Пробный тур: Классификация английских и испанских текстов

Легенда:
Помогите роботу определить, на каком языке написаны короткие фразы: английском или испанском. Данные чистые — без опечаток и лишних символов.

Задание

Обучите модель, которая классифицирует текст на два класса:

- **0**: английский
- **1**: испанский

Данные

1. **Тренировочные данные (`train_trial.csv`)** со структурой:

   ```
   text,language
   "Hello world","english"
   ...
   ```

2. **Тестовые данные (`test_trial.csv`)** со структурой:
   ```
   id,text
   0,"Good night"
   1,"¿Cómo estás?"
   …
   ```

Требования к решению

1. Файл **`predictions_trial.csv`** в формате:

   ```
   id,predicted_class
   0,0
   1,1
   2,0
   ...
   ```

2. Краткий отчет **`report_trial.txt`** (2-3 предложения):
   - Как решали задачу?

- Какие признаки из текста выбрали?

Критерии оценки

- **10 баллов** за точность (1 балл за каждый правильный ответ).
- **5 баллов** за полноту описания методов, читаемость кода, оптимизацию алгоритмов в `report_trial.txt`.

Пример решения

```
import pandas as pd
import re

# Функция для очистки текста: оставляем только буквы и нижний регистр
def clean_text(text):
    # Удаляем все символы, кроме букв, заменяя их на пробелы
    text_clean = re.sub(r'[^a-zA-Z]', ' ', text)
    # Приводим к нижнему регистру и разбиваем на слова
    return text_clean.lower().split()

# Загрузка тренировочных данных
train_df = pd.read_csv("train_trial.csv")

# Создаем словари уникальных слов для каждого языка
english_vocab = set()
spanish_vocab = set()

for index, row in train_df.iterrows():
    words = clean_text(row['text'])
    if row['language'] == 'english':
        english_vocab.update(words)
    else:
        spanish_vocab.update(words)

# Функция для предсказания языка
def predict(text):
    words = clean_text(text)
    # Считаем совпадения с каждым словарем
    en_score = sum(1 for word in words if word in english_vocab)
    es_score = sum(1 for word in words if word in spanish_vocab)
    return 0 if en_score > es_score else 1

# Обработка тестовых данных
```

```
test_df = pd.read_csv("test_trial.csv")
predictions = []
for index, row in test_df.iterrows():
    lang = predict(row['text'])
    predictions.append({'id': row['id'], 'predicted_class': lang})

# Сохранение результатов
pd.DataFrame(predictions).to_csv("predictions_trial.csv", index=False)
```

Объяснение кода

1. **Очистка текста**:
    - `clean_text()` удаляет все не-буквенные символы (включая цифры и пунктуацию)
    - Текст разбивается на слова после очистки
2. **Создание словарей**:
    - Для английских текстов сохраняем все уникальные слова в `english_vocab`
    - Для испанских — в `spanish_vocab`
3. **Логика предсказания**:

- Для тестового текста подсчитываем, сколько слов есть в каждом словаре
- Выбираем язык с наибольшим числом совпадений

Цель задания: Познакомить участников с форматом задач и базовой обработкой текста. Удачи!

Задача 1. Классификация текстов на близкородственные языки с зашумленными данными

Вводная часть
Вы разрабатываете классификатор для текстов на четырех языках: испанском, португальском, индонезийском и малайском. Данные содержат шумы: случайные пробелы внутри слов, опечатки (например, замены символов на соседние клавиши) и некорректные символы. Ваша задача — создать модель, устойчивую к шумам и способную различать близкородственные языки (испанский/португальский и индонезийский/малайский).

Задание состоит из двух частей:

1. Обучение модели на данных с текстами и названиями языков.
2. Предсказание классов для зашумленных текстов из тестовой выборки.

Входные данные:

- Тренировочный файл **`train.csv`** (4000 текстов) с колонками:
  - **language**: название языка (spanish, portuguese, indonesian, malay).
  - **text**: текст с шумами (опечатки, пробелы внутри слов, некорректные символы).
- Тестовый файл **`test.csv`** (1000 текстов) с колонками:
  - **id**: уникальный идентификатор текста.
  - **text**: текст для классификации.

Особенности данных:

- Тексты содержат шумы: опечатки, пробелы внутри слов, некорректные символы.
- Близкородственные языки: испанский (0) и португальский (1), индонезийский (2) и малайский (3).

Требования к результатам:

1. **CSV-файл `predictions.csv`** с предсказаниями:
   - **id**: идентификатор из тестового файла.
   - **predicted_class**: числовой код языка:
     - 0: испанский
     - 1: португальский
     - 2: индонезийский
     - 3: малайский

Пример файла:

```
id,predicted_class
0,2
1,0
2,3
```

2. **Текстовый отчет `report_task1.txt`** с описанием:
    - Методы обработки шумов (например, коррекция раскладки клавиатуры).
    - Выбор модели и её параметров.
    - Стратегия различия близких языков (испанский/португальский, индонезийский/малайский).

Критерии оценки (100 баллов):

1. **Качество предсказаний (90 баллов):**
    - Тестовая выборка содержит 1000 текстов.
    - За каждый правильно классифицированный текст начисляется **0.09 балла**.
2. **Технические требования (10 баллов):**
    - Полнота описания методов, читаемость кода, оптимизация алгоритмов в `report_task1.txt` (10 баллов).

Ограничения:

- Запрещено использовать библиотеки автоматического определения языка (например, *langdetect, fasttext*).
- Разрешённые инструменты: *pandas, scikit-learn, numpy, torch.*

Советы:

- Для испанского/португальского: ищите уникальные артикли (например, "o" vs "el").
- Для индонезийского/малайского: анализируйте суффиксы (например, "-nya" в индонезийском).
- Для различия языков: анализируйте редкие грамматические конструкции или уникальные n-граммы.
- Шумы можно смягчить через регулярные выражения (удаление лишних символов, объединение слов).

Удачи в создании точного классификатора!

Задача 2. Формула цифрового счастья

Вводная часть
Уровень счастья (субъективного благополучия) в социологии — это интегральный показатель, включающий удовлетворенность жизнью, эмоциональный фон и социальную активность. Цифровой след пользователей социальных сетей (тексты, активность, визуальный контент) позволяет косвенно оценить эти параметры.

Цель:
Разработать математическую формулу, оценивающую уровень счастья на основе данных ВКонтакте пользователей из Республики Татарстан.

Для количественной оценки уровня счастья индивида на основе его цифрового следа в открытых источниках нужно разработать метрику "Индекс Счастья". Метрика должна быть основана на многомерном анализе социальной активности пользователя и учитывать ключевые компоненты, каждая из которых должна быть взвешена соответствующим коэффициентом важности. **Например**, можно предложить такую формулу:

$$Индекс\ Счастья = \alpha1 * S1 + \alpha2 * S2 + \alpha3 * S3,$$

где:

- $S1$ = количество друзей + количество подписчиков + количество групп + количество сообществ) - характеризует объем социальных связей пользователя;
- $S2$ = (количество постов + количество лайков на постах) - отражает общую социальную активность;
- $S3$ = количество положительных постов - определяется через семантический анализ текстового контента и характеризует эмоциональное состояние;
- $\alpha1$, $\alpha2$, $\alpha3$ - нормированные весовые коэффициенты, учитывающие региональные особенности социального поведения.

При расчете Индекса Счастья можно отметить, что каждая составляющая должна рассматриваться комплексно. Например, абсолютное количество друзей или постов само по себе не является достаточным показателем уровня счастья и требует дополнительной квалификации через взаимодействие (лайки, комментарии) и тональный анализ контента. Также требуется обязательная оценка тональности публикаций пользователя, поскольку простое наличие большого числа постов (например, 1000 негативно окрашенных сообщений) не может быть автоматически интерпретировано как признак социальной активности и, тем более, счастья. Анализируемая модель должна включать информацию о количестве и характере сообществ, в которых состоит пользователь, так как участие в различных группах является индикатором наличия у него конкретных интересов и увлечений, что является значимым фактором эмоционального благополучия.

Для повышения точности моделей оценки уровня счастья целесообразно проводить сегментацию пользователей, например, по гендерному и/или возрастному признакам, что обусловлено существенными различиями в их социальном поведении. Гендерная сегментация обоснована фундаментальными отличиями в использовании социальных платформ мужчинами и женщинами. Согласно исследованию, женщины демонстрируют более высокую степень вовлеченности в социальные сети, характеризуемую большим временем пребывания на платформе и большей вероятностью авторства публичных сообщений (минимум одно сообщение в месяц). Возрастная сегментация основывается на особенностях взаимодействия различных демографических групп с социальной платформой. Особенно значимые различия отмечаются среди молодых пользователей (18–25 лет), которые характеризуются меньшей пользовательской активностью. Возрастные особенности также проявляются в стратегиях самопрезентации.

Входные данные

1. **API VK.com** для сбора данных:
    - Разрешено использовать **только открытые данные** с соблюдением этических норм:
    - **Количественные показатели**, например:
        - Количество друзей/подписчиков, групп, постов за последний год.
        - Активность: лайки, комментарии, репосты (свои и полученные).
        - Соотношение "друзья vs. подписчики".
    - **Качественные показатели**, например:
        - Тональность текстовых постов (анализ ключевых слов, эмодзи, смайлов).
        - Визуальный контент: преобладание фото с людьми/одиночные селфи/мемы.
        - Тематика групп (хобби, образование, мемы, токсичные сообщества).
    - **Социальные паттерны**, например:
        - Частота взаимных комментариев с одними и теми же пользователями.
        - Активность в ночное время (22:00–6:00).
        - Упоминания мест/событий из реальной жизни.
2. **Готовые инструменты для анализа:**
    - Для текстов, например:
        - Библиотека *Dostoevsky* для анализа тональности русского текста
        - Библиотека *NLTK* для базовой обработки текста
        - Модель *RuBERT* для получения эмбеддингов текста
        - Библиотека *DeepPavlov* с готовыми моделями тональности
    - Для изображений: *OpenCV* или *TensorFlow* (анализ эмоций на фото через предобученные модели).

Требования к результатам:

1. Все данные сохранять в **JSON-файл `data.json`** с подобной структурой:

```
{
 "id": "1142588",
 "city": "Kazan",
 "age": 25,
 "gender": "female",
 "university": "abc",
 "photos_num": 123,
 "videos_num": 123,
 "audio_num": 123,
 "self_posts_num": 123,
 "reposts_num": 123,
 "followers_num": 123,
 "groups_num": 123,
 "friends_num": 123,
 …
 "posts": {
   "id1": {
     "text": "abc",
     "date": "11.11.2011 11:11:11"
   },
   …
 },
 "groups": {
   "groupid1": {
     "name": "abc",
     "description": "abc abc ..."
   },
   …
 },
 "friends": [id11, id12, id13, ...],
 …
 }
```

2. **Текстовый отчет `report_task2.pdf`** с описанием:
   - особенностей сбора данных;
   - предобработки данных (очистка текстов, обработка изображений);
   - инструментов анализа (использованные библиотеки, алгоритмы анализа тональности, классификации изображений);
   - листинга основного кода;

- математического представления формулы с пояснениями;
- графиков (распределение индекса счастья, зависимость индекса от возраста, пола и т.д.);
- интерпретации результатов.

Критерии оценки (100 баллов):

1. Сбор данных (15 баллов)
   - Собрать данные не менее **100 пользователей** из Республики Татарстан (чем больше пользователей, тем лучше статистика).
   - Сохранить в JSON-формате.
2. Разработка формулы (50 баллов)
   - Создать математическую формулу, которая на основе данных ВК оценивает уровень счастья.
   - **Обязательные компоненты формулы:**
     - Тональность текстов (позитивные/негативные слова).
     - Социальная активность (лайки, комментарии).
     - Участие в тематических группах (хобби, образование).
   - **Параметры качественной формулы:**
     - Нормирование всех компонентов к единой шкале (например, от 0 до 1)
     - Учет веса каждого фактора с обоснованием выбора весов
     - Математическая корректность и интерпретируемость результатов
     - Устойчивость к выбросам в данных
     - Учет социально-психологических аспектов счастья (напр., частота социальных взаимодействий)
   - **Опционально:** анализ изображений через OpenCV.
3. Визуализация (20 баллов)
   - Построить графики, например:
     - Распределение уровня счастья.
     - Зависимость счастья от активности/тематики групп.
4. Интерпретация (15 баллов)
   - Объяснить, как формула связана с социологическим понятием счастья.
   - Указать ограничения: например, посты могут маскировать реальные эмоции.

Ограничения:

- Запрещено использовать технологии генеративного искусственного интеллекта на любых этапах решения задачи.

Примеры

Пример скрипта для сбора данных:

```
import vk_api  # Импортируем библиотеку для работы с VK API
import json  # Импортируем модуль для работы с JSON

# Создаем объект VkApi, используя токен доступа
vk_session = vk_api.VkApi(token='ВАШ_ТОКЕН')
vk = vk_session.get_api()  # Получаем объект API для выполнения запросов

# Список ID групп, связанных с целевым городом (например, Казань)
group_ids = [
    57867786,  # Пример: https://vk.com/vkazani
    # Добавьте другие актуальные группы города
]

members = []  # Создаем пустой список для хранения участников групп

# Проходим по каждому ID группы и получаем ее участников
for group_id in group_ids:
    try:
        # Задержка в 0.5 секунды для избежания слишком частых запросов
        sleep(0.5)

       # Получаем участников группы с дополнительными полями (город и дата
рождения)
        chunk = vk.groups.getMembers(
            group_id=group_id,  # ID группы
            fields="city,bdate",  # Поля, которые хотим получить
            count=1000  # Максимальное количество участников за один запрос
        )

        # Добавляем полученных участников в общий список
        members.extend(chunk['items'])

    except vk_api.exceptions.ApiError as e:
        # Обработка ошибок API (например, если группа закрыта или удалена)
        print(f"Ошибка для группы {group_id}: {e}")
        continue  # Переходим к следующей группе

# Проходим по каждому пользователю из списка участников
for user in members:
    # Проверяем наличие информации о городе
```

```
    if 'city' not in user:
        continue  # Пропускаем пользователя, если нет информации о городе

    # Проверяем наличие даты рождения
    if 'bdate' not in user:
        continue  # Пропускаем пользователя, если нет даты рождения
    # Разбиваем дату рождения на части (день, месяц, год)
    bdate = user['bdate'].split('.')
    if len(bdate) < 3:
        continue  # Пропускаем, если дата рождения не содержит года

    # Выводим информацию о пользователе (ID, дата рождения, город)
    print(user['id'], user['bdate'], user['city']['title'])

    # Добавляем ID пользователя в список подходящих пользователей
    PT_users.append(int(user['id']))
    print('--------------')  # Разделитель для удобства чтения вывода


# Функция для получения последних постов пользователя
def get_posts(user_id):
    try:
        # Получаем последние 5 постов пользователя со стены
        posts = vk.wall.get(owner_id=user_id, count=5)["items"]
        # Возвращаем тексты постов
        return [post["text"] for post in posts]
    except vk_api.exceptions.ApiError as e:
        # Обработка ошибок API (например, если стена закрыта)
        print(f"Ошибка для пользователя {user_id}: {e}")

users_data = {}  # Словарь для хранения данных о пользователях

# Пример списка ID пользователей Татарстана (заменить на реальные)
user_ids = [879258, ]

# Проходим по каждому ID пользователя
for user_id in user_ids:
    users_data[user_id] = {
        "posts": get_posts(user_id),  # Получаем последние посты
пользователя
        # Здесь можно добавить сбор других данных (например, лайки,
подписки и т.д.)
    }

# Сохраняем собранные данные в файл data.json
with open("data.json", "w") as f:
```

```
    # Записываем данные в JSON-формате, сохраняя русские символы
    json.dump(users_data, f, ensure_ascii=False)
```

Примеры использования библиотек для анализа тональности

```
# Пример использования RuBERT
from transformers import pipeline
classifier = pipeline("sentiment-analysis",
model="blanchefort/rubert-base-cased-sentiment")
text = "Я счастлив!"
result = classifier(text)  # {'label': 'POSITIVE', 'score': 0.98}
```

Счастье — это алгоритм, который пишется между строк. Ваша задача — найти его паттерны в цифровом шуме.