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

1. 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 izlala, nego broj izlaza (*labela*) varira. Zadatak je riješen korištenjem neuronskih mreža i biblioteke f*astText.*

2. fastText

fastText 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 je jednostavna neuronska mreža s jednim slojem. *Bag-of-words* reprezentacija teksta se prvo unosi u *lookup* sloj, gdje se preuzimaju uležištenja (*embeddings*) za svaku pojedinu riječ. Zatim se uzima prosjek uležištenja riječi kako bi se dobilo jedno prosječno uležištenje za cijeli tekst. Na skrivenom sloju završavama s *n_words* * *dim* brojem parametara, gdje je *dim* veličinja uležištenja, a *n_words* veličina rječnika. Nakon usrednjavanja, imamo samo jedan vektor koji se zatim unosi u linearni klasifikator: primjenjujemo softmax preko linearne transformacije izlaza ulaznog sloja. Linearna transformacija je matrica veličine *dim* * *n_output,* gdje je *n_output* broj izlaznih klasa.

3. Dohvaćanje podataka

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

3.1.Dohvaćanje poveznica na stranice filmova

```
import requests
import bs4

listOfGenres = ["Action", "Drama", "Comedy", "Crime",
                "Mystery", "Adventure", "Animation", "Horror",
                "War", "Documentary", "Sport", "Talk-Show",
                "News", "Film-Noir", "Romance", "Short",
                "Thriller", "Family", "Fantasy", "Sci-Fi",
                "History", "Music", "Biography", "Music",
                "Reality-TV", "Western", "Game-Show", "Adult"]

movies_per_genre = 1000

range_num = int(movies_per_genre / 50)
starts = [1 + 50 * i for i in range(range_num)]

title_ids = []

for genre in listOfGenres:
    baseUrl = ("https://www.imdb.com/search/title?title_type=feature&num_votes=25000,&genres=" + genre).strip()
    for start in starts:
        listUrl = (baseUrl + "&view=simple&sort=user_rating,desc&start=" + str(start) + "&ref_=adv_prv").strip()
        movieIDs = []
        movies = requests.get(listUrl)
        movieshtml = movies.content
        moviesSoup = bs4.BeautifulSoup(movieshtml, features="html.parser")

        for f in moviesSoup.find_all(class_ = 'lister-item mode-simple'):
            a = f.find('a', href=True)
            title_ids.append(a['href'])
```

Slika 3.1. Dohvaćanje poveznica na stranice filmova

Dohvaćanje poveznica na stranice filmova je odrađeno na način da se uzme svih 24 žanra 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.

```
base_url = 'http://www.imdb.com'
urls = list(map(lambda title_id: base_url + title_id, title_ids))
```

Slika 3.2. Dodavanje osnove poveznice

3.2.Dohvaćanje podataka za pojedini film

```
def handle_response(url):
    response = requests.get(url)
    html = response.content

    soup = bs4.BeautifulSoup(html, features="html.parser")



    div = soup.find(class_ = 'title_wrapper')
    title = div.findChildren("h1" , recursive=False)[0]
    title = str(list(title.strings)[0]).strip()

    storyline = str(soup.find(class_ = 'inline canwrap').find("span").string).strip()

    f = soup.find(class_ = 'subtext')
    f = f.findAll("a")
    genres = []
    for a in f:
        genres.append(a.string)

    genres = genres[:len(genres) - 1]

    return {
        "title": title,
        "genre": "".join([" " + str(x) for x in genres]).strip(),
        "storyline": storyline
    }
```

Slika 3.3. Dohvaćanje podataka za jedan film

```
data_dict = []

for index, url in enumerate(urls):
    if(index > 11000):
        movie = handle_response(url)
        data_dict.append(movie)

        if(index % 1000 == 0 and index != 0):
            df_temp = pd.DataFrame(data_dict)
            df_temp.to_csv('dataset_at_index_{}.csv'.format(index))
```

Slika 3.4. Dohvaćanje podataka za sve filmove

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.

4. Predobrada podataka

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

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size = 0.25)

train.to_csv('dataset_train.csv')
test.to_csv('dataset_test.csv')
```

Slika 4.1. Podjela podataka

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.

```
train_lines = []

for index, row in train.iterrows():
    genres = row['genre'].split(" ")
    for genre in genres:
        line = '__label__' + genre + ' ' + row['storyline']
        train_lines.append(line)
```

Slika 4.2. Obrada podataka

Za ulaz u algoritam, *fastTest* prima tekstualnu datoteku koja je strukturirana na način da se svaki redak sastoji od *labela* i ulaza, odnosno kratke priče. Zadani indetifikator 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.

```
__label__Thriller In 1941, New York intellectual playwright Barton Fink comes to Hollywood to write a Wallace Beery wrestling picture. Staying in the eerie Hotel
Earle, Barton develops severe writer's block. His neighbor, jovial insurance salesman Charlie Meadows, tries to help, but Barton continues to struggle as a bizarre
sequence of events distracts him even further from his task.
```

Slika 4.3. Primjer retka sa žanrom *Thriller*

5. Treniranje modela

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

```
import fasttext

model = fasttext.train_supervised('dataset_train.txt', lr = 0.04, dim = 100, ws = 5, epoch = 200, minCount = 4)
```

Slika 5.1. Treniranje modela

```
lr                  # learning rate [0.1]
dim                 # size of word vectors [100]
ws                  # size of the context window [5]
epoch               # number of epochs [5]
minCount            # minimal number of word occurences [1]
minCountLabel       # minimal number of label occurences [1]
minn                # min length of char ngram [0]
maxn                # max length of char ngram [0]
neg                 # number of negatives sampled [5]
wordNgrams          # max length of word ngram [1]
loss                # loss function {ns, hs, softmax, ova} [softmax]
bucket              # number of buckets [2000000]
thread              # number of threads [number of cpus]
lrUpdateRate        # change the rate of updates for the learning rate [100]
t                   # sampling threshold [0.0001]
label               # label prefix ['__label__']
verbose             # verbose [2]
pretrainedVectors # pretrained word vectors (.vec file) for supervised learning []
```

Slika 5.2. Hiperparametri modela

fastText model omogućuje postavnjanje 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.

Hiperparametri sa slike 5.1. su odabrani eksperimentalno jer su imali najbolje rezultate od testiranih parametara.

6. 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*. *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.

$$\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \operatorname{xor}(y_{i,j}, z_{i,j})$$

Slika 6.1. Formula za izračun *Hamming Loss*

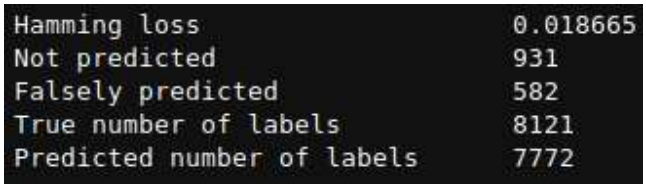
- |N| - ukupan broj primjera
- |L| - ukupan broj klasa

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.

```
def hamming_loss(test_df):
    not_predicted = 0
    falsely_predicted = 0
    number_of_true_labels = 0
    number_of_predicted_labels = 0

    for index, row in test_df.iterrows():
        predict = model.predict(row['storyline'],3)
        number_of_true_labels += len(row['genre'].split(" "))
        genres = []
        for i,p in enumerate(predict[1]):
            if(p > 0.140):
                genres.append(predict[0][i][9:])

        number_of_predicted_labels += len(genres)

        results = calculate_errors(genres, row['genre'].split(" "))
        not_predicted += results[0]
        falsely_predicted += results[1]

    return (not_predicted + falsely_predicted) / (28*len(test_df)), not_predicted, falsely_predicted, number_of_true_labels, number_of_predicted_labels

def calculate_errors(predicted, actual):
    not_predicted = 0
    falsely_predicted = 0

    for genre in actual:
        if genre not in predicted:
            not_predicted += 1
    for genre in predicted:
        if genre not in actual:
            falsely_predicted += 1

    return not_predicted, falsely_predicted
```

Slika 6.2. Implementacija *hamming lossa*

Na slici 6.2. prikazana je 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 6.3. Rezultati modela

Slika 6.3. prikazuje rezultate konačnog modela, uz nizak *hamming loss* od 0.018665.

7. Zaključak

Problem *multilabel* klasifikacije predstavlja problem strojnog učenja drugačiji od obične *multiclass* klasifikacije. Iako se koriste relativno slične motode 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. Najveći problemi pri izradi projekta su bili u zahtjevima na sklopovlje i čak zahtjevima na programsku podršku.

8. Literatura

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

https://towardsdatascience.com/fasttext-under-the-hood-11efc57b2b3?fbclid=IwAR3onpBT4FbpvNR2pV5rh3NkBt9C7Y839WTbmsHiIUh7p85OgcgkemxAwnE

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

https://fasttext.cc/

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