**Методические указания к лабораторной работе № 2**

**Автор: Якунин Е.М.**

**Классификация**

Данная лабораторная работа посвящена обучению классификации текста по категориям.

**Цель работы:** классифицировать спарсенные данные.

**Задачи:**

1. Получить теоретические знания по теме.
2. Загрузить данные из csv-файлов.
3. Обработать данные (очистить).
4. Классифицировать данные.

**Классификация**

**Классификация** — задача компьютерной лингвистики, заключающаяся в отнесении документа к одной из нескольких категорий на основании содержания документа.

Классификация текстов одна из важнейших задач обработки естественного языка. Это процесс классификации текстовых строк или документов по различным категориям в зависимости от содержания строк. Классификация текста имеет множество приложений, таких как определение настроений пользователей по твиту, классификация электронной почтв как спама, классификация сообщений в блогах по различным категориям и т.д.

Использование алгоритмов машинного обучения для решения данных задач достаточно распространенное явление на сегодняшний день, поскольку программы, основанные на данных алгоритмах, имеют достаточно высокий показатель эффективности в сравнении с другими подходами классификации.

Задача классификация текстов содержит в себе несколько шагов:

1. Импорт набора данных
2. Предварительная обработка текста
3. Преобразование текста в числа
4. Учебные и тестовые наборы
5. Обучающая модель классификации текста и прогнозирование
6. Оценка модели.

На прошлом занятии мы уже спарсили новости спорта. Чтобы нам было, что классифицировать, получим еще новости политики, бизнеса и культуры. Для этого достаточно поменять переменную «URL» и название файла сохранения данных. Для удобства сразу создадим дополнительные столбец в csv-файле «Topic».

Данная работа будет выполнять в google.colab, т.к. там можно запускать код отдельно по блокам, что делает работу нагляднее.

Для начала импортируем необходимые библиотеки:

import pandas as pd

import numpy as np

from tqdm.auto import tqdm, trange

from google.colab import files

from sklearn.utils import shuffle

Загрузим csv-файлы:

# загрузим спарсенные данные

uploaded = files.upload()

 **News.csv**(text/csv) - 596462 bytes, last modified: 03.05.2022 - 100% done

 **NewsBisnes.csv**(text/csv) - 169830 bytes, last modified: 03.05.2022 - 100% done

 **NewsCulture.csv**(text/csv) - 1245419 bytes, last modified: 03.05.2022 - 100% done

 **NewsPolitics.csv**(text/csv) - 2207207 bytes, last modified: 03.05.2022 - 100% done

Saving News.csv to News.csv

Saving NewsBisnes.csv to NewsBisnes.csv

Saving NewsCulture.csv to NewsCulture.csv

Saving NewsPolitics.csv to NewsPolitics.csv

Прочитаем данные и посмотрим первые 5 записей (рис.1):

sport = pd.read\_csv('/content/News.csv', delimiter = ';')

bisnes = pd.read\_csv('/content/NewsBisnes.csv', delimiter = ';')

culture = pd.read\_csv('/content/NewsCulture.csv', delimiter = ';')

politics = pd.read\_csv('/content/NewsPolitics.csv', delimiter = ';')

politics.head()

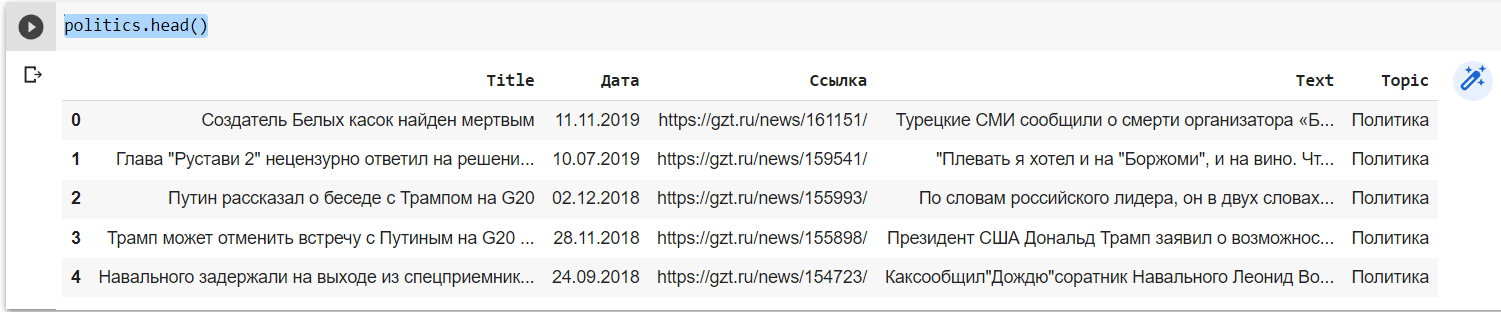


Рисунок 1 − Данные NewsPolitics.csv

Объединим данные в одну таблицу и проверим размерность:

# Создадим одну таблицу со всеми данными

news = pd.concat([sport, bisnes, culture, politics], axis=0)

print('Shape of the `sport` table: ', sport.shape)

print('Shape of the `bisnes` table: ', bisnes.shape)

print('Shape of the `culture` table: ', culture.shape)

print('Shape of the `politics` table: ', politics.shape)

print(news.shape)

Shape of the `sport` table: (224, 5)

Shape of the `bisnes` table: (57, 5)

Shape of the `culture` table: (455, 5)

Shape of the `politics` table: (758, 5)

(1494, 5)

Проверим: 224+57+455+758 = 1494. Все правильно.

Посмотрим информацию о нашем датасете:

news.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1494 entries, 0 to 757

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Title 1494 non-null object

1 Дата 1494 non-null object

2 Ссылка 1494 non-null object

3 Text 1494 non-null object

4 Topic 1494 non-null object

dtypes: object(5)

memory usage: 70.0+ KB

Проверим данные на наличие нулевых строк:

# Проверим данные на нулевые значения

news.isnull().sum()

Title 0

Дата 0

Ссылка 0

Text 0

Topic 0

dtype: int64

Перемешаем данные и выведем первые 5 строк датасета (рис.2):

# Перемешаем данные

news = shuffle(news).reset\_index(drop=True)

news.head()

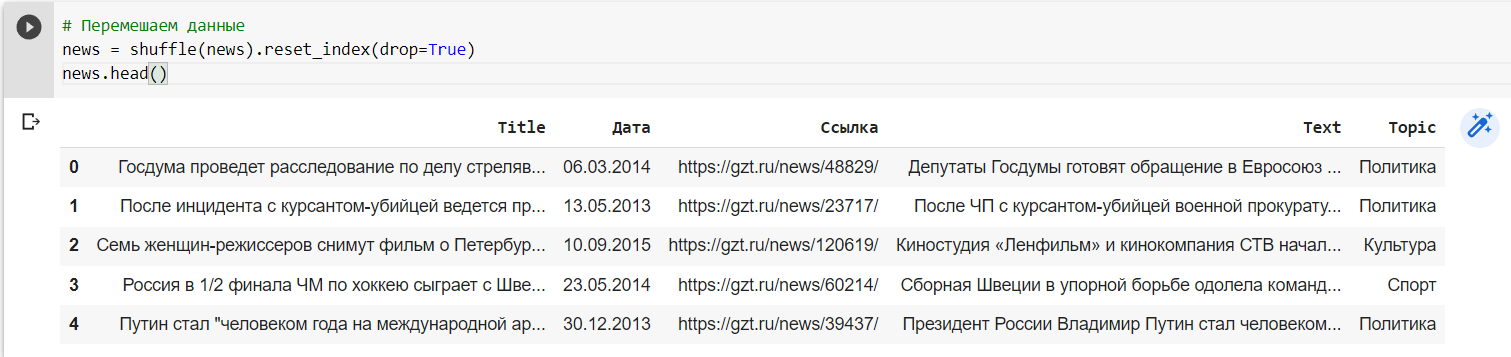


Рисунок 2 − Данные news

Выведем наши категории:

print(news['Topic'].unique(), len(news['Topic'].unique()))

['Политика' 'Культура' 'Спорт' 'Бизнес'] 4

На этом этап импорта данных закончен. Дальше нужно очистить данные.

Импортируем необходимые библиотеки:

import string

import re

import nltk

from nltk.stem import \*

from nltk.corpus import stopwords

from pymystem3 import Mystem

from string import punctuation

nltk.download('stopwords')

nltk.download('punkt')

nltk.download('wordnet')

Создадим функции для удаления пунктуации, цифр и пустот в тексте. Инициализируем переменную с русскими стоп-словами и дополним их. Стоп-слова − это слова, которые не несут никакой смысловой нагрузки, предлоги, местоимения, символы и т.д. Такие слова исключают из данных.

# import string

def remove\_punctuation(text):

    return "".join([ch if ch not in string.punctuation else ' ' for ch in text])

def remove\_numbers(text):

    return ''.join([i if not i.isdigit() else ' ' for i in text])

# import re

def remove\_multiple\_spaces(text):

  return re.sub(r'\s+', ' ', text, flags=re.I)

russian\_stopwords = stopwords.words("russian")

russian\_stopwords.extend(['…', '«', '»', '...'])

Применим функции и запишем данные в таблицу:

prep\_text = [remove\_multiple\_spaces(remove\_numbers(remove\_punctuation(text.lower()))) for text in tqdm(news['Text'])]

news['text\_prep'] = prep\_text

news.head(1)

На рис. 3 видим новый столбец с предобработанным текстом.

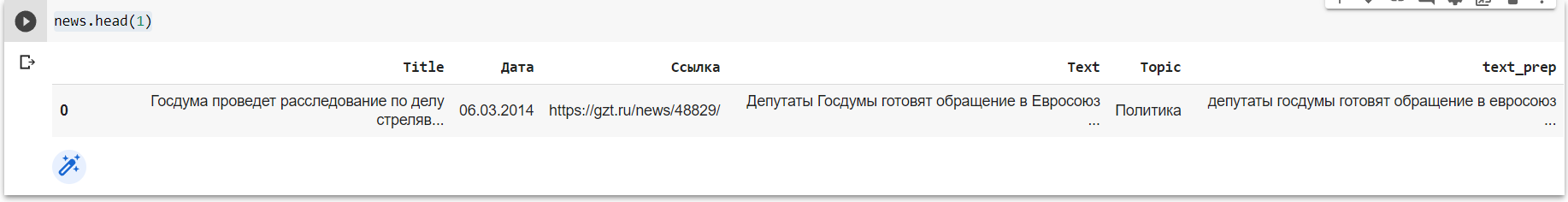


Рисунок 3 − test\_prep

Заключительным этапом предварительной обработки является стэмминг. Стэмминг — это процесс приведения слов к их корневым формам путем удаления суффиксов или префиксов, используемых со словом. Стэмминг делается для того, чтобы избежать создания признаков, которые семантически похожи, но синтаксически раличны.

from nltk.stem.snowball import SnowballStemmer

stemmer = SnowballStemmer("russian")

def remove\_stop\_words(text):

    tokens = word\_tokenize(text)

    tokens = [token for token in tokens if token not in russian\_stopwords and token != ' ']

    return " ".join(tokens)

from nltk import word\_tokenize

stemmed\_texts\_list = []

for text in tqdm(news['text\_prep']):

    tokens = word\_tokenize(text)

    stemmed\_tokens = [stemmer.stem(token) for token in tokens if token not in russian\_stopwords]

    text = " ".join(stemmed\_tokens)

    stemmed\_texts\_list.append(text)

news['text\_stem'] = stemmed\_texts\_list

news.head(2)

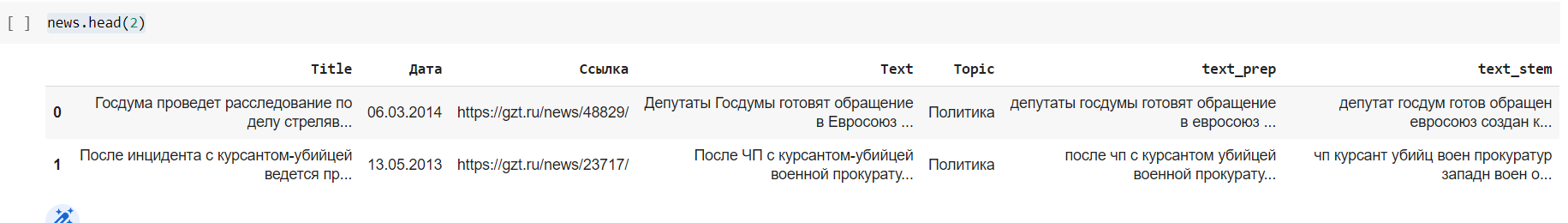


Рисунок 4 − text\_stem

Создадим список наших категорий и разобьем данные на обучающие и тестовые выборки:

my\_tags = news['Topic'].unique()

my\_tags

array(['Политика', 'Культура', 'Спорт', 'Бизнес'], dtype=object)

X = news['text\_stem']

y = news['Topic']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state = 42)

Параметр «test\_size» разделяем данные на тренировочную и тестовую выборки, 75% и 25% соотвественно. «random\_state» управляет перемешиванием данных перед применением разделения.

Применим логистическую регрессию для классификации текста:

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

logreg = Pipeline([('vect', CountVectorizer()),

                ('tfidf', TfidfTransformer()),

                ('clf', LogisticRegression(n\_jobs=1, C=1e5)),

               ])

Машины в отличие от людей, не могут понять сырой текст. Машины могут видеть только цифры. В частности, статические методы, такие как машинное обучение могут работать только с числами. Поэтому на нужно преобразовать наш текст в числа.

Модуль «Pipeline» сделает всю работу за нас. Класс «CountVectorizer» преобразует текстовые документы в соответствующие числовые объекты. Класс «TfidfTransformer» придаст веса словам. Класс «LogisticRegression» исполнит модель классификации.

Обучим нашу модель, передав тренировочные данные:

logreg.fit(X\_train, y\_train)

X\_train – наши подготовленные и отобранные тексты.

Y\_train – категории, к которым принадлежит каждый текст.

Это пример обучения модели с учителем.

Получим предсказания модели на тестовой выборке и оценим точность, сравнив с правильными ответами:

y\_pred = logreg.predict(X\_test)

print('accuracy %s' % accuracy\_score(y\_pred, y\_test))

print(classification\_report(y\_test, y\_pred,target\_names=my\_tags))

accuracy 0.7834224598930482

precision recall f1-score support

Политика 0.38 0.30 0.33 10

Культура 0.67 0.69 0.68 108

Спорт 0.82 0.84 0.83 195

Бизнес 0.91 0.87 0.89 61

accuracy 0.78 374

macro avg 0.70 0.67 0.68 374

weighted avg 0.78 0.78 0.78 374

Как видим, точность модели 78%, что очень хорошо, т.к. объем нашего датасета небольшой. Думаю, если было бы больше данных и равномерно распределенных по количеству, то результат был бы еще лучше.

Также приведены результаты точности по каждой категории.

Проверим обученную модель на новых текстах, которые я возьму из таких же категорий с других сайтов:

# Спорт

econ\_text = '''

Российский боксер Арест Саакян скончался через несколько дней после нокаута в бое на турнире «Короли нокаутов», сообщила сестра спортсмена Светлана Петросян на своей странице в Instagram.

Турнир состоялся 26 декабря в Тольятти. От полученных травм после нокаута Саакян впал в кому. У него был диагностирован отек головного мозга. Врачи несколько дней боролись за жизнь 26-летнего боксера, но он скончался, так и не придя в сознание.

'''

econ\_text = remove\_multiple\_spaces(remove\_numbers(remove\_punctuation(econ\_text.lower())))

econ\_text = remove\_stop\_words(econ\_text)

tokens = word\_tokenize(econ\_text)

stemmed\_tokens = [stemmer.stem(token) for token in tokens if token not in russian\_stopwords]

econ\_text = " ".join(stemmed\_tokens)

ect\_pred = logreg.predict([econ\_text])

ect\_pred

array(['Спорт'], dtype=object)

Еще один пример из политики:

# Политика

econ\_text = '''

США рассчитывают на сотрудничество с Россией по Международной космической станции по крайней мере до 2030 года, заявил глава NASA Билл Нельсон, его цитирует РИА «Новости».

«У нас есть все основания полагать, что русские будут [продолжать работу по] космической станции в ближайшем будущем», — сказал он на слушаниях в конгрессе.

Нельсон также отметил, что организация надеется на продолжение сотрудничества по программе до 2030 года и что его слова основываются на существующих между NASA и «Роскосмосом» «профессиональных отношений». Он также указал, что информация по поводу выхода России из программы МКС неверна.

Ранее глава «Роскосмоса» Дмитрий Рогозин сообщил, что Россия определилась со сроками завершения работы на МКС. Он не назвал сроки, но уточнил, что РФ продолжает работу на орбитальной станции до 2024 года. По мнению эксперта Института космических исследований РАН Натан Эйсмонт заявил, что эксплуатация МКС станет практически невозможной, если Россия выйдет из программы.

'''

ect\_pred = logreg.predict([econ\_text])

ect\_pred

array(['Политика'], dtype=object)

Как видим, модель верно предсказала категорию новостей. Можете провести собственные эксперименты с другими новостями.

**Вывод**

В результате выполнения данной лабораторной работы познакомились с классификацией текста. Выполнили загрузку данных и проверку перед очисткой. Очистили данные от стоп-слов, пунктуации и ненужных символов. Привели слова к корневой форме. Создали тестовую и тренировочную выборки. Создали, обучили и протестировали модель для классификации текста.

**Критерии оценки**

…