NLP - Raport

Jakub Solecki

Trenowanie modeli klasyfikacyjnych dla zbioru filmów

Model neuronalny (biblioteka Simple Transformers)

Zbiór danych o filmach był już dostosowany do wymagań bibliotek. Zawiera on 1283 filmy z gatunku komedia (0) oraz 1273 z gatunku thriller (1):

```
[24] from sklearn.model_selection import train_test_split

     all_data = pd.read_csv("selected_films.csv")
     all_data = all_data.rename(columns={'label': 'labels', 'description': 'text'})
     all_data['text'] += ' ' + all_data['title']
     all_data['labels'] = all_data['labels'].map({'thriller': 1, 'komedia': 0})
     print(all_data.columns)
     print(all_data['labels'].value_counts())

     Index(['title', 'year', 'labels', 'text'], dtype='object')
     0    1283
     1    1273
     Name: labels, dtype: int64
```

Następnie zbiór wszystkich danych został podzielony na podzbiory: treningowy (`train_df`) oraz testowy (`test_df`)

```
[13] train_df, test_df = train_test_split(all_data, train_size=0.9)
     print(train_df.columns)
     print(train_df['labels'].value_counts())
     print(test_df['labels'].value_counts())

     Index(['title', 'year', 'labels', 'text'], dtype='object')
     0    1155
     1    1145
     Name: labels, dtype: int64
     1    128
     0    128
     Name: labels, dtype: int64
```

Różnica w liczebności gatunków filmów w podzbiorze treningowym jest marginalna a w testowym nie występuje. Model można ostatecznie poddać treningowi.

W wyniku treningu modelu otrzymałem następujące rezultaty:

```
{'mcc': 0.8517964850293742, 'tp': 120, 'tn': 117, 'fp': 11, 'fn': 8,
'acc': 0.92578125, 'eval_loss': 0.3432560822329833}
```

Należy je interpretować następująco:

- **mcc** - jest to współczynnik korelacji Matthewsa (Matthews correlation coefficient), pomiędzy obserwowanymi oraz przewidywanymi wynikami. Dla naszego modelu wynosi on w przybliżeniu 0.8518. Predykcja na tym poziomie, a więc znacznie zbliżona do 1 (doskonała predykcja), jest dobrym wynikiem.
- **Macierz błędów** prezentuje się następująco:

		Klasa rzeczywista	
		Pozytywna	**Negatywna**
Klasa przewidywana	**Pozytywna**	**120**	11
	Negatywna	8	**117**

  Poprawnie zostały zidentyfikowane 237 z 256 filmów ze zbioru testowego.

- **acc** - dokładność, wynosi w przybliżeniu 0.9258. Dzięki pomijalnej różnicy w liczebności klas (gatunków) filmów W obu podzbiorach, można uznać, że ta wartość jest miarodajna.

Na potrzeby porównania z metodą opartą o klasyfikator bayesowski musimy wyliczyć jeszcze następujące miary:

- **Czułość (recall)** - Rc = TP/(TP+FN) = 120/(120+8) = 0.9375
- **Swoistość (precision)** - Pr = TP/(TP+FP) = 120/(120+11) = 0.9160
- **Miara F1** - F1 = (2*Rc*Pr)/(Rc+Pr) = (2*0.9375*0.9160)/(0.9375+0.9160) = 0.9266

Poniżej zamieszczam filmy, dla których została przypisana niewłaściwa klasa (wraz z tą, którą powinny mieć)

```
Marge i Dick Nelson zostają uprowadzeni przez cesarza Toda na odległą
planetę Spengo, którą zamieszkują sami idioci. Okrutny władca
zamierza poślubić kobietę oraz zniszczyć Ziemię. Mama i tata ocalają
świat
komedia

Nałogowy hazardzista zdobywa informację o pewniaku na jedną z gonitw.
W nadchodzący weekend postanawia wykorzystać nadarzającą się szansę.
Niech się dzieje co chce
komedia

Topper wyrusza w niebezpieczną podróż, aby ocalić pułkownika Dentona.
Hot Shots 2
komedia

Kat po kolejnym upokorzeniu przez męża postanawia się na nim zemścić.
Kaliber 45
thriller

Trzech najlepszych przyjaciół dopada refleksja na temat spraw
sercowych. Ten niezręczny moment
komedia
```

Życie popularnego pisarza zamienia się w koszmar, gdy nawiązuje znajomość ze swoim trzynastoletnim fanem. Nocny słuchacz
thriller

Rok 1944. Rozwiedziona Nita Longley przeprowadza się z dwoma synami do małego miasta w Teksasie, by dzień i noc pracować jako telefonistka. Przybłęda
thriller

Podczas wyprawy do Los Angeles matka z synem doznają poważnego wypadku, przez który zmuszeni są walczyć o przetrwanie. Monolit
thriller

Do USA przylatuje Królowa Elżbieta II. Porucznik Frank Drebin ma za zadanie zapewnić jej bezpieczeństwo. Naga broń: Z akt Wydziału Specjalnego
komedia

Zach jest załamany po niespodziewanej śmierci swojej dziewczyny. Kiedy Beth w tajemniczy sposób powraca, chłopak dostaje drugą szansę na miłość. Life After Beth
komedia

Podczas wigilijnego wieczoru grupa nieznajomych zostaje uwięziona w windzie. Christmas Eve
komedia

Młody surfer wyjeżdża wraz z bratem do Kolumbii, gdzie zakochuje się w pięknej Marii. Wkrótce poznaje jej wuja Pablo Escobara, handlarza kokainy. Escobar: Historia nieznana
thriller

Dziennikarz poszukuje córki, którą przed laty oddał do adopcji. The Near Room
thriller

Gospodarz telewizyjnego show wraca w rodzinne strony, by zweryfikować swoje podej
ście do życia. Witaj w domu, panie Jenkins
komedia
Niechcący przegapić przyjęcia urodzinowego swojej córki mężczyzna tkwi w ulicznym korku. Kiedy puszczają mu nerwy, kontynuuje wędrówkę pieszo, wyładowując swoją złość na innych ludziach. Upadek
thriller

Upokorzona przez męża i jego kochankę Birdee wraca do rodzinnego miasta. Próbuje rozpocząć nowe życie. Ulotna nadzieja
komedia

Małżeństwu wiodącemu nudne życie wydaje się, że u sąsiadów doszło do zbrodni. Postanawiają rozwiązać zagadkę na własną rękę. Tajemnica morderstwa na Manhattanie
komedia

Rok 1997. Zamrożony w latach 60. superszpieg Austin Powers zostaje aktywowany przez rząd brytyjski, aby powstrzymać Doktora Zło i jego

```
organizację. Austin Powers: Agent specjalnej troski
komedia

Billy, Bones i Rat starają się przetrwać w Detroit, rządzonym przez
okrutnego Bully'ego.  Lost River
thriller
```

Model Bayesowski

Następnie użyłem klasyfikatora bayesowskiego z wagowaniem TD*IDF. Po jego wytrenowaniu otrzymałem następujące wyniki:

```
2300 256 2300 256
Fitting 2 folds for each of 18 candidates, totalling 36 fits
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent
workers.
[Parallel(n_jobs=3)]: Done   2 tasks      | elapsed:    2.1s
[Parallel(n_jobs=3)]: Done   7 tasks      | elapsed:    2.7s
[Parallel(n_jobs=3)]: Done  12 tasks      | elapsed:    3.2s
[Parallel(n_jobs=3)]: Done  19 tasks      | elapsed:    4.0s
[Parallel(n_jobs=3)]: Done  26 tasks      | elapsed:    4.6s
Best parameters set:
[('tfidf', TfidfVectorizer(analyzer='word', binary=False,
decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=0.5,
max_features=None,
                min_df=1, ngram_range=(1, 1), norm='l2',
preprocessor=None,
                smooth_idf=True, stop_words=[], strip_accents=None,
                sublinear_tf=False,
token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)),
('clf', OneVsRestClassifier(estimator=MultinomialNB(alpha=0.001,
class_prior=None,
                                                    fit_prior=True),
                    n_jobs=None))]

Applying best classifier on test data:
              precision    recall  f1-score   support

           0       0.88      0.95      0.91       128
           1       0.94      0.88      0.91       128

    accuracy                           0.91       256
   macro avg       0.91      0.91      0.91       256
weighted avg       0.91      0.91      0.91       256

[Parallel(n_jobs=3)]: Done  36 out of  36 | elapsed:    5.6s finished
```

Porównanie wyników działania obu metod dla zbioru filmów

Miara	NLP	Klasyfikator Bayesowski
acc	0.9258	0.9100
Rc	0.9375	Komedia: 0.8800
		Thriller: 0.9400
		Ogółem: 0.9100
Pr	0.9160	Komedia: 0.9500
		Thriller: 0.8800
		Ogółem: 0.9100
F1	0.9266	Komedia: 0.9100
		Thriller: 0.9100
		Ogółem: 0.9100

Jak wynika z powyższego zestawienia, nowoczesne metody NLP, oparte o bibliotekę Simple Transformers dają (nie)znacznie lepsze rezultaty. Z kolei klasyfikator bayesowski zapewnia równy rozkład we wszystkich miarach.

Trenowanie modeli klasyfikacyjnych dla zbioru danych PolEval

Dostosowanie formatu zbioru danych do wymagań biblioteki Simple Transformers

Rozpocząłem od wczytania danych z pliku. Następnie z każdego wiersza (komentarza, tweeta) usunąłem znak nowej linii ‘\n’. Kolejnym krokiem było mapowanie tagów tweetów zgodnie z konwencją (analogicznie jak w przypadku zbioru filmów) na 1 (cybernękanie) lub 0 (zwykły wpis). Tak przygotowane dane zostały umieszczone w słowniku gdzie każdy z 2 podzbiorów został odpowiednio oznaczy jako ‘text’ lub ‘labels’. Słownik ten został następnie ostatecznie dostosowany za pomocą biblioteki pandas. Widoczna jest silna dysproporcja w liczebnościach poszczególnych klas (wpisy noszące znamiona cybernękania oraz pozostałe). Może być to potencjalną przeszkodą w rzetelności przyszłych wyników.
Zbiór główny zawiera 8286 wpisów kategorii 0 oraz 750 kategorii 1, z kolei testowy zbiór zawiera 904 wpisy kategorii 0 oraz 10 kategorii 1.

```
[14] pol_eval = open("training_set_clean_only_text.txt", "r")
     tweets = pol_eval.readlines()
     tweets = [tweet.rstrip() for tweet in tweets]
     # print(tweets)
     pol_eval_tags = open("training_set_clean_only_tags.txt", "r")
     labels = pol_eval_tags.readlines();
     labels = [int(label) for label in labels]

     pol_eval_data = {}
     pol_eval_data['labels'] = labels
     pol_eval_data['text'] = tweets

     pol_eval_df = pd.DataFrame(data=pol_eval_data)
```

```
[15] train_pol_eval, test_pol_eval = train_test_split(pol_eval_df, train_size=0.9)
     print(train_pol_eval.columns)
     print(train_pol_eval['labels'].value_counts())
     print(test_pol_eval['labels'].value_counts())
```

```
Index(['labels', 'text'], dtype='object')
0    8286
1     750
Name: labels, dtype: int64
0    904
1    101
Name: labels, dtype: int64
```

Trening sieci neuronowej

Poniżej prezentuję wyniki treningu:

```
{'mcc': 0.0, 'tp': 0, 'tn': 904, 'fp': 0, 'fn': 101, 'acc':
0.8995024875621891, 'eval_loss': 0.3370856500807263}
```

Pierwsze co rzuca się w oczy to parametr mcc wynoszący 0. Jest to wynik zbyt dużej dysproporcji w liczebności poszczególnych klas. Na podstawie pozostałych możemy zbudować macierz błędów:

		Klasa rzeczywista	
		Pozytywna	**Negatywna**
Klasa przewidywana	**Pozytywna**	**0**	101
	Negatywna	0	**904**

Jak widać model praktycznie nie działa ponieważ wszystkie przykłady ze zbioru testowego zostały zaliczone jako wpisy bez znamion cybernękanie, pomimo, że ponad 11% z nich je posiada. Z tego powodu miara dokładności, choć wynosząca w przybliżeniu 0.8995 i potencjalnie obiecująca jest w rzeczywistości całkowicie niemiarodajna. W wyniku przepuszczenia przez sieć znacząco mniejszej liczby przykładów cybernękania (w stosunku do zwykłych) model nie nauczył się ich poprawnie rozpoznawać. Przykładów jest tak mało, że zaliczając wszystkie jako zwyczajne tweety osiągnięto potencjalnie wysoką dokładność. Są to jednak tylko pozory.

Trening modelu bayesowskiego

Dane poddane zostały analogicznym zabiegom jak te ze zbioru filmów;

```
train_pol_eval_x = [x.strip() for x in
train_pol_eval['text'].tolist()]
test_pol_eval_x = [x.strip() for x in test_pol_eval['text'].tolist()]
train_pol_eval_y = [str(x) for x in
train_pol_eval['labels'].tolist()]
test_pol_eval_y = [str(x) for x in test_pol_eval['labels'].tolist()]
print(len(train_pol_eval_x), len(test_pol_eval_x),
len(train_pol_eval_y), len(test_pol_eval_y))
grid_search(train_pol_eval_x, train_pol_eval_y, test_pol_eval_x,
test_pol_eval_y, ['0', '1'], parameters, pipeline)
```

Wyniki treningu prezentują się następująco:

```
9036 1005 9036 1005
Fitting 2 folds for each of 18 candidates, totalling 36 fits
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent
workers.
[Parallel(n_jobs=3)]: Done   2 tasks      | elapsed:    2.0s
[Parallel(n_jobs=3)]: Done   7 tasks      | elapsed:    3.4s
[Parallel(n_jobs=3)]: Done  12 tasks      | elapsed:    5.1s
[Parallel(n_jobs=3)]: Done  19 tasks      | elapsed:    7.1s
[Parallel(n_jobs=3)]: Done  26 tasks      | elapsed:    9.1s
Best parameters set:
[('tfidf', TfidfVectorizer(analyzer='word', binary=False,
decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=0.5,
max_features=None,
                min_df=1, ngram_range=(1, 1), norm='l2',
preprocessor=None,
                smooth_idf=True, stop_words=[], strip_accents=None,
                sublinear_tf=False,
token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)),
('clf', OneVsRestClassifier(estimator=MultinomialNB(alpha=0.01,
class_prior=None,
                                          fit_prior=True),
                    n_jobs=None))]

Applying best classifier on test data:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       904
           1       0.65      0.36      0.46       101

    accuracy                           0.92      1005
   macro avg       0.79      0.67      0.71      1005
weighted avg       0.90      0.92      0.91      1005

[Parallel(n_jobs=3)]: Done  36 out of  36 | elapsed:   12.2s finished
```

W przypadku klasyfikatora bayesowskiego wyniki są podobne do tych pozyskanych z sieci neuronowej. W rozpoznawaniu tweetów noszących znamiona cybernękania model nie radzi sobie zbyt dobrze. Powód jest oczywisty i wcześniej już wspomniany: zbyt znaczna dysproporcja w zbiorach przykładów treningowych. Sytuacja jest dokładnie taka sama jak poprzednio: wszystkie przykłady ze zbioru testowego zostały zakwalifikowane jako zwykłe tweety. Porównanie wyników miar dla poszczególnych klas pokazuje skalę błędu.

Modyfikacja hiperparametrów sieci neuronowej

- Batch size: 16 - brak poprawy

```
{'mcc': 0.0, 'tp': 0, 'tn': 927, 'fp': 0, 'fn': 78, 'acc':
0.9223880597014925, 'eval_loss': 0.26408732765250736}
```

- Batch size: 4 - brak poprawy

```
{'mcc': 0.0, 'tp': 0, 'tn': 927, 'fp': 0, 'fn': 78, 'acc':
0.9223880597014925, 'eval_loss': 0.26939207099614637}
```

- Number of epochs: 10 - brak poprawy

```
{'mcc': 0.0, 'tp': 0, 'tn': 927, 'fp': 0, 'fn': 78, 'acc':
0.9223880597014925, 'eval_loss': 0.2743388610108504}
```

- Number of epochs: 2 - brak poprawy

```
{'mcc': 0.0, 'tp': 0, 'tn': 927, 'fp': 0, 'fn': 78, 'acc':
0.9223880597014925, 'eval_loss': 0.27382335669937585}
```

- Zmiana wag klas: 0.1 i 0.9 - brak poprawy

```
{'mcc': 0.0, 'tp': 0, 'tn': 927, 'fp': 0, 'fn': 78, 'acc':
0.9223880597014925, 'eval_loss': 0.7421786063128993}
```

- Zmiana wag klas: 0.9 i 0.1 - brak poprawy

```
{'mcc': 0.0, 'tp': 0, 'tn': 927, 'fp': 0, 'fn': 78, 'acc':
0.9223880597014925, 'eval_loss': 0.058149016543572386}
```

- Modyfikacja wielu parametrów jednocześnie:
    - Number of epochs: 10
    - Wagi klas: 0.5 i 0.5
    - Batch size: 128

```
{'mcc': 0.49572522983446343, 'tp': 39, 'tn': 890, 'fp': 15,
'fn': 61, 'acc': 0.9243781094527364, 'eval_loss':
0.2523548046296965}
```

Widać znaczącą poprawę w wynikach. Sieci udało się poprawnie rozpoznać 39 przykładów cybernękania, jednak zaliiczono także omylnie 15 zwykłych wpisów.

- Modyfikacja wielu parametrów jednocześnie:
    - Number of epochs: 10
    - Wagi klas: 1 i 2
    - Batch size: 64

```
{'mcc': 0.54509398624971, 'tp': 46, 'tn': 891, 'fp': 23, 'fn':
45, 'acc': 0.9323383084577115, 'eval_loss': 0.47893669660247506}
```

Widoczna jest jeszcze większa poprawa wyników niż poprzednio.

Jak widać na powyższych przykładach modyfikowanie pojedynczych parametrów nie przynosi oczekiwanych rezultatów. Dopiero jednoczesne ich dostosowanie pozwala na stopniową niwelację dysproporcji w zbiorze danych. Przy odpowiednim dobraniu wartości parametrów prawdopodobnie będzie można osiągnąć bardzo zadowalające wyniki. Dzięki hiperparametrom możemy dostosowywać własności modelu do różnych zbiorów danych. Zyskujemy dzięki temu ogromną elastyczność w dostosowywaniu go do wymaganej sytuacji.