Министерство науки и высшего образования Российской Федерации ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Национальный исследовательский университет ИТМО» (Университет ИТМО)

Факультет Инфокоммуникационных технологий (ИКТ)

Образовательная программа Мобильные и сетевые технологии

ОТЧЕТ

по Лабораторной работе

Дисциплина: Методы сетевого анализа.

Специальность: 09.03.03 Прикладная информатика.

Выполнил:

Балцат К. И.,

студент группы К33401

ЗАДАНИЕ

На практике анализ текстовых данных в социальных сетях может быть полезен для целого ряда задач. К примеру, можно распознавать токсичные посты для их модерации или анализировать тональность текста, чтобы оценить отношение группы людей к тому или иному явлению. В этой лабораторной работе мы будем исследовать пользовательские посты, посвящённые COVID-19

Ход работы

1. Скачайте набор данных Covid 19 Indian Sentiments on covid19 and lockdown по ссылке

Covid 19 Indian Sentiments on covid19 and lockdown

С его помощью мы будем обучать модель распознавать настроение автора поста.

В датафрейме оставьте только те строки, у которых в столбце sentiment написано sad и joy

Для простоты выполнения оставляем только 2 возможных настроения, строки с постами, не соответствующими им, можно удалить.

Примените на этом наборе данных шаги предварительной обработки

К шагам предварительной обработки относятся:

приведение к нижнему регистру,

токенизация,

удаление стоп-слов (обратите внимание на то, что стоп-слова должны соответствовать языку, с которым ведётся работа, в данном случае это английский),

уберите заведомо неинформативные данные (хэштеги и отметки пользователей).

Также необходимо произвести векторизацию.

Пример выполнения этих шагов на другом наборе можно подсмотреть здесь:

Чтобы улучшить качество обучения можно попробовать перед векторизацией также применить лемматизацию (приведение слов к исходной форме). Для этого подойдёт, к примеру, nltk. WordNetLemmatizer (pymorphy2).

Плавное введение в Natural Language Processing (NLP)

Обучите классификатор

Можно использовать реализацию из библиотеки Scikit-learn

Оцените качество классификации. Обучите еще одну или несколько классический моделей классификаторов и сравните качество. Подберите наиболее оптимальные гиперпараметры векторизатора и классификатора, при которых метрики будут выше.

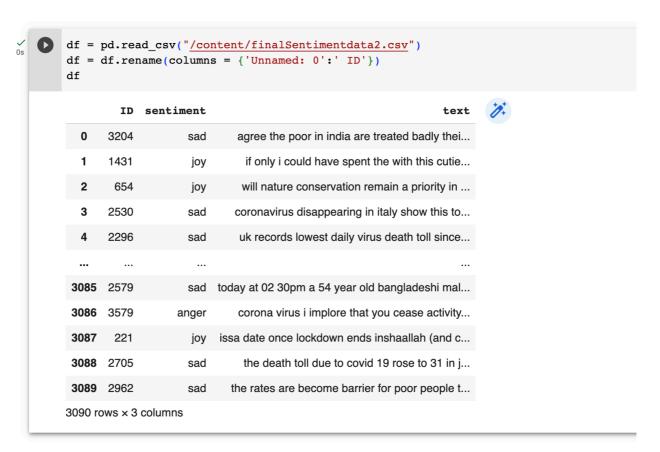
Примените шаги предобработки на наборе данных COVID19 Tweets

С помощью обученного классификатора найдите посты из набора данных COVID19 Tweets, которые можно отнести к классу sad. Какой процент от общего количества постов они составляют?

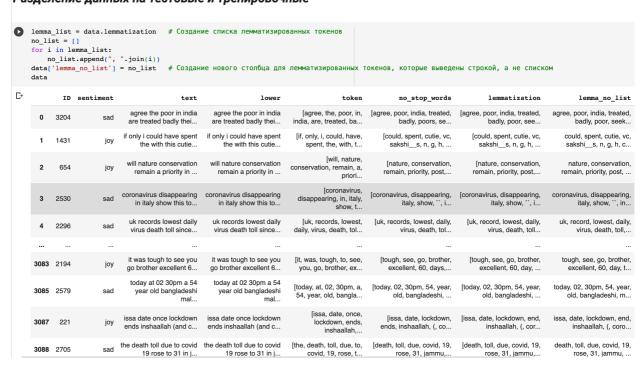
Проанализируйте тональность постов из набора данных COVID19 Tweets с помощью библиотеки TextBlob (Dostoevsky). Какое процентное соотношение постов разных настроений наблюдается?*

ВЫПОЛНЕНИЕ

- Выгрузка датасета



Разделение данных на тестовые и тренировочные



Обучение многоклассового Баейсовского наивного классификатора

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

```
[17] clf = MultinomialNB(force_alpha=True, alpha=0)
                                                                                             # Импорт классификатора
     cti - MutthomianNo(totec_atpha=ride, alpha=v)
clf.fit(vectorized_x_train, y_train)
pred = clf.predict(vectorized_x_test)
print("Точность предсказаний равна", accuracy_score(y_test, pred))
                                                                                            # Обучение классификатора на векторизованной тренировочной выборке
                                                                                             # Получение предсказаний для векторизованной тестовый выборки
     print(classification_report(y_test, pred))
     Точность предсказаний равна 0.6819672131147541
                      precision
                                        recall f1-score
                joy
sad
                             0.83
                                        0.47
                                                      0.60
                                                                     155
     macro avg
weighted avg
                             0.73
                                                       0.67
                                                                     305
                                         0.68
```

Попытка улучшения результатов

0.79

0.85 0.85

accuracy

macro avg weighted avg

0.92

0.83 0.84

0.85

0.84

0.83

155

305

```
+ Code - + Text -
                                                                                  # Импорт нового классификатора с включекным сглаживанием Лапласса
[18] clf = MultinomialNB(force_alpha=False, alpha=1)
                                                                                 # Обучение классификатора на векторизованной тренировочной выборке
# Получение предсказаний для векторизованной тестовый выборки
     clf.fit(vectorized_x_train, y_train)
    pred = clf.predict(vectorized_x_test)
     print("Точность предсказаний равна", accuracy_score(y_test, pred))
    print(classification_report(y_test, pred))
    Точность предсказаний равна 0.8360655737704918
                    precision
                                   recall f1-score support
              joy
sad
```

Обучение классификатора алгоритма случайного леса

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

0.82

```
olf1 = ensemble.RandomForestClassifier(n_estimators=1522, random_state=0, criterion="gini") # Импорт классификатора с количеством деревьев, равным количеству строк в дата clf1.fit(vectorized x_train, y_train) # 05/чение классификатора на векторизованной тренировочной выборке print(Townorts предсказаний равна", ассигасу_score(y_test, predl)) print(classification_report(y_test, predl))

[-] Точность предсказаний равна 0.819672131147541 precision recall fl-score support
```

Попытка улучшения результатов

0.84

accuracy macro avg weighted avg

accuracy macro avg weighted avg

```
[20] clfl = ensemble.RandomForestClassifier(n_estimators=100, random_state=0, criterion="gini")

clfl.fit(vectorized_x_train, y_train)
predl = clfl.predict(vectorized_x_test)
print("TowHorth ppepckxasamum pasma 0.8065573770491803
print(classification_report(y_test, predl))

TowHooth предсказаний разма 0.8065573770491803
precision recall fl-score support

joy 0.74 0.94 0.83 150
sad 0.92 0.68 0.78 155
```

Обучение классификатора метода опорных векторов

0.80

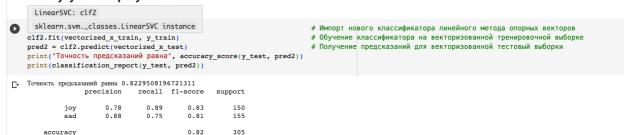
0.81

305 305 305

https://scikit-learn.org/stable/modules/svm.html

```
[23] clf2 = svm.SVC()
                                                                                                            # Импорт классификатора метода опорных векторов
                                                                                                            # Обучение классификатора на векторизованной тренировочной выборке # Получение предсказаний для векторизованной тестовый выборки
       clf2.fit(vectorized_x_train, y_train)
      print("Toчность предсказаний равна", accuracy_score(y_test, pred2))
print(classification_report(y_test, pred2))
       Точность предсказаний равна 0.7540983606557377 precision recall f1-sco
                                              recall f1-score
                    joy
sad
                                  0.98
                                                 0.53
                                                                0.69
                                                                                155
                                                               0.75
0.74
0.74
                                                                                305
305
       macro avg
weighted avg
                                                                                305
                                  0.83
```

Попытка улучшения результатов



Обучение классификатора метода ближайших соседей

 $\underline{https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html}$

```
[25] clf3 = KNeighborsClassifier(n neighbors=5)
                                                                               # Импорт классификатора с количеством соседей, равным 5
     clf3.fit(vectorized_x_train, y_train)
                                                                               # Обучение классификатора на векторизованной тренировочной выборке
    pred3 = clf3.predict(vectorized_x_test)
print("Точность предсказаний равна", accuracy_score(y_test, pred3))
                                                                               # Получение предсказаний для векторизованной тестовый выборки
    print(classification_report(y_test, pred3))
    Точность предсказаний равна 0.49836065573770494
                 precision
                                 recall f1-score support
                                              0.50
         accuracy
                       0.50
0.56 0.51 0.36
0.56 0.50 0.35
        macro avg
                                                           305
    weighted avg
                                                          305
```

Попытка улучшения результатов

```
[26] clf3 = KNeighborsclassifier(n_neighbors=3) # Импорт классификатора с количеством соседей, равным 3 clf3.fit(vectorized_x_train, y_train) # 0бучение классификатора на векторизованной тренировочной выборке print("Tочность предсказаний равна", асситасу_score(y_test, pred3))

Точность предсказаний равна 0.5245901639344263 precision recall fl-score support

joy 0.51 0.89 0.65 150 sad 0.61 0.17 0.27 155

асситасу 0.52 305 macro avg 0.56 0.53 0.46 305 weighted avg 0.56 0.52 0.46 305
```

Обучение классификатора метода градиентного бустинга

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

```
[29] clf4 = ensemble.GradientBoostingClassifier() # Импорт классификатора
clf4.fit(vectorized_x_train, y_train) # Обучение
pred4 = clf4.predict(vectorized_x_test) # Получение
print("TOHHOCTS предсказаний равна 0.819672131147541
precision recall f1-score support

joy 0.77 0.91 0.83 150
sad 0.90 0.73 0.80 155

accuracy 0.83 0.82 0.82 305
macro avg 0.83 0.82 0.82 305
weighted avg 0.83 0.82 0.82 305
```

Попытка улучшения результатов

```
[30] accuracy = 0
    for i in np.arange(0.0, 1.05, 0.05):
        clf4 = ensemble.GradientBoostingClassifier(learning_rate=i, random_state=0)
        clf4.fit(vectorized_x_train, y_train)
        pred4 = clf4.predict(vectorized_x_test)
        new_accuracy = accuracy_score(y_test, pred4)
        if new_accuracy > accuracy:
            accuracy = new_accuracy
        state = i

print(accuracy)
print(state)

0.8459016393442623
0.300000000000000004
```



```
| System to the continuence of t
```

вывод

Я выполнил лабораторную работу и на практике освоил методы анализа сетей.