УДК 004

СИСТЕМА АВТОМАТИЧНОЇ КАТЕГОРИЗАЦІЇ ТЕКСТОВИХ ДОКУМЕНТІВ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Сацько В.М., студентка, ЛНТУ vikasatcko@gmail.com
Мельник К.В., к.т.н., доцент, ЛНТУ ekaterinamelnik@gmail.com

В сучасному світі, коли обсяги інформації стрімко зростають, виникає необхідність в ефективних інструментах для зручної роботи з об’ємними текстовими даними. Таким чином, система для автоматичної категоризації текстових документів може значно допомогти організувати, узагальнити та здійснити пошук необхідної інформації.

Машинне навчання та штучний інтелект за останні роки стали актуальними та важливими технологіями у багатьох сферах життя. Штучний інтелект тісно пов'язаний з алгоритмами кластеризації та класифікації, оскільки ці алгоритми є ключовими для розв'язання багатьох задач у машинному навчанні [1].

Системи машинного навчання, які використовуються для категоризації текстових документів, базуються на аналізі попередніх даних. За допомогою навчальних даних, які включають попередньо розмічені приклади тексту, алгоритми машинного навчання вивчають взаємозв'язки між фрагментами тексту та попередньо заданими їм тегами.

Під час розробки системи використовувалися такі Python бібліотеки як matplotlib , sklrean, pandas, seaborn, nltk, pickle, textract, translators.

Основний принцип роботи програми полягає у класифікації тексту за допомогою попередньо навченої моделі наївного Баєсового класифікатора (рис.1) [3].

```
def classify(text, return_type="value"):
    try:
        # Завантаження моделі-класифікатора
        classifier_file = './model/naive_bayes_classifier.pkl'
        classifier = pickle.load(open(classifier_file, 'rb'))

        # Векторизація тексту
        vectorizer_file = './model/count_vectorizer.pkl'
        vectorizer = pickle.load(open(vectorizer_file, 'rb'))

        prediction = classifier.predict(vectorizer.transform([text]))
```

Рисунок 1 – Лістинг коду функції для визначення категорії тексту

Першим етапом у тренуванні класифікатора є виділення особливостей, які існують у текстових документах. Це зазвичай виконується за допомогою методів, що перетворюють текст у числове представлення у вигляді вектора. Один із популярних підходів - це "мішок слів", де вектор відображає частоту кожного слова у попередньо визначеному словнику (рис.2).

```
Бізнес: [('сша', 500), ('зростання', 481), ('млрд', 417), ('фунтів', 353), ('2004', 303), ('
Розваги: [('фільм', 410), ('the', 276), ('шоу', 246), ('роль', 233), ('фунтів', 228), ('філь
Політика: [('пан', 569), ('партії', 470), ('заявив', 396), ('торі', 388), ('фунтів', 349), (
Спорт: [('рахунком', 329), ('збірної', 293), ('англії', 266), ('метрів', 229), ('світу', 204
Технології: [('людей', 398), ('забезпечення', 304), ('пан', 278), ('мережі', 217), ('сша', 2
```

Рисунок 2 – Вигляд "мішку слів" який модель використовує під час навчання

Процес тренування класифікатора проводиться за допомогою методів бібліотеки sklearn (рис. 3).

```
def train_classifier(documents):
    X_train, X_test, y_train, y_test = get_splits(documents)

    # Векторизація тексту
    vectorizer = CountVectorizer(stop_words='none', ngram_range=(1, 3), min_df=3, analyzer='word')

    # Створення матриці термінів
    dtm = vectorizer.fit_transform(X_train)

    # Тренування Наївного баєсівого класифікатора
    naive_bayes_classifier = MultinomialNB()
    naive_bayes_classifier.fit(dtm, y_train)

    evaluate_classifier( title: "Naive Bayes\tTRAIN\t", naive_bayes_classifier, vectorizer, X_train, y_train)
    evaluate_classifier( title: "Naive Bayes\tTEST\t", naive_bayes_classifier, vectorizer, X_test, y_test)
```

Рисунок 3 – Лістинг коду, що забезпечує тренування моделі наївного Баєсового класифікатора

Після закінчення тренування, виводиться оцінка результатів (рис. 4), де найкращий результат – 1, найгірший – 0.

```
▰▰▰▰▱ Навчання моделі
Naive Bayes TRAIN      0.996629    0.996629    0.996629
Naive Bayes TEST       0.961798    0.961798    0.961798
```

Рисунок 4 – Оцінка результатів тренування моделі

Після того як модель було натреновано, можна користуватися класифікатором для визначення категорії текстів. Система в автоматизованому режимі читає та аналізує вміст усіх файлів в папці unsorted, визначає їх категорію, та виконує системні команди з переміщення файлів у відповідні папки, в залежності від визначеної категорії тексту (рис. 5)

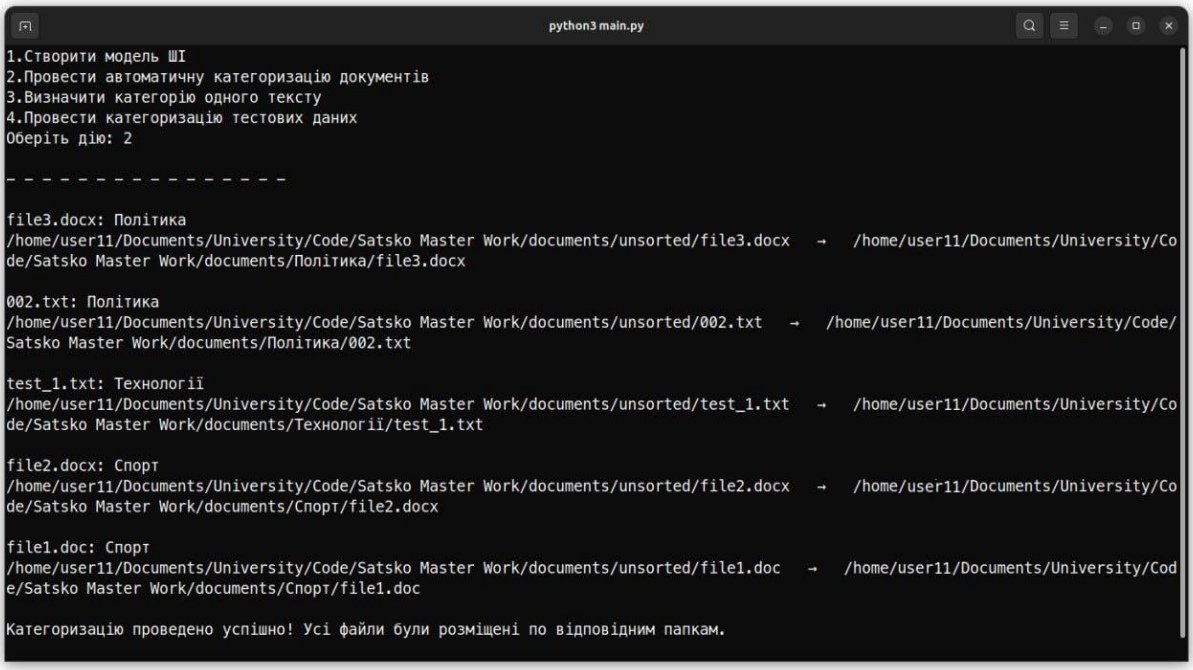

Рисунок 5 – Процес категоризації текстових файлів

Для навчання моделі було використано набір даних з новинами BBC, які були попередньо розбиті на 5 категорій: бізнес, розваги, політика, спорт, технології. Оригінальний набір даних був написаний англійською мовою, але під час розробки системи був створений перекладач, що в автоматизованому режимі переклав українською мовою більше ніж 2000 файлів, необхідних для навчання моделі. Після навчання на цих даних, модель навчилась розпізнавати тексти українською (рис. 6). Таким чином, при необхідності, розроблена система може працювати з будь якою мовою.

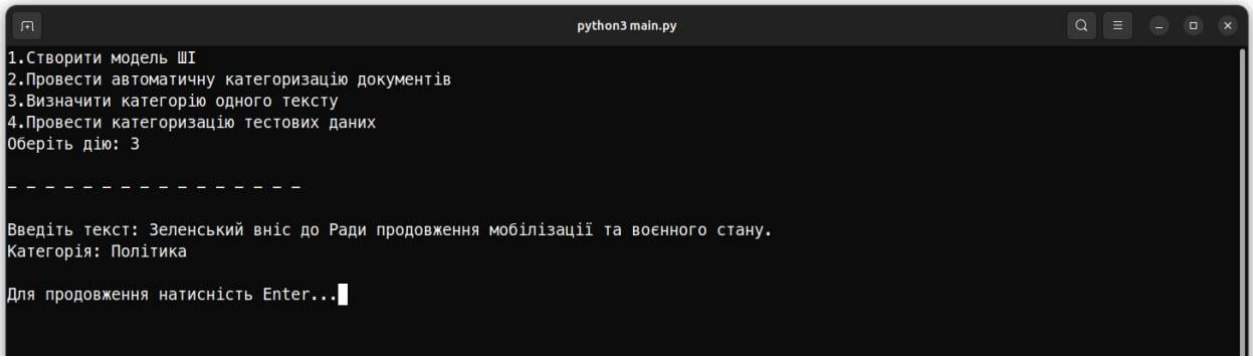

Рисунок 6 – Визначення категорії для довільного тексту

Висновок. У результаті проведеного дослідження були описані засоби для розробки системи для автоматичної категоризації текстових документів на основі алгоритмів кластеризації та класифікації. Важливим аспектом роботи є практична спрямованість системи, що дозволяє застосування її в різних галузях. А наявність вбудованого перекладача підкреслює можливість роботи з різними мовами, що значно підвищує універсальність системи.

Список використаних джерел

1. Pedamkar P. Machine Learning Methods | Types of Classification in Machine Learning. EDUCBA. URL: https://www.educba.com/machine-learning-methods/ (дата звернення: 21.02.2024).
2. Поліщук М.М., Цибень Д.В., Карплюк Ю.І. Обробка інформації за допомогою машинного навчання засобами Python. Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво", № 53: 2023. 203.
3. Machine learning: the problem setting. scikit-learn. URL: https://scikit-learn.org/0.21/tutorial/basic/tutorial.html (дата звернення: 22.02.2024).

УДК 004.42

ЗАСТОСУВАННЯ АЛГОРИТМУ ДЕЙКСТРИ В ЕЛЕКТРОННІЙ ТОРГІВЛІ

Селіванова А.В., старший викладач, ann.selivanova1@gmail.com, ДТЕУ
Самойленко Г.Т., канд.фіз.-матем. наук, доцент, anna_zak@ukr.net, ДТЕУ

Збільшення обсягів інтернет-продажів ставить нові виклики для підприємств, зокрема необхідність забезпечення вчасної доставки та одночасне скорочення логістичних витрат. Оптимізація маршрутів є ключовим фактором для підвищення ефективності логістичних процесів, особливо у сфері електронної торгівлі та доставки товарів. Застосування алгоритмів, що базуються на теорії графів, наприклад, алгоритми пошуку найкоротшого шляху, надають можливість урахування різноманітних обмежень та специфіки доставки товарів. Графи дозволяють моделювати мережу доставки, де вузли відповідають точкам видачі, складам або адресам доставки, а ребра відображають шляхи між цими точками [1].

Використання різних алгоритмів пошуку шляху, таких як DFS, BFS та алгоритм Дейкстри, дозволяє знаходити оптимальні маршрути в залежності від особливостей задачі та структури графа. Кожен з цих алгоритмів має свої