**Sveučilište Josipa Juraja Strossmayera u Osijeku**

**Fakultet elektrotehnike, računarstva i informacijskih tehnologija**

**Raspoznavanje uzoraka i strojno učenje**

**Predikcija žanra filma prema kratkoj priči**

Matej Džijan

# Uvod

Seminarski rad obuhvaća dohvaćanje podataka s IMDb-a, obradu istih te izradu modela za klasifikaciju filmova u žanrove prema zadanoj kratkoj priči. Za ulaz se koristi kratki sadržaj filma, te za izlaz se dobiva do 3 žanra koje model predvidi kao ispravne za zadanu priču.

Sam *dataset* je napravljen pomoću *scrappera* koji dohvaća podatke s IMDb-a. Konačni *dataset* sadržava 11579 filmova (uz duplikate) raznih žanrova. Svaki od filmova ima listu 1, 2 ili 3 žanra, kratku priču te ime, koje se ne koristi za klasifikaciju.

Ovakva klasifikacija, gdje svaki ulaz može imati nekoliko različitih klasa kao izlaz, se naziva *multilabel* klasifikacija za razliku od obične *multiclass* klasifikacije gdje ima nekoliko različitih klasa, ali svaki ulaz za izlaz ima točno jednu klasu. Dodatni problem ovog specifičnog zadatka je u tome što svaki ulaz nema jednak broj izlaza, nego broj izlaza (*labela*) varira. Zadatak je riješen korištenjem neuronskih mreža i biblioteke f*astText.*

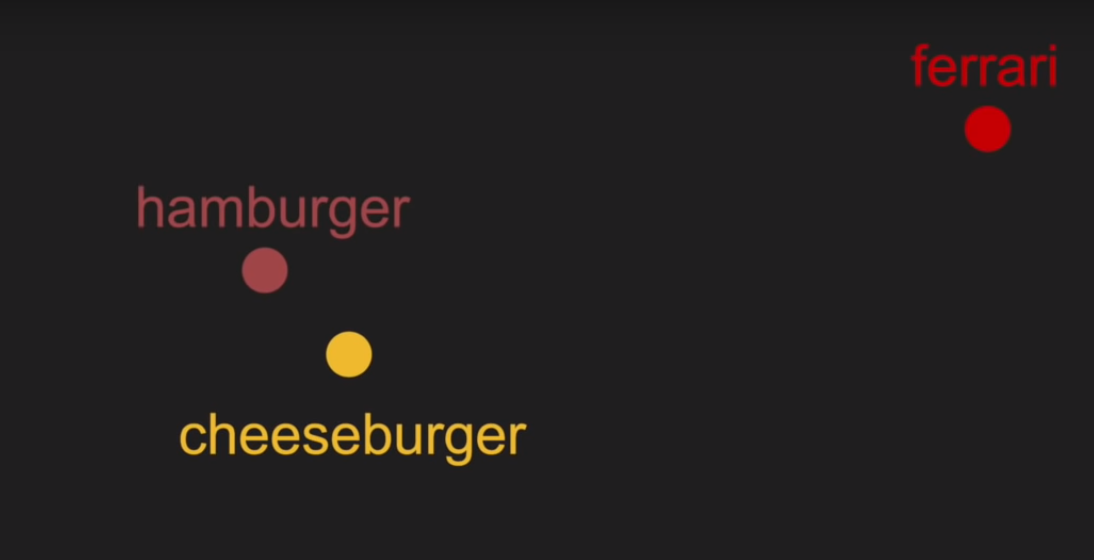
# fastText

fastText[1] je biblioteka za učinkovito učenje reprezentacija riječi i klasifikacija rečenica. Razvio ju je Facebook. 2016. postaje *open-source* i postaje široko prihvaćena zbog brzine treniranja i visokog performansa. Napisana je u C++-u i podržava višenitnost pri treniranju. fastText je omogućen na Linux i macOS operacijskim sustavima. Omogućava nadgledano i nenadgledano učenje. U ovom projektu je korišteno nadgledano učenje.

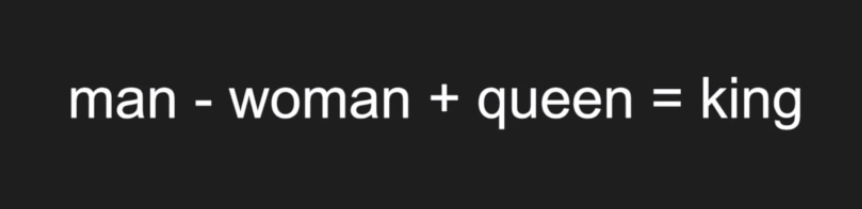
fastText model algoritam se sastoji od dva dijela, ugradnje riječi (*word embeddings*) i klasifikacije pomoću multinomijalne logističke regresije.

## *Word embeddings*

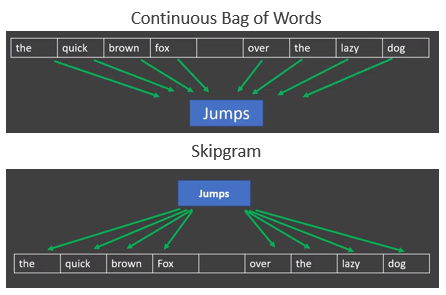
Ugradnja riječi je proces u kojem se riječi pretvaraju u vektore. Ovo se radi kako bi te riječ imale nekakav smisao računalu, računalo može puno toga raditi s vektorima, ali vrlo malo toga s običnim riječima. Ovim se korakom postiže da slične riječi budu bliže u nekakvom vektorskom prostoru, nego riječi koje nisu slične (slika 2.1.).

  
Slika 2.1. Prikaz riječi u 2D vektorskom prostoru

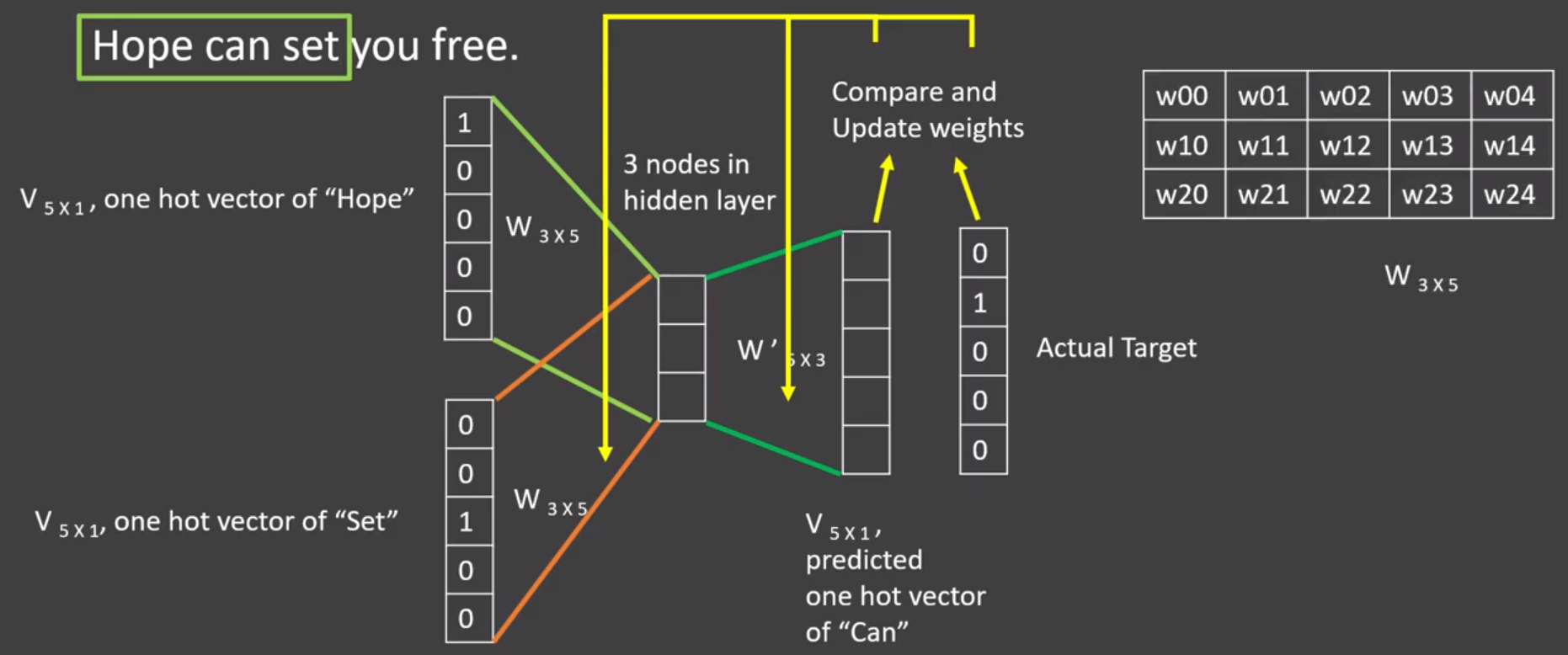
Ovakve riječi se mogu koristiti na razne načine kao vektori. Na slici 2.2. je simbolički prikazano što se može činiti s vektorskim reprezentacijama riječi, gdje imamo riječ *man* kojoj oduzimamo riječ *woman*, te razlici dodajemo riječ *queen*. Rezultat ove operacije približno daje riječ *king*. Drugim riječima, razlika između muškarca i žene je približno jednaka razlici između kralja i kraljice.

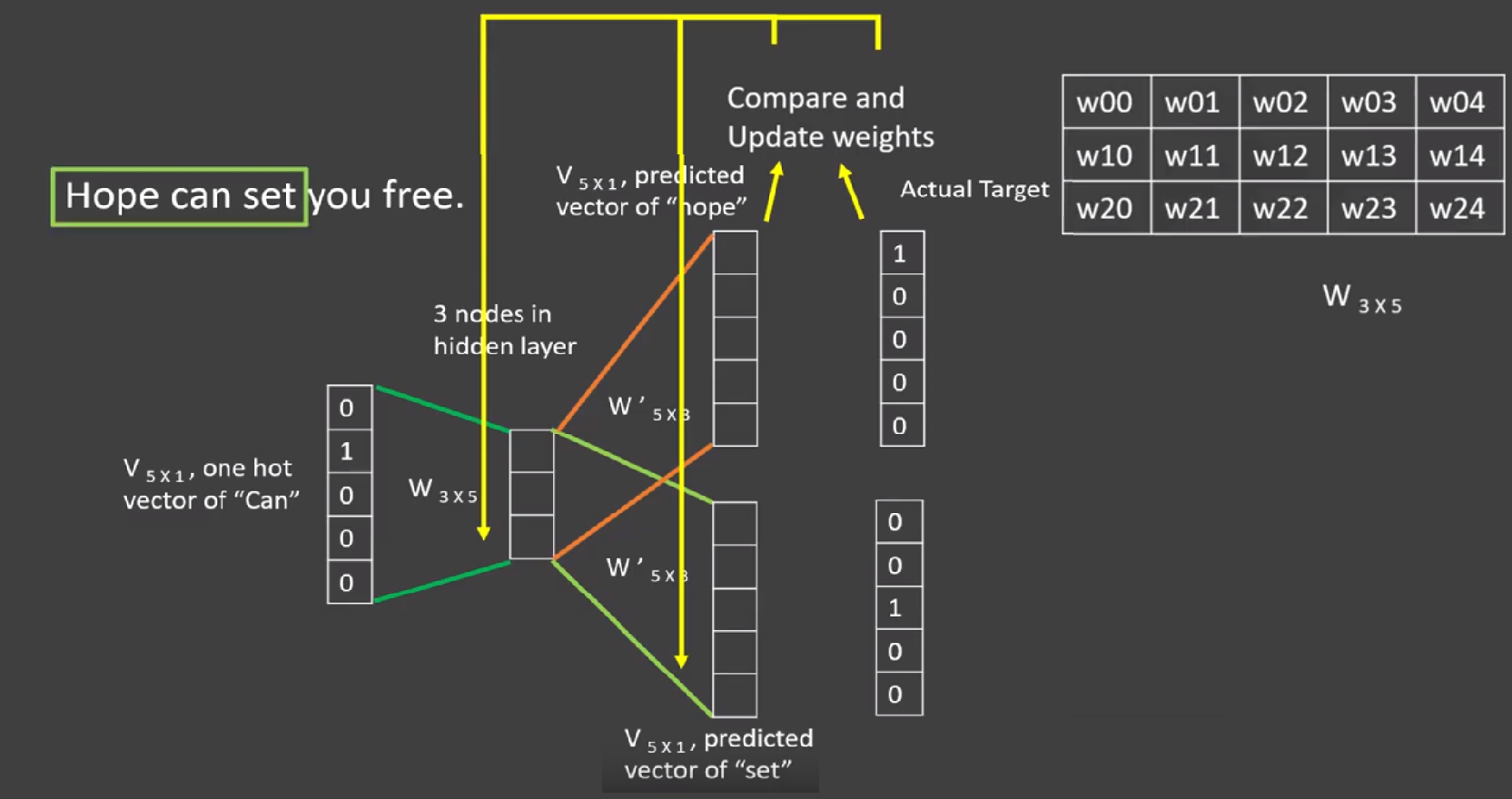
  
Slika 2.2. Simbolički prikaz matematičkih operacija s riječima

Sama ugradnja riječi se postiže korištenjem algoritma *word2vec*. Ovaj algoritam koristi neuronske mreže kako bi pretvorio riječi u pripadne vektore. Ima dva načina da ovo postigne, a to su korištenjem *Continuous Bag of Words* ili korištenjem *Skipgrama* (slika 2.3.).

  
Slika 2.3.

*CBOW* (slika 2.4.) predviđa željenu riječ iz konteksta, a *Skipgram* (slika 2.5.) predviđa kontekst iz zadane riječi.

  
Slika 2.4. Prikaz rada *CBOW*-a

  
Slika 2.5. Prikaz rada *Skipgrama*

Ovako istrenirana neuronska mreža daje vektor za svaku od riječi, a taj vektor su težine u skrivenom sloju. Moguće je odabrati broj čvorova u skrivenom sloju kako je prikazano u poglavlju 5. Također je moguće i odabrati veličinu kontekstnog prozora, odnosno broj riječi koji se gleda kao kontekst oko promatrane riječi. Dio riječi koji se ne pojavljuje dovoljno često ili se pojavljuje prečesto se odbacuje jer nemaju dovoljno utjecaja na krajnju klasifikaciju.

## Klasifikacija

Nakon što su riječi pretvorene u vektore, potrebno je klasificirati skup podataka u zadane klase (nadzirano učenje). fastText ovo vrši korištenjem multinomijalne logističke regresije. Multinomijalna logistička regresija je nadogradnja na običnu logističku regresiju. Omogućuje višeklasno klasifikaciju, za razliku od obične logističke regresije koja omogućuje samo binarnu klasifikaciju, nesortiranih klasa, odnosno jedna klasa nije „bolja“ ili „viša“ od neke druge klase kao što je slučaj za žanrove filmova. Ovo je moguće realizirati na nekoliko načina, a dva od njih su korištenjem skupa neovisnih binarnih regresija i log-linearni model.

Skup neovisnih binarnih regresija jednostavno koristi binarnu regresiju za svaku od klasa, gdje je jedna klasa te binarne regresije promatrana klasa, a druga klasa je sve ostalo. Na ovaj način se dobiju vjerojatnosti za svaku klasu prilikom klasifikacije.

## Dodatno

fastText dodatno omogućava korištenje nGrama. nGrami su dijelovi riječi određene duljine koji omogućuju dodatan kontekst sličnim riječima, te omogućavaju prepoznavanje sličnosti između riječi istog korijena i riječima sastavljenim od više riječi. Primjer nGrama riječi „apple“ s minimalnom duljinom nGrama 2, a maksimalnom 3, je kako slijedi: ap, app, pp, ppl, pl, ple, le.

# Dohvaćanje podataka

Podaci su dohvaćeni s IMDb-a korištenjem samostalno napravljenog *scrapera*. Dohvaćanje podataka se vrši u nekoliko koraka.

## Dohvaćanje poveznica na stranice filmova

Dohvaćanje poveznica:

1. **import** requests
2. **import** bs4
4. listOfGenres = ["Action", "Drama", "Comedy", "Crime",
5. "Mystery", "Adventure", "Animation", "Horror",
6. "War", "Documentary", "Sport", "Talk-Show",
7. "News", "Film-Noir", "Romance", "Short",
8. "Thriller", "Family", "Fantasy", "Sci-Fi",
9. "History", "Music", "Biography", "Music",
10. "Reality-TV", "Western", "Game-Show", "Adult"]
12. movies\_per\_genre = 1000
14. range\_num = int(movies\_per\_genre / 50)
15. starts = [1 + 50 \* i **for** i **in** range(range\_num)]
17. title\_ids = []
19. **for** genre **in** listOfGenres:
20. baseUrl = ("https://www.imdb.com/search/title?title\_type=feature&num\_votes=25000,&genres=" + genre).strip()
21. **for** start **in** starts:
22. listUrl = (baseUrl + "&view=simple&sort=user\_rating,desc&start=" + str(start) + "&ref\_=adv\_prv").strip()
23. movieIDs = []
24. movies = requests.get(listUrl)
25. movieshtml = movies.content
26. moviesSoup = bs4.BeautifulSoup(movieshtml, features="html.parser")
28. **for** f **in** moviesSoup.find\_all(class\_ = 'lister-item mode-simple'):
29. a = f.find('a', href=True)
30. title\_ids.append(a['href'])

Dohvaćanje poveznica na stranice filmova je odrađeno na način da se uzme svih 28 žanrova s IMDb-a, te se uzima najboljih 1000 filmova tog žanra (ako ih ima toliko). Neki od filmova se pojavljuju u više kategorija što ne predstavlja problem za dani zadatak jer predstavlja realnu sliku podataka. Da bi se dobilo najboljih 1000 filmova danog žanra, može se pristupati list po 50 filmova te za svaku od tih lista uzimati stranicu i tražiti sve izlistane filmove. Na ovaj način se dobije ekstenzija filmova ili title\_id. Zatim se dodaje osnovna poveznica. Na ovaj način je dohvaćeno 11579 filmova od kojih su neki pojavljuju više puta.

## Dohvaćanje podataka za pojedini film

Dohvaćanje podataka za jedan film:

1. **def** handle\_response(url):
2. response = requests.get(url)
3. html = response.content
5. soup = bs4.BeautifulSoup(html, features="html.parser")


9. div = soup.find(class\_ = 'title\_wrapper')
10. title = div.findChildren("h1" , recursive=False)[0]
11. title = str(list(title.strings)[0]).strip()
13. storyline = str(soup.find(class\_ = 'inline canwrap').find("span").string).strip()
15. f = soup.find(class\_ = 'subtext')
16. f = f.findAll("a")
17. genres = []
18. **for** a **in** f:
19. genres.append(a.string)
21. genres = genres[:len(genres) - 1]
23. **return** {
24. "title": title,
25. "genre": "".join([" " + str(x) **for** x **in** genres]).strip(),
26. "storyline": storyline
27. }

Nakon što se dohvati lista filmova, pristupa se svakoj od stranica te se uzimaju potrebni podaci, naslov, žanrovi, te kratka priča. Ovo je bio problem zbog količine podataka i ogromnog broja pristupanja stranici te je izazivalo brojne prekide zbog nemogućnosti daljnjeg spajanja na stranicu, stoga je *dataset* periodično spreman.

# Predobrada podataka

Nakon dohvaćanja podataka i pravljenja *dataseta* podatke je potrebno obraditi kako bi bili pogodni za korištenje s *fastText*-om.

Podjela podataka:

1. **from** sklearn.model\_selection **import** train\_test\_split
3. train, test = train\_test\_split(df, test\_size = 0.25)
5. train.to\_csv('dataset\_train.csv')
6. test.to\_csv('dataset\_test.csv')

Podaci se prvo dijele na *train* i *test* podatke u omjeru 0.75 : 0.25 kako bi se moglo ispravno testirati na podacima na kojima nije treniran model.

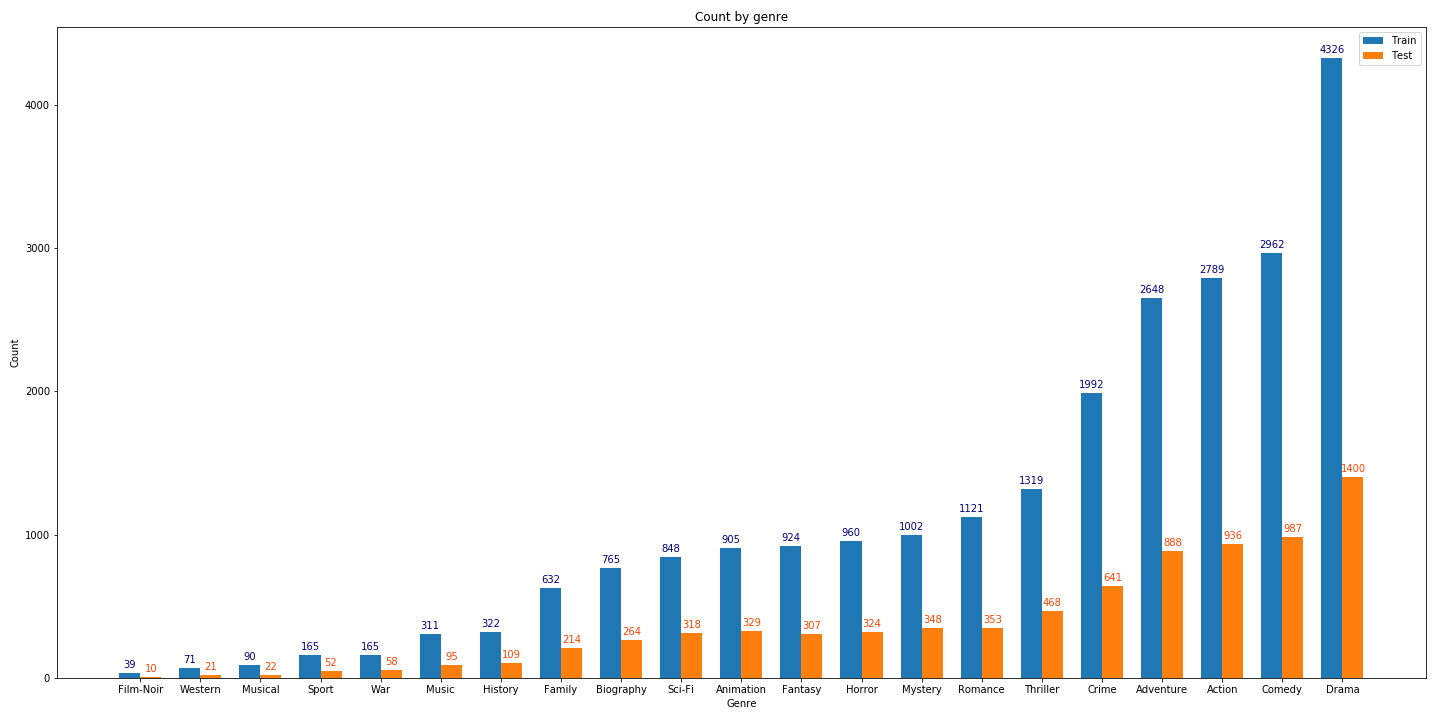
Obrada podataka za fastText:

1. train\_lines = []
3. **for** index, row **in** train.iterrows():
4. genres = row['genre'].split(" ")
5. **for** genre **in** genres:
6. line = '\_\_label\_\_' + genre + ' ' + row['storyline']
7. train\_lines.append(line)

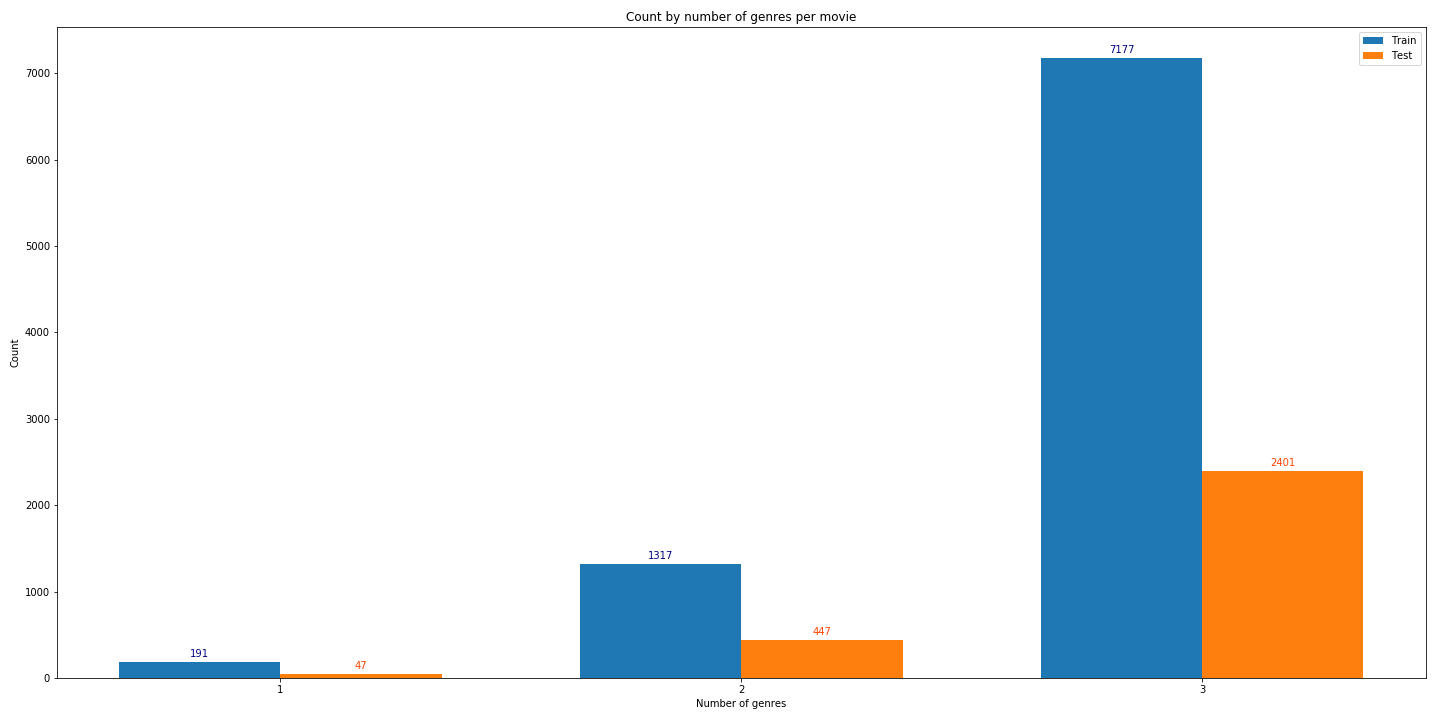
Za ulaz u algoritam, *fastText* prima tekstualnu datoteku koja je strukturirana na način da se svaki redak sastoji od *labela* i ulaza, odnosno kratke priče. Zadani idetifikator za *label* je '\_\_label\_\_' koji se dodaje na početak retka, zatim slijedi jedan žanr, razmak, te kratka priča. Za filmove koji imaju više žanrova, ovaj proces se ponavlja za svaki od žanrova.

  
Slika 4.1. Primjer retka sa žanrom *Thriller*

## Podaci

  
Slika 4.2. Broj filmova po žanru[[1]](#footnote-1)

Slika 4.2. prikazuje raspodjelu filmova po pojedinim žanrovima. Može se primijetiti da drame ima najviše, zatim komedije, akcije i avanture, a sve dalje je broj filmova po žanru sve manji. Skup podataka nije balansiran, ali nisu ni realni podaci, stoga je ovo prihvatljiv skup podataka.



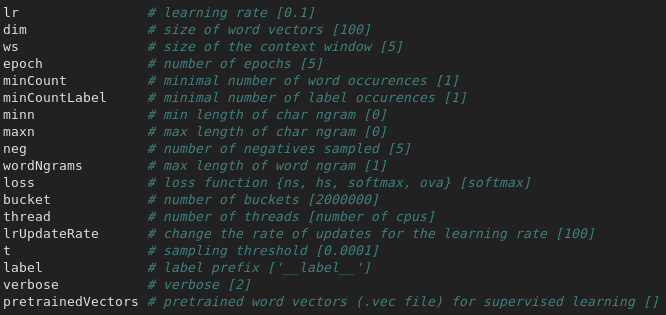
Slika 4.3. Broj žanrova po filmu

Iz slike 4.3. se može primijetiti da velika većina filmova iz danog skupa podataka ima tri zadana žanra, a manje ih ima dva ili jedan zadani žanr.

Ovako podijeljen skup podataka se zatim dalje predaje u fastText model za nadzirano učenje.

# Treniranje modela

Treniranje modela se vrši korištenjem funkcije *train\_surpervised* koja obavlja nadgledano učenje na zadanom *datasetu* uz određene hiperparametre.

  
Slika 5.1. Hiperparametri modela

*fastText* model omogućuje postavljanje raznih hiperparametara. U ovom projektu su postavljeni samo neki od njih.

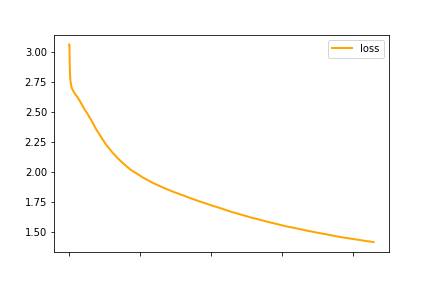
* lr – predstavlja *learning rate*, odnosno brzinu učenja neuronske mreže
* dim – dimenzija skrivenog sloja modela
* ws – veličina kontekstnog prozora, odnosno koliko riječi oko trenutne se promatra
* epoch – broj epoha treniranja modela
* minCount – predstavlja minimalan broj ponavljanja neke riječi da bi se dodala u konačni skup

Izmjenom hiperparametara se postižu razni rezultati. Povećavanjem *lr* se povećava utjecaj svakog kruga treniranja na neurone mreže. Ukoliko je *lr* premal, neće se doći do željenih rezultata u ograničenom broju epoha. Povećanjem broja epoha se znatno povećava vrijeme treniranja modela.

Tablica 5.1. Odabrani parametri modela

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **lr** | **dim** | **ws** | **epoch** | **minCount** |
| 0.04 | 100 | 5 | 200 | 4 |

Hiperparametri u tablici 5.1. su odabrani eksperimentalno jer su imali najbolje rezultate od testiranih parametara.

  
Slika 5.2. Treniranje neuronske mreže

Na slici 5.2. je prikazano opadanje *lossa* kroz treniranje neuronske mreže. Sam model ne prikazuje treniranje kroz pojedine epohe, nego samo kao postotak. Također, kroz cijeli proces treniranja, *learning rate* se smanjuje linearno.

# Validacija

Problem validacije kod *multilabel* klasifikacije je u tome što se ne mogu koristiti metode validacije kao kod klasičnih klasifikacija jer svaki izlaz može imati više klasa od kojih su neke točne, a neke netočne. Može se uzeti da samo one koje su u potpunosti točne su točne, ali to ne bi dalo realnu evaluaciju modela. Za *multilabel* klasifikaciju, najbolji pokazatelj koliko je model dobar je *Hamming Loss*[2]. *Hamming loss* prikazuje koliko puta se u prosjeku krivo predviđa klasa. Uzima u obzir i broj krivo predviđenih klasa i broj klasa koje nisu predviđene, a trebale su biti. Normalizira se preko ukupnog broja klasa i ukupnog broja primjera.

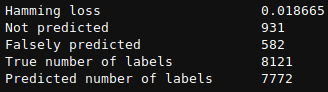


Slika 6.1. Formula za izračun *Hamming Loss*

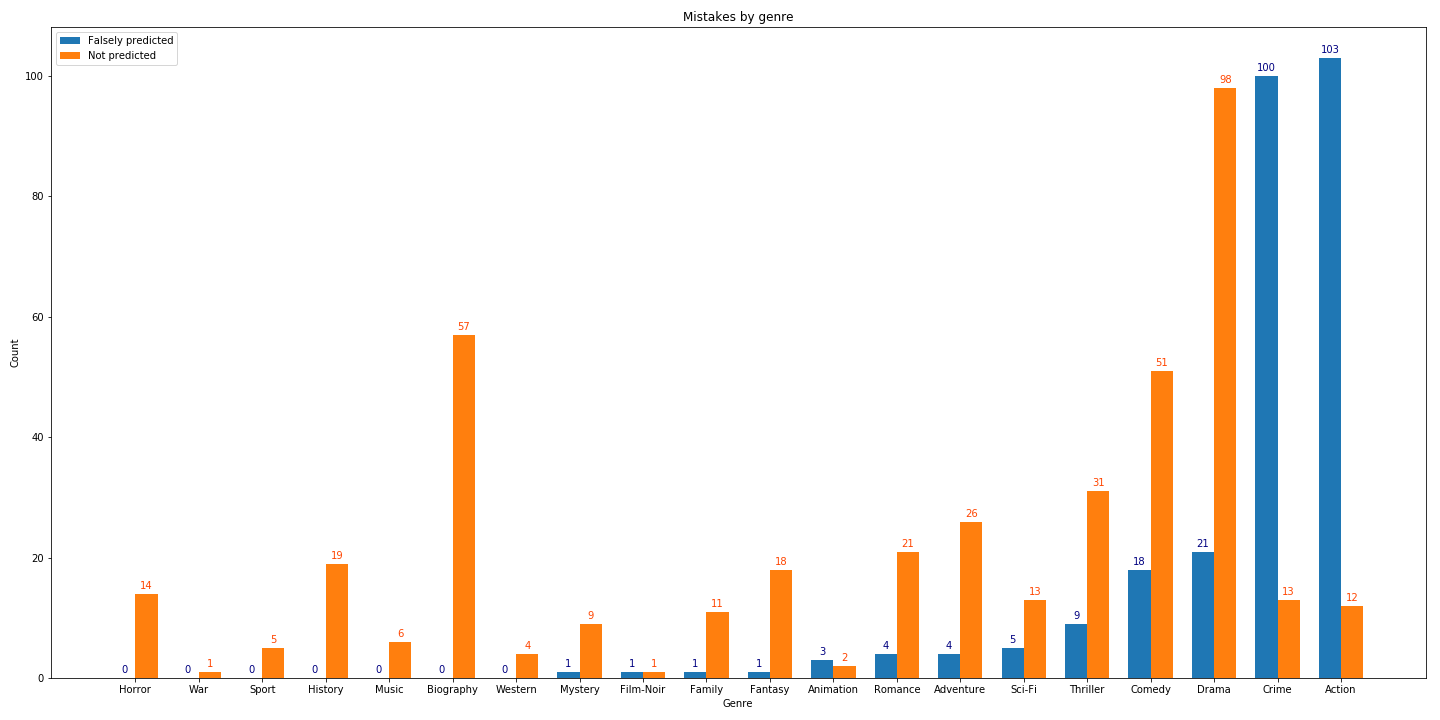
* |N| - ukupan broj primjera
* |L| - ukupan broj klasa

Xor funkcija upravo traži broj klasa koje su neispravno predviđene ili nisu predviđene, a trebale su biti predviđene. U idealnom slučaju, *hamming loss* je jednak 0.

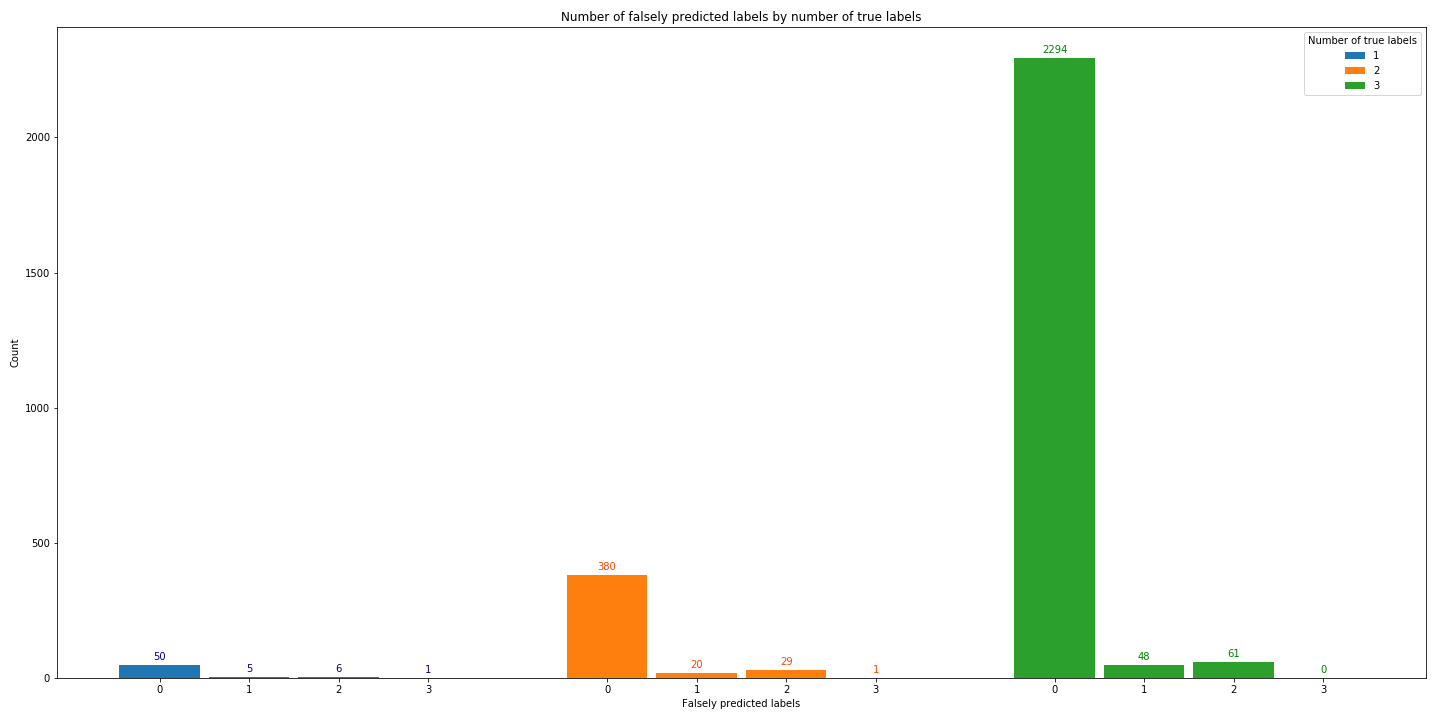
U prilogu je prikazana implementacija računanja *hamming lossa.* Funkcija, osim vrijednosti *hl*, vraća i broj *labela* koji nisu predviđeni, a trebali su biti, broj neispravno predviđenih *labela*, ukupan broj stvarnih *labela*, te ukupan broj predviđenih *labela*. Za željenu vjerojatnost žanra, eksperimentalno je odabrana vrijednost 0.140.

  
Slika 6.2. Rezultati modela

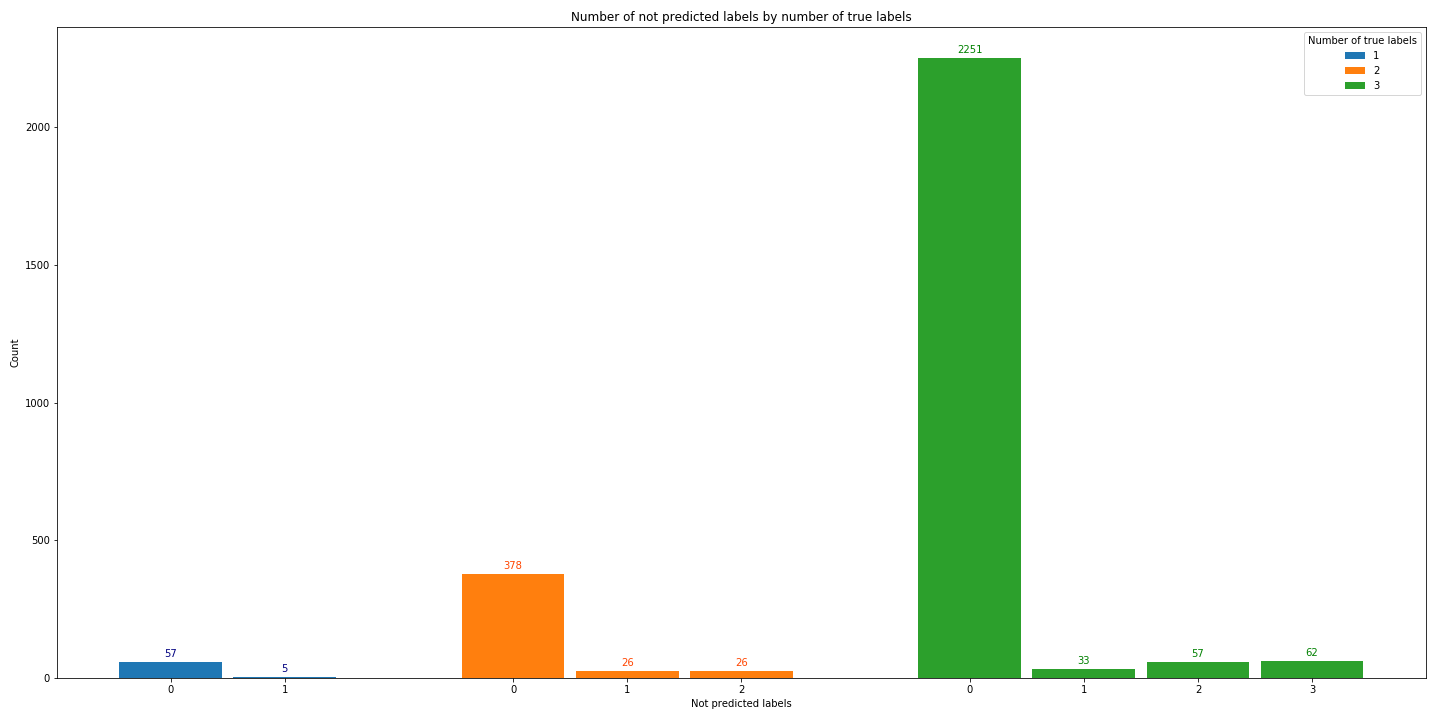
Slika 6.2. prikazuje rezultate konačnog modela, uključujući hamming loss, broj ne predviđenih *labela* koji su trebali biti predviđeni, broj *labela* koji su neispravno predviđeni, te ukupan broj stvarnih *labela* i ukupan broj predviđenih *labela*.

  
Slika 6.3.

Na slici 6.3. su prikazane pogreške po pojedinom žanru gdje su plavom bojom označeni krivo predviđeni žanrovi, a narančastom žanrovi koji su trebali biti predviđeni, ali nisu.

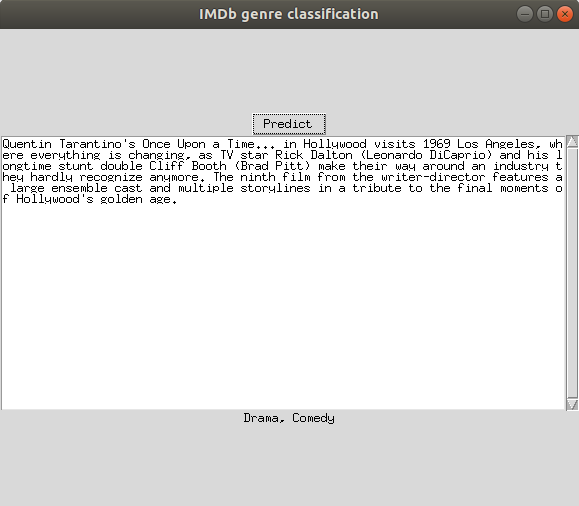
  
Slika 6.4.

Na slici 6.4. je prikaz broja neispravno predviđenih žanrova po stvarnom broju žanrova pojedinom filma. Plavom bojom su označeni filmovi koji imaju samo jedan žanr, narančastom oni koji imaju dva žanra, te oni koji imaju tri žanra su označeni zelenom bojom. Na apscisi je označen broj neispravno predviđenih žanrova za svaku skupinu, gdje 0 označava da nijedan žanr nije neispravno predviđen (ne znači da su svi predviđeni ispravno), a 3 označava da su tri žanra neispravno predviđena.

  
Slika 6.5.

Na slici 6.5. je prikaz sličan onome na slici 6.4., ali ovdje je prikazan broj žanrova koji nisu predviđeni, a trebali su biti, također grupirano po stvarnom broju žanrova filma. U ovom slučaju broj 0 uistinu znači da su svi žanrovi ispravno predviđeni, odnosno da je svaki žanr koji je trebalo predvidjeti, uistinu i je predviđen.

## Korisničko sučelje i novi podaci

  
Slika 6.6.

Na slici 6.6. je prikazana jednostavna korisnička aplikacija koja omogućava unos kratke priče filma, te ispisuje predviđene žanrove odvojene zarezom. Korištenjem ove aplikacije, uneseni su filmovi koji nisu dio skupa podataka na kojima je treniran model jer su izašli nakon prikupljanja podataka. Stvarni žanrovi filmova, te rezultati koje predviđa model su prikazani u tablici 6.1., redoslijed žanrova nije bitan.

Tablica 6.1. Predviđanje žanrova novih filmova

|  |  |  |
| --- | --- | --- |
| Naziv filma | Stvarni žanrovi | Predviđeni žanrovi |
| Once Upon a Time... in Hollywood | Comedy, Drama | Drama, Comedy |
| Edie | Drama | Drama |
| Angel Has Fallen | Action, Thriller | Action |
| Scary Stories to Tell in the Dark | Horror, Mystery, Thriller | Horror, Thriller |
| Spider-Man: Far from Home | Action, Adventure, Sci-Fi | Adventure, Animation |
| Overcomer | Drama, Sport | Drama, Comedy, Biography |

# Zaključak

Problem *multilabel* klasifikacije predstavlja problem strojnog učenja drugačiji od obične *multiclass* klasifikacije. Iako se koriste relativno slične metode kao što su *bag-of-words* i neuronske mreže, *multilabel* klasifikacija traži drugačiju obradu podataka i rezultata.

Postignuti rezultati su vrlo dobri, i znatno bolji od očekivanih zbog vrlo malog skupa podataka za treniranje i kompliciranog problema. Rezultati bi se mogli unaprijediti boljim prikupljanjem podataka. Najveći problemi pri izradi projekta su bili u zahtjevima na sklopovlje i čak zahtjevima na programsku podršku.

# Literatura

[1] <https://fasttext.cc/>

[2] <https://www.researchgate.net/profile/Mohammad_Sorower/publication/266888594_A_Literature_Survey_on_Algorithms_for_Multi-label_Learning/links/58d1864392851cf4f8f4b72a/A-Literature-Survey-on-Algorithms-for-Multi-label-Learning.pdf>

Ostalo:

<https://medium.com/@mariamestre/fasttext-stepping-through-the-code-259996d6ebc4>

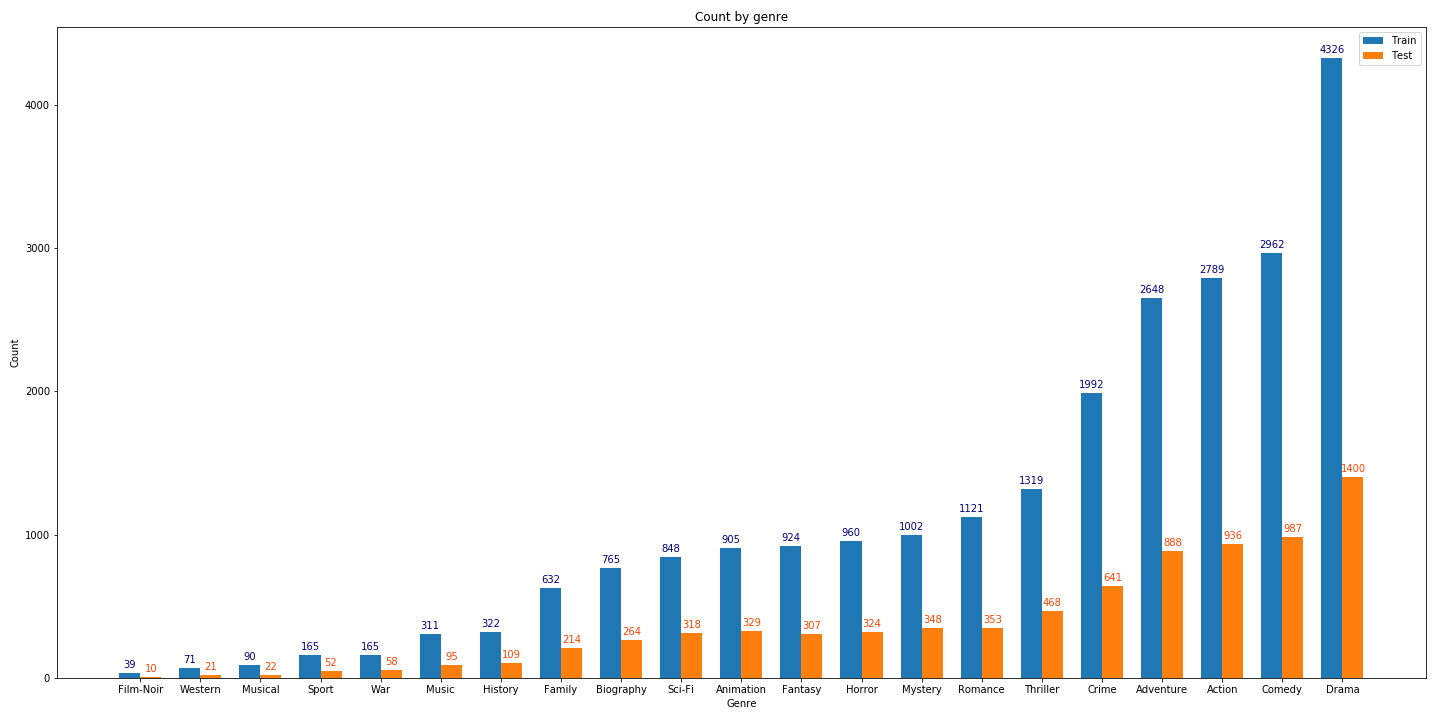
<https://towardsdatascience.com/fasttext-under-the-hood-11efc57b2b3>

<https://stats.stackexchange.com/questions/12702/what-are-the-measure-for-accuracy-of-multilabel-data>

<https://pypi.org/project/fasttext/>

# Prilog

Sav kod i podaci se nalaze u direktoriju projekta, a mogu se i naći na gitlab-u na <https://gitlab.com/dzijo/imdb-genre-classification>.



1. Prilog [↑](#footnote-ref-1)