste neuronske mreže).

1.3 Cilj rada

Cilj ovog rada je reprodukovati HAN model u okviru PyTorch okruženja i primeniti ga na dataset Amazon recenzija kao i uporediti dobijene rezultate sa onima iz postojeće literature.

---

2. Opis rešenja

2.1 Princip HAN arhitekture

Hierarchical Attention Network (HAN) model koristi hijerarhijsku strukturu teksta kako bi efikasno obradio dokumente. Model se sastoji od: enkodera sekvenci reči, mehanizma pažnje na nivou reči, enkodera sekvenci rečenica i mehanizma pažnje na nivou rečenica. Enkoderi su bazirani na GRU(Gated Recurrent Unit) arhitekturi i odlikuju ih dve kapije: update i reset kapija. Njihova uloga ukratko jeste da odrede koliko će na trenutno skriveno stanje uticati novi ulaz i prethodno skriveno stanje. Dakle u vremenskom trenutku t GRU računa sledeće stanje po formuli:

$$h\mathrm{t} = (1 - z\mathrm{t}) \odot h\mathrm{t}{-}1 + z\mathrm{t} \odot \tilde{h}\mathrm{t}.$$

Update i reset kapija (zt i rt) kao i kandidat stanje (~ht) računaju se po sledećim formulama:

$$z\mathrm{t} = (W\mathrm{z}x\mathrm{t} + U\mathrm{z}h\mathrm{t}{-}1 + b\mathrm{z}),$$

$$\sim\mathrm{ht} = \tanh(W\mathrm{h}x\mathrm{t} + r\mathrm{t} \odot (U\mathrm{h}h\mathrm{t}{-}1) + b\mathrm{h})$$

$$r\mathrm{t} = (W\mathrm{r}x\mathrm{t} + U\mathrm{r}h\mathrm{t}{-}1 + b\mathrm{r}).$$

Dakle HAN prvo propusti reči (embedovane) kroz GRU enkoder u oba smera i zatim spoji (konkatenira) njihov izlaz (skrivena stanja) kako bi dobio reprezentaciju reči. Zatim ta reprezentacija prolazi kroz jedan sloj MLP-a (Multi Layer Perceptron) i na osnovu njegovog izlaza se računa težina pažnje (govori koliko je reč značajna za odredjivanje konteksta rečenice tj. koliko nam informacija daje). Pažnja se računa kao sličnost sa globalnim kontekstnim vektorom (možemo ga posmatrati kao upit „koje reči daju značajne informacije“). Vektor rečenice dobijamo tako težinskom sumom svih vektora reči u toj rečenici (težine pažnje * vektori reči). Isti proces se ponavlja na nivou rečenice kako bi se dobio vektor dokumenta. Taj vektor prolazi kroz jednoslojni MLP sa aktivacionom funkcijom softmax kako bi mreža na izlaz dala verovatnoće svake klase.

Globalni kontektni vektori (jedan za reči i jedan za rečenice) se inicijalizuju nasumično i uče se kroz treniranje mreže.

2.2 Implementacija

Za implementaciju ovog modela korišćena je biblioteka pytorch koja daje interfejs za kreiranje neuronskih mreža kroz objektno orijentisano programiranje. Za implementaciju radi vežbe nije korišćen modul ove biblioteke za GRU. Implementacija se sastoji iz GRU modela, Attention modela i najzad samog HAN modela koji spaja ove komponente. Implementacija prihvata „bečovane“ ulaze mada uprkos tome ima znatno sporije performanse nego što bi to bio slučaj da je

koršćen nn.GRU modul pytorch biblioteke zato što je nn.GRU značajno optimizovan u svim poljima, a pogotovo na polju paralelizacije. Kao što će kasnije biti ponovljeno zbog ovoga, a i zbog hardverskih i vremenskih ograničenja model je treniran na znatno manjem skupu podataka od onog koji je opisan u originalnom papiru.

2.3 Preprocesiranje podataka

Pre nego što se podaci proslede modelu, primenjuju se sledeći koraci:

1. **Tokenizacija** – dokumenti se dele na rečenice, a rečenice na reči koristeći NLTK.
2. **Padding i skraćivanje** – kako bi svi ulazi imali istu dimenziju, rečenice se skraćuju na maksimalnu dužinu, a kraći nizovi se dopunjuju sa `<PAD>` tokenom. Isto važi za broj rečenica u dokumentu.
3. **Zamena reči van vokabulara** – reči koje se pojavljuju manje od 5 puta zamenjuju se sa tokenom `<UNK>`.
4. **Kreiranje embedding matrice** – Word2Vec embedding se trenira na skupu trening podataka i koristi se za inicijalizaciju embedding sloja modela. Za ovo je korišćena gensim biblioteka tj njen Word2Vec model.
5. **Konverzija u indekse** – tokenizovane reči se mapiraju na indekse iz vokabulara i transformišu u PyTorch tensor.

2.4 Trening i hiperparametri

Model je treniram na Amazon reviews polarity skupu (tačnije na malom uzorku tog skupa od 8000 recenzija zbog vremenskih i hardverskih ograničenja kao i zbog sporije implementacije u odnosu na nn.GRU) sa sledećim parametrima: batch_size=256, embedding_size=100, hidden_size=25 (ovi parametri su namerno umanjeni u odnosu na parametre navedene u orignialnom papiru radi ubrzanja procesa treniranja). Learning rate pronadjen je kao i u gore opisanom papiru korišćenjem tehnike grid search. Za nekoliko vrednosti ovog parametra model je treniran (u svega nekoliko epoha, dovoljno samo da se vidi trend opadanja loss-a) i evaluiran na validacionom skupu. Za svaku vrednost ispisane su metrike i grafički je prikazan trend opadanja gubitka kroz epohe kako bi se stekla što bolja slika o tome koji learning rate treba prihvatiti. Model je nakon toga treniran sa learning rate-om od 0.01 (on se pokazao kao najbolji u grid search-u) kroz 15 epoha (idealno bi bilo više mada zbog smanjene veličine skupa i gore navedenih ograničenja i ovo je dovoljno). Za optimizator korišćen je Adam, a funkcija gubitka bila je nn.NLLLoss() odnosno Negative Log Likelihood.

Za evaluaciju modela od metrika su korišćene: accuracy, loss, f1 score i matrica konfuzije.

---

3. Eksperimentalni rezultati

3.1 Originalna implementacija u odnosu na prethodnike

	Methods	Yelp'13	Yelp'14	Yelp'15	IMDB	Yahoo Answer	Amazon
Zhang et al., 2015	BoW	-	-	58.0	-	68.9	54.4
	BoW TFIDF	-	-	59.9	-	71.0	55.3
	ngrams	-	-	56.3	-	68.5	54.3
	ngrams TFIDF	-	-	54.8	-	68.5	52.4
	Bag-of-means	-	-	52.5	-	60.5	44.1
Tang et al., 2015	Majority	35.6	36.1	36.9	17.9	-	-
	SVM + Unigrams	58.9	60.0	61.1	39.9	-	-
	SVM + Bigrams	57.6	61.6	62.4	40.9	-	-
	SVM + TextFeatures	59.8	61.8	62.4	40.5	-	-
	SVM + AverageSG	54.3	55.7	56.8	31.9	-	-
	SVM + SSWE	53.5	54.3	55.4	26.2	-	-
Zhang et al., 2015	LSTM	-	-	58.2	-	70.8	59.4
	CNN-char	-	-	62.0	-	71.2	59.6
	CNN-word	-	-	60.5	-	71.2	57.6
Tang et al., 2015	Paragraph Vector	57.7	59.2	60.5	34.1	-	-
	CNN-word	59.7	61.0	61.5	37.6	-	-
	Conv-GRNN	63.7	65.5	66.0	42.5	-	-
	LSTM-GRNN	65.1	67.1	67.6	45.3	-	-
This paper	HN-AVE	67.0	69.3	69.9	47.8	75.2	62.9
	HN-MAX	66.9	69.3	70.1	48.2	75.2	62.9
	HN-ATT	**68.2**	**70.5**	**71.0**	**49.4**	**75.8**	**63.6**

Table 2: Document Classification, in percentage

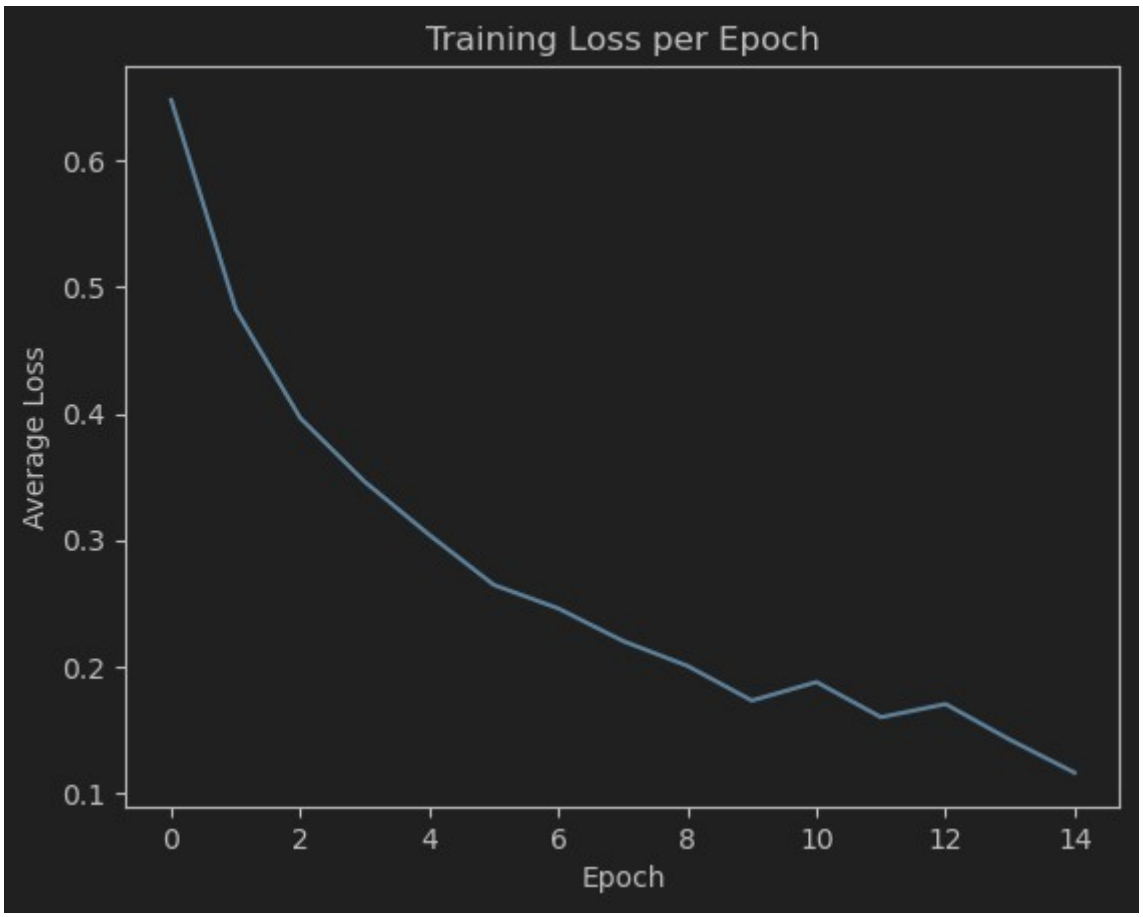
Na datoj tabeli možemo da vidimo kako se originalna implementacija pokazala u odnosu na njene prethodnike. Vidimo da na svim skupovima ima prednost od bar nekoliko procenata u odnosu na modele koji ne uzimaju u obzir hijerarhijsku strukturu dokumenata.

3.2 Dataset

Implementacija napravljena za ovaj projekat trenirana je na uzorku od 8000 recenzija sa Amazona. Za validacioni kao i za test skup korišćeno je po 1000 recenzija. Labela svake recenzije označava da li je recenzije bila pozitivna ili ne, što je jednostavniji problem od onog za koji je gore navedena tabela (gore je problem bio predvideti tačnu ocenu 1-5).

3.3 Rezultati treninga

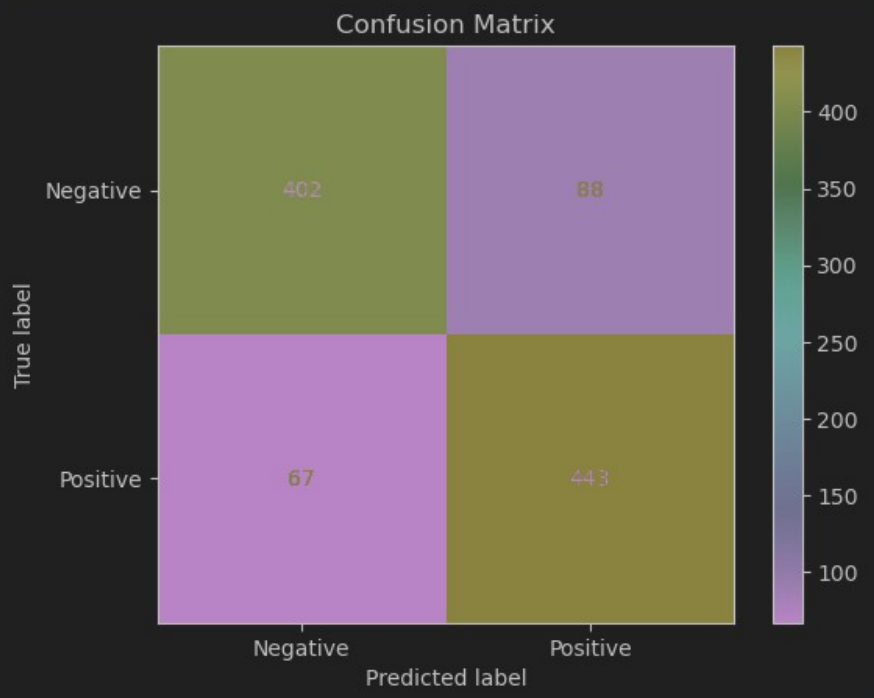
Kao što možemo da vidimo sa slike model je dosta jednostavan i već nakon desete epohe počinju fluktuacije u gubitku. To nam donekle sugeriše da bi smanjivanjem parametra learning rate mogli dodatno da optimizujemo konvergenciju, ali takodje i da bi ovako jednostavan model mogao lako da overfittuje.



Iz matrica konfuzije kao i metrika prikazanih na naredne dve slike vidmo da je model zaista overfittovao u nekoj meri zato što se bolje pokazuje na podacima koje je već video. Takodje iz f1 score-a vidimo i da model nije samo dobar u prepoznavanju jedne brojnije klase nego da je naučio da prepozna obe klase u većini slučajeva (~84.5%).

```
evaluate_model(model=trained_model, dataloader=test_dataloader, criterion=criterion, device=device)
✓ [17] 10s 757ms
```

```
Accuracy: 0.8450, Loss: 0.4431, F1 Score: 0.8511
```




```
evaluate_model(model=trained_model, dataloader=train_dataloader, criterion=criterion, device=device)
✓ [18] 1m 36s
```

```
Accuracy: 0.9724, Loss: 0.0889, F1 Score: 0.9723
```

3.4 Poređenje sa literaturom

Originalni model je imao veće vektorske reprezentacije reči i veća skrivena stanja pa je mogao da „uhvati“ veći broj informacija o samim rečima i o njihovim kontekstualnim odnosima. Ipak sa druge strane vidimo da je njegov procenat pogodaka manji od svog „mladjeg brata“, ali razlog tome je kao što je već navedeno, kompleksniji zadatak, dakle predvidjanje tačne ocene 1-5 naspram predvidjanja samo binarno odredjenog sentimenta recenzije.

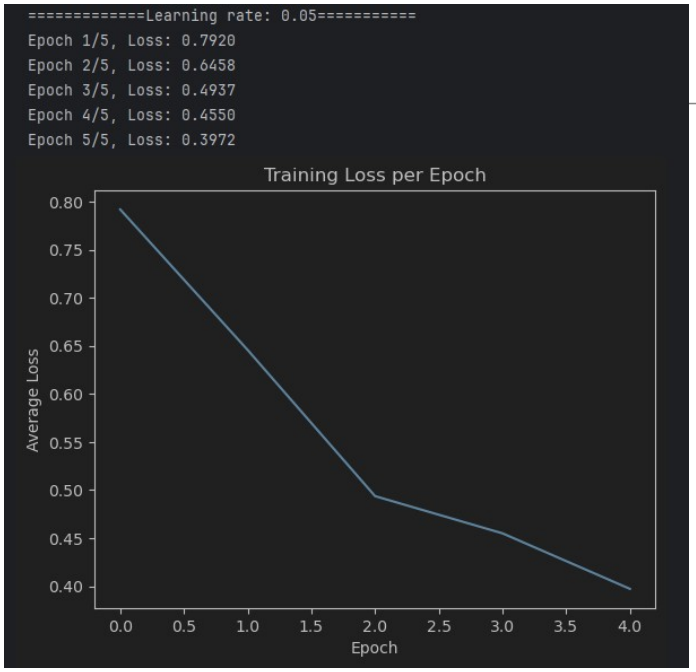
---

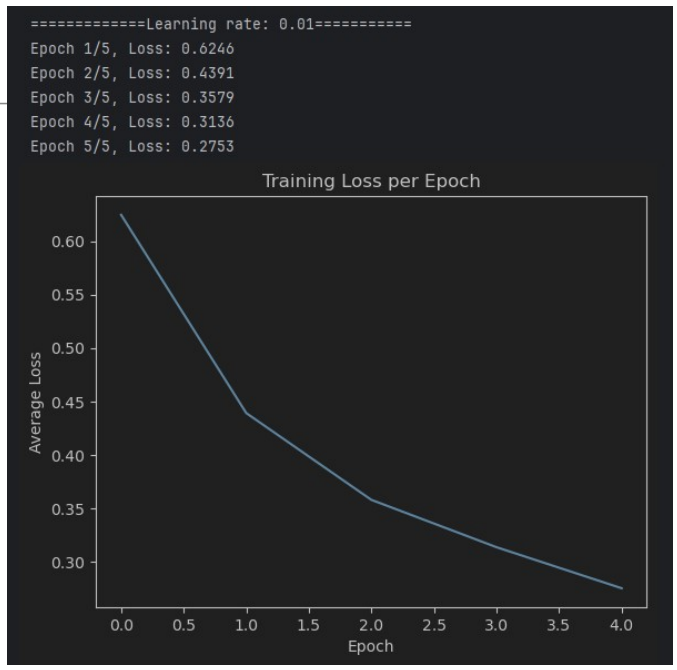
4. Zaključak

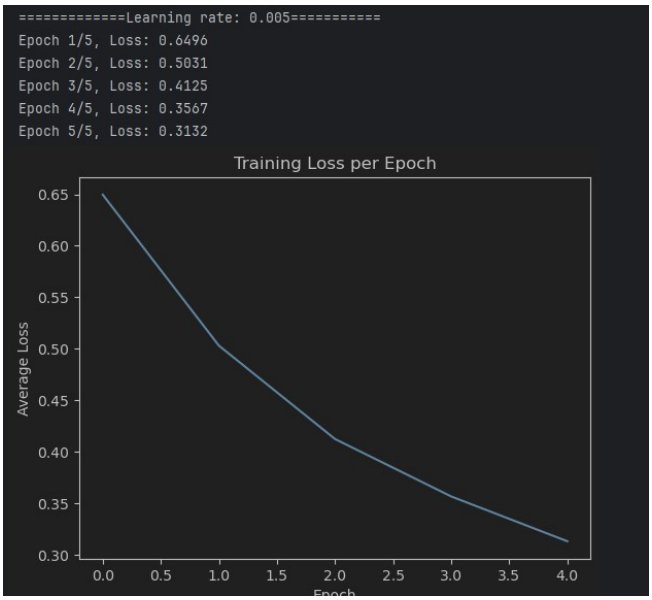
U ovom radu smo implementirali i testirali **Hierarchical Attention Network (HAN)** za klasifikaciju sentimenta na recenzijama proizvoda. Cilj rada bio je da reprodukujemo pristup iz rada Yang et al., 2016, i analiziramo performanse HAN modela na našem dataset-u.

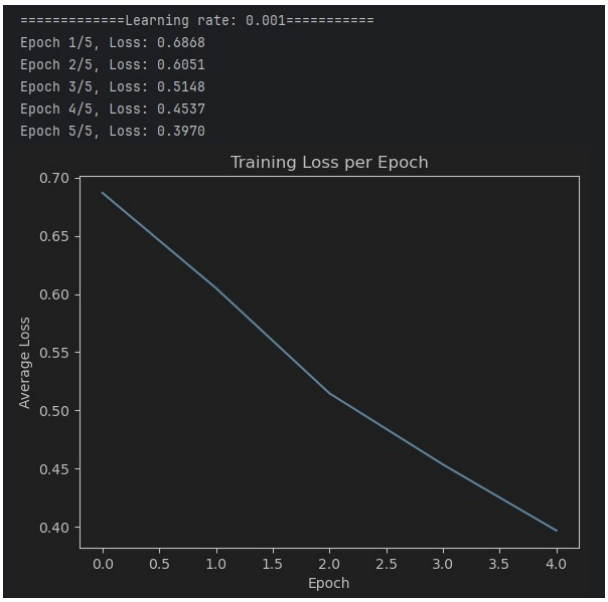
Eksperimentalni rezultati su pokazali da HAN može efikasno da procesuira dokumente koji imaju hijerarhijsku strukturu (rečenice i reči) i da iskoristi mehanizam pažnje za identifikaciju ključnih delova teksta. Evaluacija na test skupu je pokazala zadovoljavajuće rezultate, uprkos overfittovanju.

Pored toga, važno je napomenuti uticaj hiperparametara na brzinu konvergencije kao i na tačnost modela. U ovom projektu, nažalost tačnost je morala biti žrtvovana u nekoj meri radi brzine treniranja, ali time je pokazano kako parametri deluju na brzinu. Na primer parametar batching_size direktno utiče i na brzinu konvergencije ali i na brzinu samog izvršavanja trening petlje. Dakle veće vrednosti ovog parametra daju precizniji gradijent ali samim tim i manje koraka po epohi što usporava konvergenciju gubitka ali ubrzava izvršavanje trening petlje (manje puta se računa gradijent propagacijom unazad). Što se tiče learning rate parametra, za njega možemo sa slika ispod da vidimo kako utiče na konvergenciju gubitka.









Važno je i napomenuti da overfittovanje koje se dogodilo može da se izbegne na različite načine. Najočigledniji je skratiti sam trening, ali pored toga isti efekat se može postići i korišćenjem većeg broja trening podataka za isti ili veći broj epoha (u zavisnosti koliko podataka se obradjuje).

5. Literatura

- Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical Attention Networks for Document Classification.
- Kumar, Sandeep. "GRU: Recurrent Neural Networks – A Smart Way to Predict Sequences in Python." *Towards Data Science*, 20. feb. 2020, https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6.

Napomena: Za implementaciju GRU i HAN modela korišćeni su tutorijali i zvanična dokumentacija PyTorch biblioteke, dok su podaci preuzeti sa Kaggle platforme