Neurónové siete – projekt Detekcia plagiátorstva

(riešenie problému predikcie/klasifikácie pomocou hlbokých neurónových sietí)

Riešiteľ: **Matúš Revický 1Im**, Ústav informatiky PF UPJŠ

Rok: 2020/2021 zimný semester

Formulácia problému:

Úlohou projektu je zostaviť detektor plagiátorstva, ktorý skúma textový súbor a vykonáva binárnu klasifikáciu a to označenie tohto súboru ako plagiátu alebo orignálu, na základe podobnosti s poskytnutým zdrojovým textom. Odhaľovanie plagiátu je aktívnou oblasťou výskumu. Úloha je netriviálna a rozdiely medzi parafrázovanými odpoveďami a pôvodnou prácou často nie sú také zrejmé.

Príprava dát:

Zadanie: *Dáta sú dostupné na stránke:* ***https://pan.webis.de/shared-tasks.html***

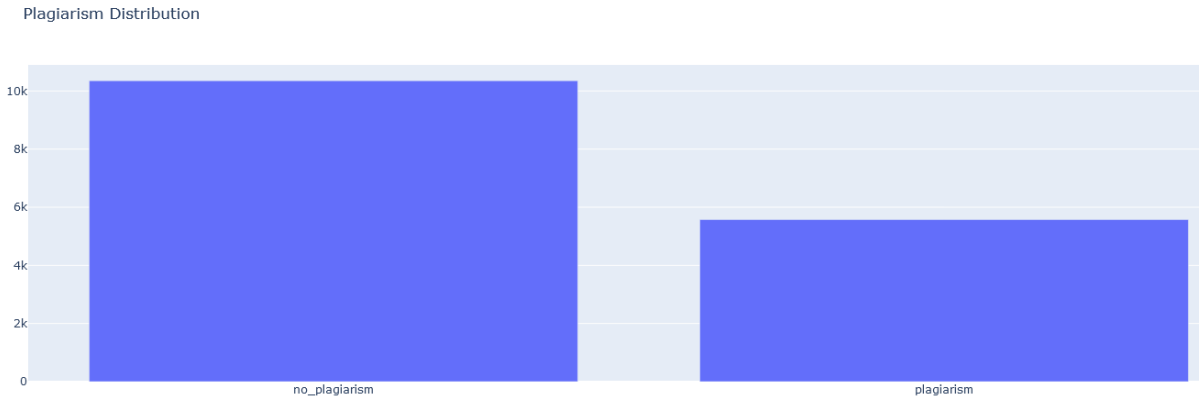
V úvode projektu bolo nevyhnutné nájsť si dataset (odporúčané bolo nájsť si dataset, ktorý obsahuje plagiáty zo stránky https://pan.webis.de/data.html)

Rozhodol som sa pre PAN plagiarism 2010 (PAN-PC-10). Tento korpus slúži na hodnotenie algoritmov automatickej detekcie plagiátorstva. Na výskumné účely je možné korpus použiť bezplatne.

PAN-PC-10 obsahuje dokumenty, do ktorých sa automaticky vkladá plagiát, ako aj dokumenty, do ktorých sa plagiát vkladá ručne. Prvé boli vyrobené pomocou počítačového programu, ktorý konštruuje plagiát podľa mnohých parametrov, zatiaľ čo druhé boli získané pomocou crowdsourcingu prostredníctvom Amazon Mechanical Turk.

Dataset PAN Plagiarism Corpus 2010 (PAN-PC-10)[1] obsahuje potencionálne anglické plagiáty (napr. suspicious-document00001.txt), ktoré majú dané z akých textov vychádzajú (napr. suspicious-document00001.xml). Dataset obsahuje 15 925 podozrivých dokumentov.

Zdrojové texty obsahujú okrem angličtiny aj inojazyčné slová.



Obr. 1 Distribúcia dokumentov medzi plagiáty a originály

[1] Dostupný na https://zenodo.org/record/3250123#.X6q4RFA1WUk

Predspracovanie dát:

Zadanie: *Predspracovanie dát: Z textov vytvoriť slovník a tabuľku obsahujúcu frekvenciu výskytov slov. Každému slovu priradiť farbu(nevýznamným slovám, ktoré definujete sami– spojky, predložky, a pod. -je možné priradiť rovnakú farbu). Navrhnúť rozdelenie textu na rovnaké úseky. K úsekom vytvoriť zafarbené obrázky. Tak vznikne dataset, na ktorom budú neurónové siete trénované a testované.*

- Rozhodol som sa spracovať texty nasledovne (*data_prepare_final.py*):
    1. Prevedenie textu na malé písmená
    2. Odstránenie diakritiky
    3. Odstránenie nepotrebných slov (pomocou vhodných slovníkov)[2]
    4. Zo slov som odstránil predložky a prípony pomocou algoritmu *Porter stemming*
- Pred spracovaním mal dataset 622 219 312 slov
- Celkovo po spracovaní obsahujú 295 442 061 slov v 15 925 podozrivých dokumentoch.
- Z takto spracovaných textov som vytvoril tabuľku obsahujúcu frekvenciu výskytov slov. (frequency_table2.csv) (data_coloring.py)
- Následne som všetkým jedinečným slovám priradil jedinečnú farbu vo formáte RGB[0-255, 0-255, 0-255] (words_to_color_dictionary.csv) (data_coloring.py)
- Po prehodnotení dostupných hardwarových prostriedkov som vybral z každého dokumentu prvých 40 000 slov. Kratšie som doplnil nulami, dlhšie orezal. Pole som previedol na maticu o rozmere 200x200. Každému slovu je priradená jedinečná farba. (data_coloring.py)

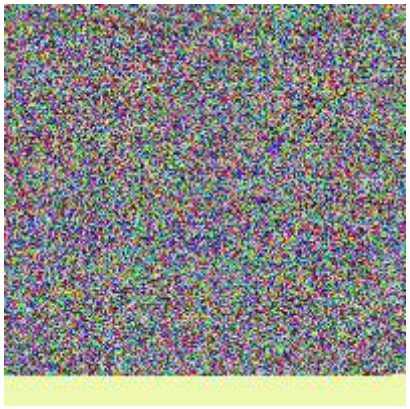
Pozn. Podrobnejšie informácie o použitom HW a dĺžke trvania sa nachádzajú v skriptoch *data_coloring.py, data_prepare_final.py*

Ukážka spracovania textu:
Pôvodný text: I have emphasized the from the discussion themselves of but that practically in the enforcement of the law, the legislative ideal EARLY is still pegged out.
Po spracovaní: emphas discuss practic enforc law legisl ideal earli still peg

Ukážka frekvenčnej tabuľky, slovníka s farbením, obrázku.

Slovo	Farba		
bibl	81	131	133
studi	229	142	117
life	162	85	153
paul	80	111	227
histor	135	66	166
construct	119	249	181
rev	184	244	238
henri	213	79	19
t	16	113	189

Slovo	Frekvencia
s	2 651 013
one	2 153 004
would	1 763 591
said	1 602 901
could	1 198 288
time	1 181 367
man	1 111 073
nt	1 007 002
like	976 235



- Týmto vznikol dataset, na ktorom budú neurónové siete trénované a testované.

[2] https://pythonspot.com/nltk-stop-words/

Príprava dvoch modelov neurónovej siete:

Zadanie: *Navrhnúť vhodné modely Kohonenovej a konvolučnej neurónovej siete na klasterizáciu a klasifikáciu obrázkov. Klasterizáciu použijeme na vytvorenie tried pre konvolučnú sieť.*

1. **pokus: použiť multilabel klasifikáciu**.
   Podozrivé dokumenty sú reprezentované ako obrázky a zdrojové dokumenty nie sú použité. Nastal problém, keďže počet zdrojových dokumentov je až 11 148 a toľko je aj label-ov. Tým by som získal nielen oznam či dokument je alebo nie je plagiát, ale aj z ktorých dokumentov bol kopírovaný. Label-y nie sú rovnomerne rozložené. Pri tréningu bol problém už s rozdelením na tréningovú a validačnú vzorku tak, aby tréningová a validačná vzorka obsahovala tie isté lable. Pokiaľ sa vyskytol vo validačnej sade label, ktorý nebol v tréningovej, tak učenie skončilo errorom.

2. **pokus**: **roztriediť iba na dve kategórie - plagiát, orginál.** Podozrivé dokumenty sú reprezentované ako obrázky a zdrojové dokumenty nie sú použité.

Ďalej sa budem venovať prípadu, keď **roztriedim iba na dve kategórie - plagiát, orginál.**

Kohonen/klasterizácia

Klastrovanie je možné vykonať pomocou rôznych techník, ako je K-means clustering, Mean Shift clustering, DB Scan clustering, Hierarchical clustering... Ja som sa rozhodol pre *K-means clustering.*

Dáta je potrebné zhlukovať podľa podobnosti. Podobnosť môže znamenať byť podobne vyzerajúcim obrázkom alebo mať podobnú veľkosť alebo podobnú distribúciu pixelov, podobné pozadie atď. Pre rôzne prípady použitia je nutné odvodiť konkrétny obrazový vektor. Vektor obrázku obsahujúci charakteristické črty obrázka sa bude líšiť od vektora obrázka s distribúciou pixelov.

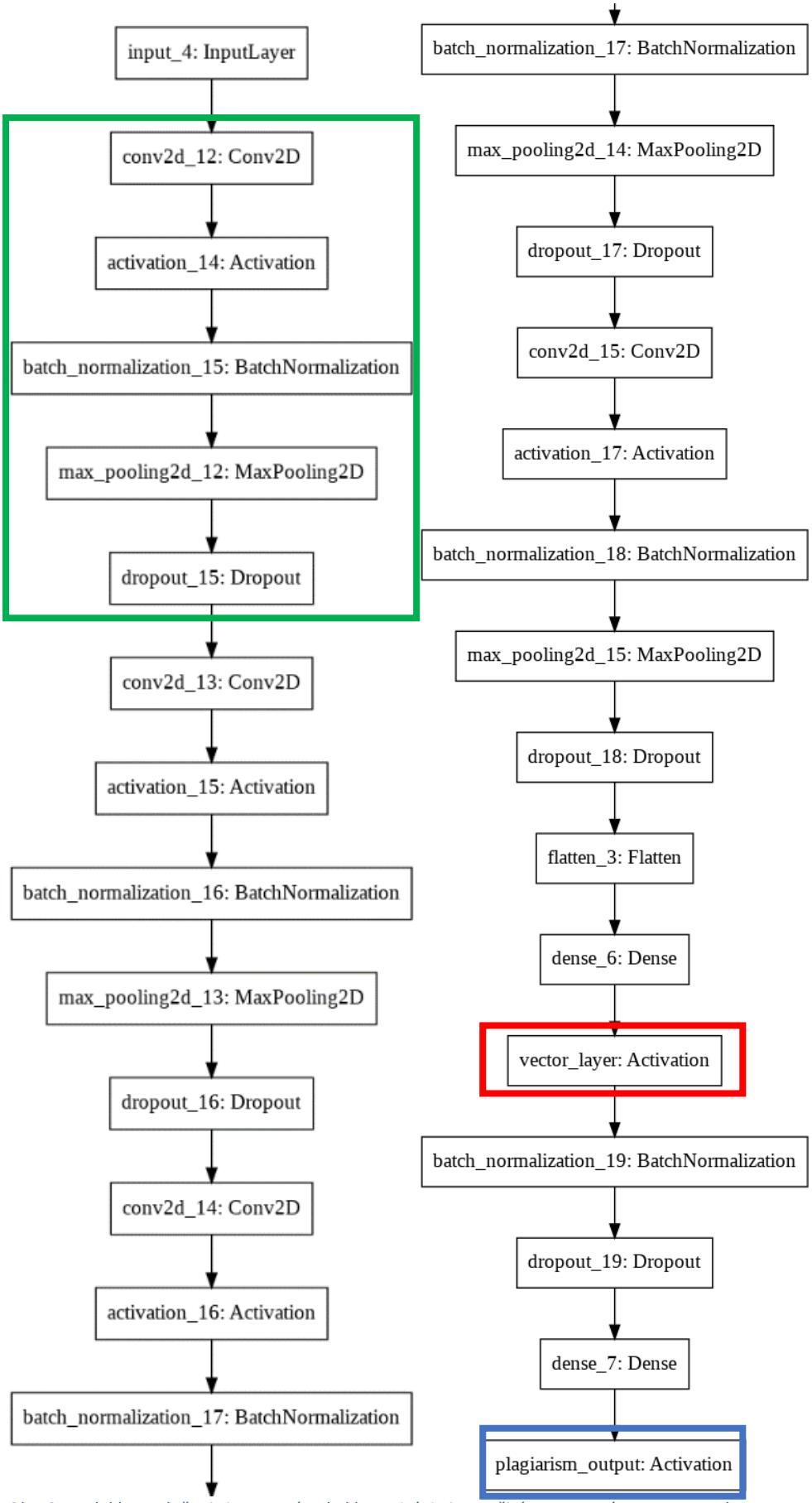
Obrázky plagiátov a originálov je potrebné zoskupiť. Na tento účel je vhodné odvodiť vektor s vyextrahovanými charakteristickými črtami z modelu konvolučnej siete (viď. Obr. 2, znázornené červeným). Ako vstup pre K-means clustering je možné použiť vrstvu vector_layer s veľkosťou 128.

Zhrnutie: Predpripravenie vektora do K-means clustering pomocou konvolučnej siete a nie len vo vloženie obrázka rovno ako vektor o rozmere 40 000. Počet centier je 2, a to plagiát, orginál.

Konvolučná sieť/klasifikácia

Navrhol som nasledovný model **konvolučnej siete:**

Štruktúra pre navrhnuté konvolučné vrstvy je založená na vrstvách v tvare: *Conv2D* s aktiváciou *ReLU*, po ktorej nasleduje vrstva *BatchNormalisation*, *MaxPooling* a nakoniec vrstva *Dropout* (viď. Obr. 2 znázornené zeleným). Po týchto vrstvách nasleduje posledná vrstva, ktorú tvorí *Dense* vrstva. Jedná sa o binárnu klasifikáciu a teda posledná vrstva obsahuje 2 výstupné neuróny (viď. Obr. 2 znázornené modrým).



Obr. 2 Model konvolučnej siete, v prípade klasterizácie je použitá vrstva s názvom vector_layer,

Layer (type)	Output Shape	Param #	Memory
input4 (InputLayer)	[(None, 200, 200, 3)]	0	200*200*3=120000
conv2d12 (Conv2D)	(None, 200, 200, 16)	448	200*200*16=640000
activation14 (Activation)	(None, 200, 200, 16)	0	200*200*16=640000
batchnormalization15 (Batc	(None, 200, 200, 16)	64	200*200*16=640000
maxpooling2d12 (MaxPooling	(None, 66, 66, 16)	0	66*66*16=69696
dropout15 (Dropout)	(None, 66, 66, 16)	0	66*66*16=69696
conv2d13 (Conv2D)	(None, 66, 66, 32)	4640	66*66*32=139392
activation15 (Activation)	(None, 66, 66, 32)	0	66*66*32=139392
batchnormalization16 (Batc	(None, 66, 66, 32)	128	66*66*32=139392
maxpooling2d13 (MaxPooling	(None, 33, 33, 32)	0	33*33*32=34848
dropout16 (Dropout)	(None, 33, 33, 32)	0	33*33*32=34848
conv2d14 (Conv2D)	(None, 33, 33, 64)	18496	33*33*64=69696
activation16 (Activation)	(None, 33, 33, 64)	0	33*33*64=69696
batchnormalization17 (Batc	(None, 33, 33, 64)	256	33*33*64=69696
maxpooling2d14 (MaxPooling	(None, 16, 16, 64)	0	16*16*64=16384
dropout17 (Dropout)	(None, 16, 16, 64)	0	16*16*64=16384
conv2d15 (Conv2D)	(None, 16, 16, 64)	36928	16*16*64=16384
activation17 (Activation)	(None, 16, 16, 64)	0	16*16*64=16384
batchnormalization18 (Batc	(None, 16, 16, 64)	256	16*16*64=16384
maxpooling2d15 (MaxPooling	(None, 8, 8, 64)	0	8*8*64=4096
dropout18 (Dropout)	(None, 8, 8, 64)	0	8*8*64=4096
flatten3 (Flatten)	(None, 4096)	0	4096
dense6 (Dense)	(None, 128)	524416	128
vectorlayer (Activation)	(None, 128)	0	128
batchnormalization19 (Batc	(None, 128)	512	128
dropout19 (Dropout)	(None, 128)	0	128
dense7 (Dense)	(None, 2)	258	2
plagiarismoutput (Activatio	(None, 2)	0	2
Total params: 586,402			
Trainable params: 585,794			
Non-trainable params: 608			

```
class PlagiarismOutputModel():

    def make_default_hidden_layers(self, inputs):
        # veľkosť filtrov 16, veľkosť kernelu je 3x3
        x = Conv2D(16, (3, 3), padding="same")(inputs)
        x = Activation("relu")(x)
        x = BatchNormalization(axis=-1)(x)
        x = MaxPooling2D(pool_size=(3, 3))(x)
        x = Dropout(0.1)(x)

        x = Conv2D(32, (3, 3), padding="same")(x)
        x = Activation("relu")(x)
        x = BatchNormalization(axis=-1)(x)
        x = MaxPooling2D(pool_size=(2, 2))(x)
        x = Dropout(0.1)(x)

        x = Conv2D(64, (3, 3), padding="same")(x)
        x = Activation("relu")(x)
        x = BatchNormalization(axis=-1)(x)
        x = MaxPooling2D(pool_size=(2, 2))(x)
        x = Dropout(0.1)(x)

        x = Conv2D(64, (3, 3), padding="same")(x)
        x = Activation("relu")(x)
        x = BatchNormalization(axis=-1)(x)
        x = MaxPooling2D(pool_size=(2, 2))(x)
        x = Dropout(0.1)(x)

        return x

    def build_plagiarism_branch(self, inputs, num_plagiarism=2):

        x = self.make_default_hidden_layers(inputs)

        x = Flatten()(x)
        x = Dense(128)(x)
        x = Activation("relu", name="vector_layer")(x)
        x = BatchNormalization()(x)
        x = Dropout(0.5)(x)
        x = Dense(num_plagiarism)(x)
        x = Activation("sigmoid", name="plagiarism_output")(x)

        return x
```

Model je naprogramovaný *v Kerase* a vzniká spojením `make_default_hidden_layers` a `build_plagiarism_branch`. Z kódu je už vidieť veľkosti filtrov a kernelu. Napr. `x = Conv2D(16, (3, 3), padding="same")(inputs)` znamená veľkosť filtrov 16, veľkosť kernelu je 3x3.

Natrénovať pripravené modely siete na pripravených dátach:

Zadanie: *Veľkosť datasetu treba prispôsobiť tak, aby ste to stihli spočítať vreálnom čase.*

Sieť som sa rozhodol trénovať na platforme ***Colaboratory***, ktorá umožňuje s istými obmedzeniami (12 GB RAM) prístup ku trénovaniu neurónovej siete pomocou grafických kariet zdarma. To mi umožnilo použiť všetkých 15 925 obrázkov o rozmere 200x200, ktoré som rozdelil medzi **trénovacie**, **validačné** a **testovacie** a to v pomere 7802:3345:4778. Počas trénovania sa používajú **trénovacie** a **validačné dáta.** Testovacie dáta sa použijú na otestovanie už natrénovanej siete.

Na trénovanie som použil: (detaily v)

- **učiaci pomer:** 0,0004, pričom sa postumne s ratúcimi epochami znižuje
- **loss funkcia:** binary_crossentropy
- **optimalizátor:** Adam
- **batch_size:** 64

Počas trénovania sa model ukladá po každej epoche. Model som trénoval na 150 epochách, pričom trvanie jednej epochy je cca. 30 sekúnd a teda celkový čas trénovania siete je 75 minút.

Ukážka poslednej epochy:

```
Epoch 150/150
121/121 [==============================] - ETA: 0s - loss: 0.2890 -
accuracy: 0.8693INFO:tensorflow:Assets written to:
/content/drive/MyDrive/model_checkpoint_plagiarism/assets
121/121 [==============================] - 29s 240ms/step - loss: 0.2890 -
accuracy: 0.8693 - val_loss: 5.2973 - val_accuracy: 0.4435
```

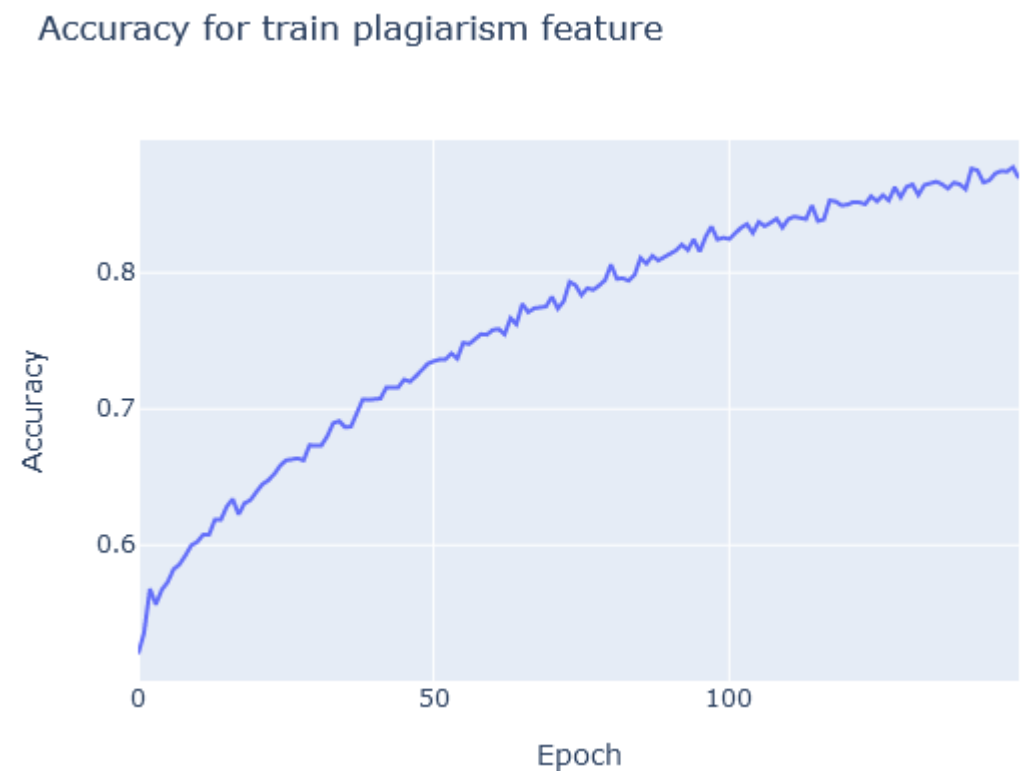
Overiť funkčnosť klasifikácie/klasterizácie na testovacích dátach, dosiahnuté výsledky oboch sietí vyhodnotiť a porovnať:

KONVOLUČNÁ SIEŤ

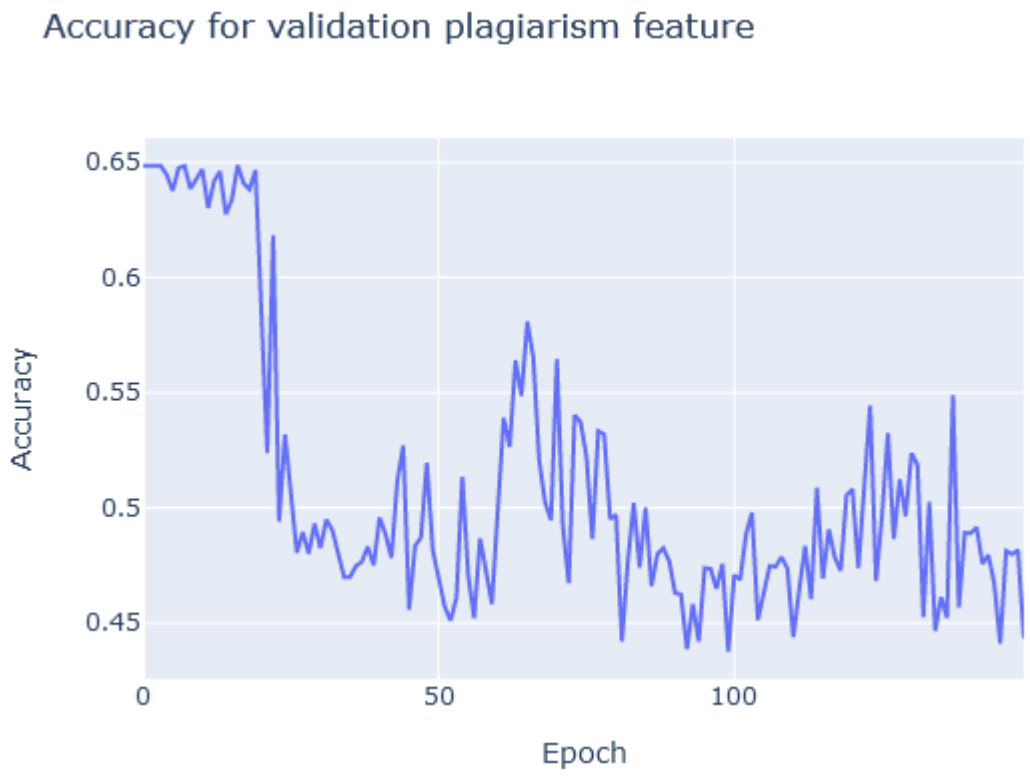
Na obr. 3, obr. 4, obr. 5 vidíme priebeh učenia **konvolučnej neurónovej siete**:

- Presnosť na trénovacích dátach: 0.8693
- Presnosť na validačných dátach: 0.4435
- Loss na trénovacích dátach: 0.2890
- Loss na validačných dátach: 5.2973

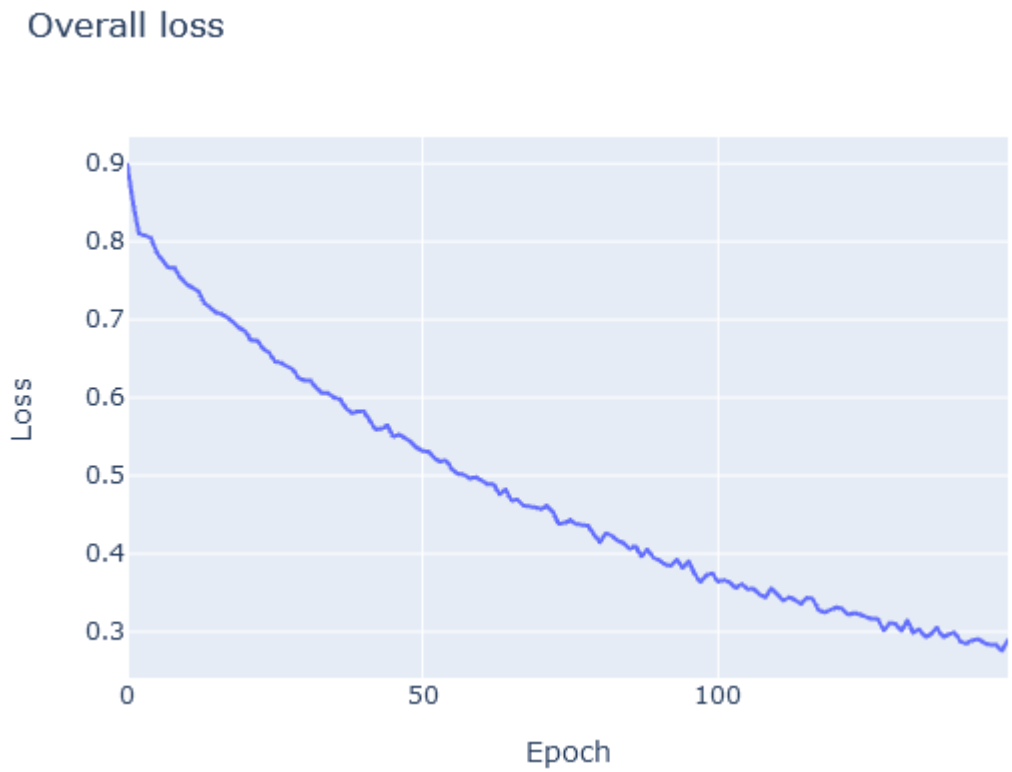
Loss funkcia sa má čo najviac priblížiť 0 a presnosť k 1. Z obr. 3, obr. 4, obr. 5 vidíme, že na trénovacích dátach sa sieť zlepšovala, ale na validačných sa dokonca oproti prvej epoche zhoršila o cca. 20%.



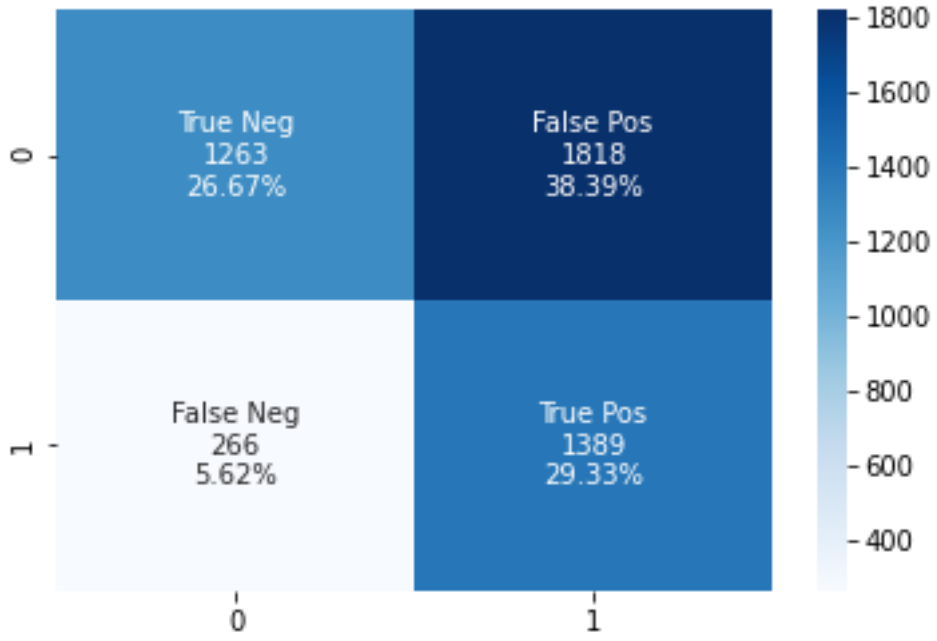
Obr. 3 Konvolučná sieť - Dosiahnutá presnosť na trénovacích dátach



Obr. 4 Konvolučná sieť - Dosiahnutá presnosť na validačných dátach



Obr. 5 Konvolučná sieť - Loss na trénovacích dátach



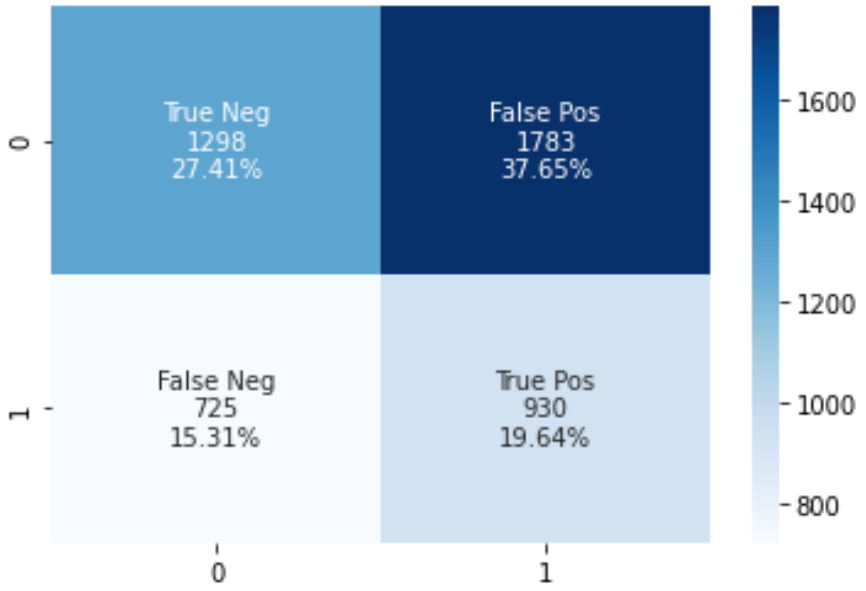
Obr. 6 Konvolučná sieť – confusion matrix

	precision	recall	f1-score	support
no_plagiarism	0.83	0.41	0.55	3081
plagiarism	0.43	0.84	0.57	1655
accuracy			0.56	4736
macro avg	0.63	0.62	0.56	4736
weighted avg	0.69	0.56	0.56	4736

Tab. 1 Konvolučná sieť – classification report

KLASIFIKÁCIA

Pri klasifikácií sa ako vstupný vektor do K-means použila vrstva vector_layer o veľkosti 128. Tento vektor by už mal obsahovať extrahované charakteristické črty obrázkov plagiátov.



Obr. 7 Klasifikácia - confusion matrix

	precision	recall	f1-score	support
no_plagiarism	0.64	0.42	0.51	3081
plagiarism	0.34	0.56	0.43	1655
accuracy			0.47	4736
macro avg	0.49	0.49	0.47	4736
weighted avg	0.54	0.47	0.48	4736

Tab. 2 Klasifikácia – classification report

ZHRNUTIE

Pri klasifikácií na testovacích dátach mala sieť horšie výsledky ako pri konvolučnej sieti.

- Ak daný **dokument je plagiát**, tak **konvolučná** sieť to odhalí s pravdepodobnosťou **43%** ale **klasifikačná** sieť s pravdepodobnosťou **34%**
- Ak daný **dokument nie je plagiát**, tak **konvolučná** sieť to odhalí s pravdepodobnosťou **83%** ale **klasifikačná** sieť s pravdepodobnosťou **64%**

Vážený priemer presnosti pri klasifikácií bol 54%
Vážený priemer presnosti pri konvolučnej sieti bol 69%

Zhrnutie :

Konfigurácia 0 :

- **učiaci pomer:** 0,0004, pričom sa postumne s ratúcimi epochami znižuje
- **loss funkcia:** binary_crossentropy
- **optimalizátor:** Adam
- **batch_size:** 64
- **počet epoch:** 150
- **colab notebook:** Plagiarism_adam_150_0.0004.ipynb

Výsledky na testovacích dátach:

K-means

	precision	recall	f1-score	support
no_plagiarism	0.64	0.42	0.51	3081
plagiarism	0.34	0.56	0.43	1655
accuracy			0.47	4736
macro avg	0.49	0.49	0.47	4736
weighted avg	0.54	0.47	0.48	4736

Konvolučná sieť

	precision	recall	f1-score	support
no_plagiarism	0.83	0.41	0.55	3081
plagiarism	0.43	0.84	0.57	1655
accuracy			0.56	4736
macro avg	0.63	0.62	0.56	4736
weighted avg	0.69	0.56	0.56	4736

Ďalšie testovacie konfigurácie:

Konfigurácia 1 :

- **učiaci pomer:** 0,04, pričom sa postumne s ratúcimi epochami znižuje
- **loss funkcia:** binary_crossentropy
- **optimalizátor:** Adam
- **batch_size:** 64
- **počet epoch:** 150
- **colab notebook:** Plagiarism_adam_150_0.04.ipynb

Výsledky na testovacích dátach:

K-means

	precision	recall	f1-score	support
no_plagiarism	0.64	1.00	0.78	3042
plagiarism	0.00	0.00	0.00	1694
accuracy			0.64	4736
macro avg	0.32	0.50	0.39	4736
weighted avg	0.41	0.64	0.50	4736

Konvolučná sieť

	precision	recall	f1-score	support
no_plagiarism	0.65	0.99	0.79	3042
plagiarism	0.77	0.04	0.08	1694
accuracy			0.65	4736
macro avg	0.71	0.52	0.43	4736
weighted avg	0.69	0.65	0.53	4736

Konfigurácia 2:

- **učiaci pomer:** 0,0004, pričom sa postumne s ratúcimi epochami znižuje
- **loss funkcia:** binary_crossentropy
- **optimalizátor:** Adam
- **batch_size:** 64
- **počet epoch:** 300
- **colab notebook:** Plagiarism_adam_300_0.0004.ipynb

Výsledky na testovacích dátach:

K-means

	precision	recall	f1-score	support
no_plagiarism	0.64	1.00	0.78	3042
plagiarism	0.00	0.00	0.00	1694
accuracy			0.64	4736
macro avg	0.32	0.50	0.39	4736
weighted avg	0.41	0.64	0.50	4736

Konvolučná sieť

	precision	recall	f1-score	support
no_plagiarism	0.65	0.99	0.79	3042
plagiarism	0.69	0.05	0.10	1694
accuracy			0.65	4736
macro avg	0.67	0.52	0.44	4736
weighted avg	0.67	0.65	0.54	4736

Konfigurácia 3:

- **učiaci pomer:** 0,0004, pričom sa postumne s ratúcimi epochami znižuje
- **loss funkcia:** binary_crossentropy
- **optimalizátor:** sgd
- **batch_size:** 64
- **počet epoch:** 150
- **colab notebook:** Plagiarism_sgd_150_0.0004.ipynb

Výsledky na testovacích dátach:

K-means

	precision	recall	f1-score	support
no_plagiarism	0.65	0.64	0.65	3050
plagiarism	0.36	0.37	0.37	1686
accuracy			0.55	4736
macro avg	0.51	0.51	0.51	4736
weighted avg	0.55	0.55	0.55	4736

Konvolučná sieť

	precision	recall	f1-score	support
no_plagiarism	0.64	1.00	0.78	3050
plagiarism	0.00	0.00	0.00	1686
accuracy			0.64	4736
macro avg	0.32	0.50	0.39	4736
weighted avg	0.41	0.64	0.50	4736

Konfigurácia 4:

- **učiaci pomer:** optimálny učiaci pomer pomocou lr_finder https://gist.github.com/jeremyjordan/ac0229abd4b2b7000aca1643e88e0f02
- **loss funkcia:** binary_crossentropy
- **optimalizátor:** adam
- **batch_size:** 64
- **počet epoch:** 5
- **colab notebook:** Plagiarism_cyclic_learning_rate.ipynb

Výsledky na testovacích dátach:

K-means

	precision	recall	f1-score	support
no_plagiarism	0.66	0.59	0.62	3095
plagiarism	0.35	0.41	0.38	1641
accuracy			0.53	4736
macro avg	0.50	0.50	0.50	4736
weighted avg	0.55	0.53	0.54	4736

Konvolučná sieť

	precision	recall	f1-score	support
no_plagiarism	0.65	1.00	0.79	3095
plagiarism	0.00	0.00	0.00	1641
accuracy			0.65	4736
macro avg	0.33	0.50	0.40	4736
weighted avg	0.43	0.65	0.52	4736

Pozn. Veľmi rýchle učenie, na dosiahnutie takýchto výsledkov stačilo 150 sekúnd trénovania.

Záver

Úlohou projektu bolo zostaviť detektor plagiátorstva, ktorý skúma textový súbor a vykonáva binárnu klasifikáciu a to označenie tohto súboru ako plagiátu alebo orignálu, na základe podobnosti s poskytnutým zdrojovým textom. Pričom zadanie požadovalo prevedenie textu na obrázky a následné použitie Konvolučnej siete. Odhaľovanie plagiátu je aktívnou oblasťou výskumu. Úloha sa ukázala ako netriviálna a rozdiely medzi parafrázovanými odpoveďami a pôvodnou prácou často nie sú také zrejmé.

- Predpracovanie datasetu (622 219 312 slov), ktorý mal po spracovaní približne 300 000 000 slov sa ukázalo ako časovo náročné.
- Platforma Colaboratory mi umožnila použiť vzorku dát netriviálnej veľkosti. Konkrétne 15 925 obrázkov plagiátov o rozmere 200x200, pričom čas učenia bol 75 minút. Najlepšie výsledky dosiahla sieť po 150 epochách pri použití učiaceho polomeru 0,0004 s postupným zmenšovaním, optimalizátora Adam. Konvolučnú sieť sa podarilo na trénovacích dátach naučiť na 86,93%. Na testovacích to bolo 69%. Pri klasifikácií pomocou k-means som na tvorbu vektora, ktorý by zachovával charekteristické črty použil môj model konvolučnej siete, ale bez poslednej vrstvy.
- Po zvýšení počtu epoch na 300 k-means označil všetko ako neplagiát, ale konvolučná sieť mala presnosť 67%.
- Po zmenení učiaceho pomeru na 0,04 už k-means označil všetko ako neplagiát, ale konvolučná sieť mala presnosť 69%.
- Po zmene optimalizátora na SGD konvolučná sieť označila všetko ako neplagiát, ale k-means mal presnosť 55%.
- V prípade ak sa použije cyklický učiaci pomer (viď. https://arxiv.org/abs/1506.01186) stačí použiť 5 epoch na to, aby konvolučná sieť označila všetko ako neplagiát, ale k-means mal presnosť 55%. Čo je rovnaký výsledok ako po 150 epochách s použitím SGD optimalizátora.

Pravdepodobne sieť stratí príliš veľa informácii o plagiáte už tým, že sa dĺžka textu oreže alebo predĺži na 40 000 slov. Taktiež má problémy nájsť charakteristické črty plagiátov. To môže byť spôsobené tým, že obrázky plagiátov navzájom medzi sebou nemusia mať súvislosť. Napr. Ak niekto opísal Bibliu a iný kuchársku knihu, tak obaja majú plagiát, ale medzi sebou nemajú zhodu.

Matus Revicky

Digitálne podpísal Matus Revicky
Dátum: 2020.12.05 15:32:39 +01'00'