

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп'ютерних наук та кібернетики

Лабораторна робота

з дисципліни «Ком'ютерна лінгвістика»

на тему

«Класифікація фільмів

на основі алгоритмів машинного навчання та фільтрів»

Виконали
студенти групи МІ-4
Альошко Олексій, Богусевич Олексій, Гуртовий Леонід

Київ - 2020

Метою лабораторної роботи було розробити систему для користувача, за допомогою якої можна переглядати список найкращих фільмів за версією IMDb, передбачати жанр фільму по сюжету та робити огляд рецензій. Крім цього, за допомогою моделі система може встановлювати яка це рецензія: позитивна чи негативна. Також за кожним фільмом можна передивлятися теги (слова, що вживаються частіше за все в сюжеті)

Умовно роботу можна поділити на такі частини:

- 1. Розробка клієнтської частини проекту**
- 2. Розробка моделі для розпізнавання жанру фільму за сюжетом**
- 3. Розробка моделі для розпізнавання позитивної чи негативної рецензії**
- 4. Віділення тегів фільму**

Розробка клієнтської частини проекту

Виконав: Богусевич Олексій

Реалізовано веб-застосунок на основі технології ASP .NET Core за принципом MVC. Користувач заходить на головну сторінку та має можливість обрати один з трьох запитів: на отримання найпопулярніших фільмів, найбільшими касових фільмів та найкращих за версією користувачів порталу.

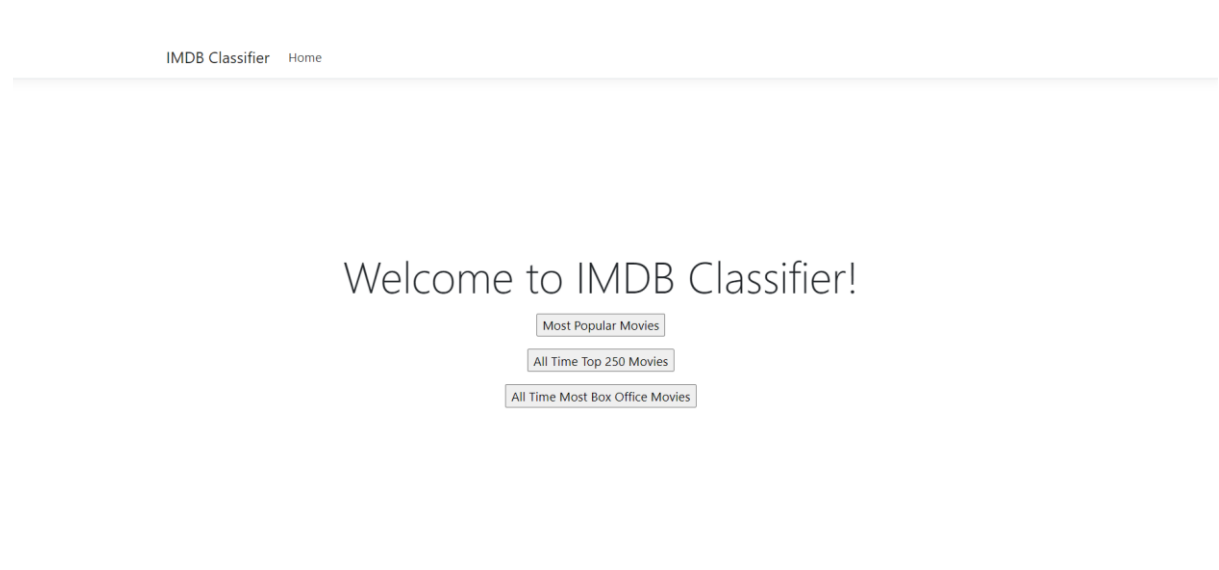


Рисунок 1. Стартова сторінка сайту

IMDB Classifier Home

IMDB All Time Top 250 Movies as on 12/3/2020 7:12:29 AM

Id	Title	Year	IMDbRating	Rank	Feature
tt0111161	The Shawshank Redemption	1994	9.2	1	Predict
tt0068646	The Godfather	1972	9.1	2	Predict
tt0071562	The Godfather: Part II	1974	9.0	3	Predict
tt0468569	The Dark Knight	2008	9.0	4	Predict
tt0050083	12 Angry Men	1957	8.9	5	Predict
tt0108052	Schindler's List	1993	8.9	6	Predict
tt0167260	The Lord of the Rings: The Return of the King	2003	8.9	7	Predict
tt0110912	Pulp Fiction	1994	8.8	8	Predict
tt0060196	The Good, the Bad and the Ugly	1966	8.8	9	Predict
tt0120737	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	10	Predict
tt0137523	Fight Club	1999	8.8	11	Predict
tt0108280	Forrest Gump	1994	8.8	12	Predict

Рисунок 2. Результат виконання запиту «Топ 250 фільмів»

The Godfather (1972)

- Year: 1972
- [Wikipedia](#)
- Predicted Genre: Documentary
- [Reviews \(predicted negative / positive\)](#)

The Godfather The Godfather is a 1972 American crime film directed by Francis Ford Coppola who co-wrote the screenplay with Mario Puzo, based on Puzo's best-selling 1969 novel of the same name. The film stars Marlon Brando, Al Pacino, James Caan, Richard Castellano, Robert Duvall, Sterling Hayden, John Marley, Richard Conte, and Diane Keaton. It is the first installment in The Godfather trilogy. The story, spanning from 1945 to 1955, chronicles the Corleone family under patriarch Vito Corleone (Brando), focusing on the transformation of one of his sons, Michael Corleone (Pacino), from reluctant family outsider to ruthless mafia boss. Paramount Pictures obtained the rights to the novel for the price of \$80,000, before it gained popularity. Studio executives had trouble finding a director; their first few candidates turned down the position before Coppola signed on to direct the film. They and Coppola disagreed over the casting for several characters, in particular, Vito and Michael. Filming took place primarily on location around New York City and in Sicily, and was completed ahead of schedule. The musical score was composed principally by Nino Rota, with additional pieces by Carmine Coppola. The Godfather premiered at the Loew's State Theatre on March 14, 1972, and was widely released in the United States on March 24, 1972. It was the highest-grossing film of 1972, and was for a time the highest-grossing film ever made, earning between \$246 and \$287 million at the box office. The film received universal acclaim from critics and audiences, with praise for the performances, particularly those of Brando and Pacino, the directing, screenplay, cinematography, editing, score, and portrayal of the mafia. The Godfather acted as a catalyst for the successful careers of Coppola, Pacino, and other relative newcomers in the cast and crew. Additionally the film revitalized Brando's career, which had declined in the 1960s, and he went on to star in films such as Last Tango in Paris, Superman, and Apocalypse Now. At the 45th Academy Awards, the film won the Oscars for Best Picture, Best Actor (Brando), and Best Adapted Screenplay (for Puzo and Coppola). In addition, the seven other Oscar nominations included Pacino, Caan, and Duvall for Best Supporting Actor, and Coppola for Best Director. Since its release, The Godfather has been widely regarded as one of the greatest and most influential films ever made, especially in the gangster genre. It was selected for preservation in the U.S. National Film Registry of the Library of Congress in 1990, being deemed "culturally, historically, or aesthetically significant" and is ranked the second-greatest film in American cinema (behind Citizen Kane) by the American Film Institute. It was followed by sequels The Godfather Part II (1974) and The Godfather Part III (1990). Plot In 1945 New York City, at his daughter Connie's wedding to Carlo, Vito Corleone, the don of the Corleone crime family listens to requests. His youngest son, Michael, who was a Marine during World War II, introduces his girlfriend, Kay Adams, to his family at the reception. Johnny Fontane, a popular singer and Vito's godson, seeks Vito's help in securing a movie role; Vito dispatches his consigliere, Tom Hagen, to Los

episode "Lisa's Pony", Lisa wakes up to find a horse in her bed and starts screaming, a reference to a scene in The Godfather. In the season 4 episode "Mr. Plow", Bart Simpson is pelted with snowballs in mimicry of Sonny Corleone's killing. The film's baptism sequence was parodied in the season 4 episode "Fulgencio", of the comedy series Modern Family. The Godfather (2006) is based upon this film and tells the story of an original character, Aldo Trapani, whose rise through the ranks of the Corleone family intersects with the plot of the film on numerous occasions. Duvall, Caan, and Brando supplied voiceovers and their likenesses, but Pacino did not. Francis Ford Coppola openly voiced his disapproval of the game.

Tags:

1. coppola

Appeared in text (times): 77

Normalized: 0.5699483426615536

2. godfath

Appeared in text (times): 58

Normalized: 0.4293117386281832

3. best

Appeared in text (times): 48

Normalized: 0.355292473347462

4. corleon

Appeared in text (times): 39

Normalized: 0.28867513459481287

5. film

Appeared in text (times): 36

Normalized: 0.2664693550105965

Рисунок 3-4. Результат роботи алгоритмів визначення тегів та прогнозу жанру

Розробка моделі для розпізнавання жанру фільму за сюжетом

Виконали: Богусевич Олексій, Альошко Олексій, Гуртовий Леонід

Розпізнавання жанру виконується за алгоритмом SVM

Принцип роботи:

Основним завданням алгоритму є пошук оптимальної лінії, або гіперплощини, що розділяє дані на два класи. SVM алгоритм отримує на вході дані і повертає таку розділяючу лінію.

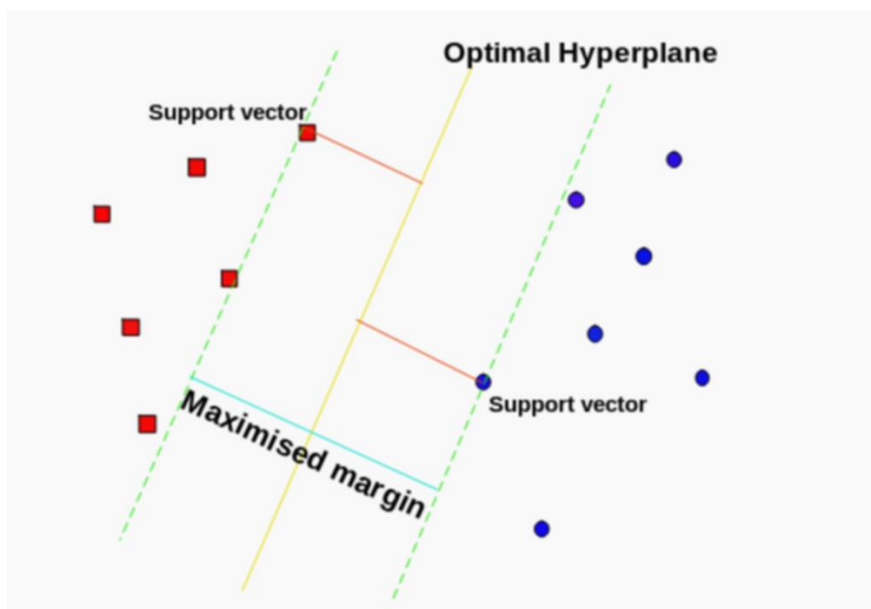


Рисунок 5. Принцип роботи SVM

- Нелінійний випадок

В нелінійному випадку виконуємо перетворення в $N+1$ простір доти, доки не можемо поділити точки гіперплощиною. У випадку наших фільмів точки на графіку це сюжети фільмів по декільком критеріям знаходимо до них опорні вектори за описаним алгоритмом, виділяємо кластери, що відносяться до одного жанру чи іншого.

```

from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from utils import predict
from preprocess import preprocess
import pandas as pd
import pickle

pickle_model_file = 'finalized_model.sav'
pickle_vectorizer_file = 'vectorizer.sav'
pred_file = r'D:\Programming\Python\imdbGenreClassification-master\data\pred.csv'

data_features = preprocess(r'D:\Programming\Python\imdbGenreClassification-master\data\trainingSet.csv')
train_data, test_data = train_test_split(data_features, test_size=0.1, random_state=42)

vectorizer = TfidfVectorizer(min_df=2, tokenizer=None, preprocessor=None, stop_words=None)

train_data_features = vectorizer.fit_transform(train_data['plot'])
train_data_features = train_data_features.toarray()

pickle.dump(vectorizer, open(pickle_vectorizer_file, 'wb'))

lin_clf = svm.LinearSVC()
lin_clf.fit(train_data_features, train_data['tags'])

predict(vectorizer, lin_clf, test_data)

pickle.dump(lin_clf, open(pickle_model_file, 'wb'))

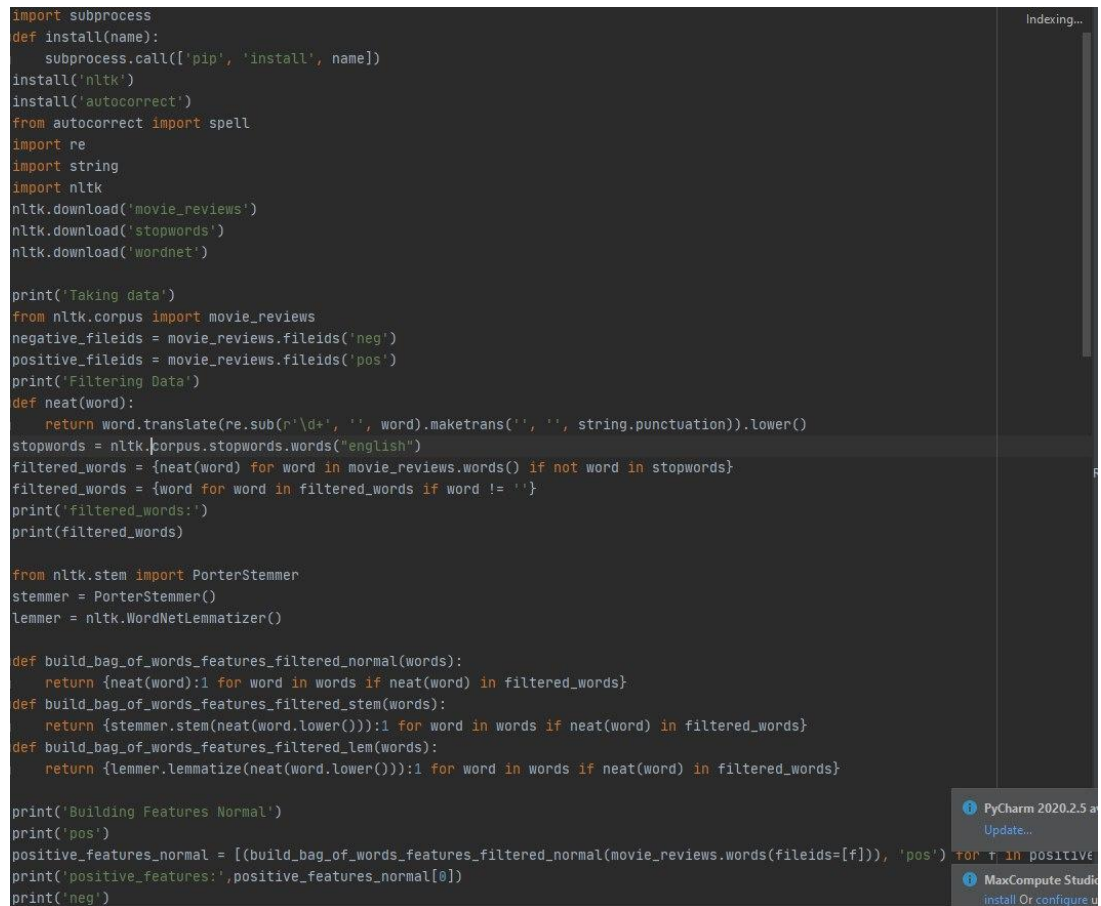
pred = pd.read_csv(pred_file)
vect = vectorizer.transform(pred['plot'])
res = lin_clf.predict(vect)
print(res[0])

```

Рисунок 6. Реалізація прогнозу жанру

Розробка моделі для розпізнавання позитивної чи негативної рецензії

Виконав: Альошко Олексій



```
import subprocess
def install(name):
    subprocess.call(['pip', 'install', name])
install('nltk')
install('autocorrect')
from autocorrect import spell
import re
import string
import nltk
nltk.download('movie_reviews')
nltk.download('stopwords')
nltk.download('wordnet')

print('Taking data')
from nltk.corpus import movie_reviews
negative_fileids = movie_reviews.fileids('neg')
positive_fileids = movie_reviews.fileids('pos')
print('Filtering Data')
def neat(word):
    return word.translate(re.sub(r'\d+', '', word).maketrans('', '', string.punctuation)).lower()
stopwords = nltk.corpus.stopwords.words("english")
filtered_words = {neat(word) for word in movie_reviews.words() if not word in stopwords}
filtered_words = {word for word in filtered_words if word != ''}
print('filtered_words:')
print(filtered_words)

from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
lemmer = nltk.WordNetLemmatizer()

def build_bag_of_words_features_filtered_normal(words):
    return {neat(word):1 for word in words if neat(word) in filtered_words}
def build_bag_of_words_features_filtered_stem(words):
    return {stemmer.stem(neat(word.lower())):1 for word in words if neat(word) in filtered_words}
def build_bag_of_words_features_filtered_lem(words):
    return {lemmer.lemmatize(neat(word.lower())):1 for word in words if neat(word) in filtered_words}

print('Building Features Normal')
print('pos')
positive_features_normal = [(build_bag_of_words_features_filtered_normal(movie_reviews.words(fileids=[f])), 'pos') for f in positive_fileids]
print('positive_features:', positive_features_normal[0])
print('neg')
```

Рисунок 7. Реалізація класифікації відгуку

```

print('neg')
negative_features_stem = [(build_bag_of_words_features_filtered_stem(movie_reviews.words(fileids=[f])), 'neg') for f in negative_fileids]
print('negative_features:', negative_features_stem[0])

print('Building Features Lem')
print('pos')
positive_features_lem = [(build_bag_of_words_features_filtered_lem(movie_reviews.words(fileids=[f])), 'pos') for f in positive_fileids]
print('positive_features:', positive_features_lem[0])
print('neg')
negative_features_lem = [(build_bag_of_words_features_filtered_lem(movie_reviews.words(fileids=[f])), 'neg') for f in negative_fileids]
print('negative_features:', negative_features_lem[0])

split=500
from sklearn.svm import SVC
from nltk.classify import SklearnClassifier

print('SVC N')
print('Training')
positive_features=positive_features_normal
negative_features=negative_features_normal
train_data = positive_features[:split]+negative_features[:split]
svcc = SklearnClassifier(SVC(), sparse=False).train(train_data)
print('Testing')
print(nltk.classify.util.accuracy(svcc, positive_features[:split]+negative_features[:split])*100)
print(nltk.classify.util.accuracy(svcc, positive_features[split:]+negative_features[split:])*100)

print('SVC S')
print('Training')
positive_features=positive_features_stem
negative_features=negative_features_stem
train_data = positive_features[:split]+negative_features[:split]
svcc = SklearnClassifier(SVC(), sparse=False).train(train_data)
print('Testing')
print(nltk.classify.util.accuracy(svcc, positive_features[:split]+negative_features[:split])*100)
print(nltk.classify.util.accuracy(svcc, positive_features[split:]+negative_features[split:])*100)

print('SVC L')
print('Training')
positive_features=positive_features_lem
negative_features=negative_features_lem
train_data = positive_features[:split]+negative_features[:split]

```

Рисунок 8. Реалізація класифікації відгуку


```
reviews_model.py x svm_train.py x remake.py x utils.py x pred.csv x preprocess.py x
print('Testing')
print(nltk.classify.util.accuracy(svcc, positive_features[:split]+negative_features[:split])*100)
print(nltk.classify.util.accuracy(svcc, positive_features[split:]+negative_features[split:])*100)

print('SVC S')
print('Training')
positive_features=positive_features_stem
negative_features=negative_features_stem
train_data = positive_features[:split]+negative_features[:split]
svcc = SklearnClassifier(SVC(), sparse=False).train(train_data)
print('Testing')
print(nltk.classify.util.accuracy(svcc, positive_features[:split]+negative_features[:split])*100)
print(nltk.classify.util.accuracy(svcc, positive_features[split:]+negative_features[split:])*100)

print('SVC L')
print('Training')
positive_features=positive_features_lem
negative_features=negative_features_lem
train_data = positive_features[:split]+negative_features[:split]
svcc = SklearnClassifier(SVC(), sparse=False).train(train_data)
print('Testing')
print(nltk.classify.util.accuracy(svcc, positive_features[:split]+negative_features[:split])*100)
print(nltk.classify.util.accuracy(svcc, positive_features[split:]+negative_features[split:])*100)

print('Testing for manual input')
tests={'its a good movie storyline best compared to story':'pos'}
test_features=[(build_bag_of_words_features_filtered_lem(each.split()),tests[each]) for each in tests.keys()]
for each in tests.keys():
    print(tests[each],':',each)
print('Accuracy:',nltk.classify.util.accuracy(svcc, test_features)*100)

import pickle

svcc_pickle = 'svcc_model.sav'
pickle.dump(svcc, open(svcc_pickle, 'wb'))
```

Рисунок 9. Реалізація класифікації відгуку

Виділення тегів фільму

Виконав: Гуртовий Леонід

Спочатку сюжет фільму звільняється від стоп-слів, після чого обрізається до кореня за алгоритмом Портера.

<https://github.com/alexeybogusevich/IMDBClassifier/blob/master/KNU.Lingua.Movies/Services/PorterStemmerFilter/PorterStemmerFilter.cs>

Реалізацію наведено за посиланням вище.

Далі беруться найбільш часто вживані слова.

```
public List<Tag> GetTopTagsForNewsItem(string text)
{
    var words = text.Split(' ', StringSplitOptions.RemoveEmptyEntries);
    var result = new List<Tag>();

    foreach (var word in words)
    {
        if (!result.Exists(delegate (Tag x)
        {
            return string.Equals(x.Name, word) ? true : false;
        }))
        {
            var newTag = new Tag(word, 1, 0);
            result.Add(newTag);
        }
        else
        {
            result.Find(x => x.Name == word).OccurrencesCount++;
        }
    }

    var sortedTags = result.OrderByDescending(x => x.OccurencesCount).Take(topTagsCount).ToList();
    double tagCountSquareSum = Math.Sqrt(sortedTags.Sum(t => t.OccurencesCount * t.OccurencesCount));

    foreach (var tag in sortedTags)
    {
        tag.NormCount = (double)tag.OccurencesCount / tagCountSquareSum;
    }

    return sortedTags;
}
```

Рисунок 10. Реалізація пошуку тегів

Посилання

- <https://tartarus.org/martin/PorterStemmer/>
- <https://www.kaggle.com/c/movie-genre-classification>
- <https://scikit-learn.org/stable/modules/svm.html>
- <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
- <https://imdb-api.com/api#Title-header>
- <https://github.com/alexeybogusevich/IMDBClassifier>
- <https://lingua-imdb.azurewebsites.net/>