7. Практическое занятие: Определение свойств слов и основы модели TF-IDF

Цель занятия

Научиться извлекать количественные характеристики слов и освоить алгоритм TF-IDF для определения статистической значимости слов в коллекции документов.

Теоретический минимум

Модель TF-IDF (Term Frequency — Inverse Document Frequency) используется для оценки важности слова в контексте документа, являющегося частью коллекции.

1. TF (Term Frequency): Частота слова. Насколько часто слово встречается в конкретном документе.

$$TF(t,d) = \frac{\text{Количество вхождений слова } t \text{ в документе } d}{\text{Общее количество слов в документе } d}$$

2. IDF (Inverse Document Frequency): Обратная частота документа. Уменьшает вес слов, которые встречаются слишком часто во всех документах (например, «и», «в», «что»), и повышает вес уникальных слов.

$$IDF(t,D) = \log \frac{\text{Общее количество документов в корпусе } D}{\text{Количество документов, где встречается слово } t}$$

Задание 1. Определение свойств слов (Длина, Частота)

Прежде чем переходить к сложным моделям, необходимо научиться извлекать базовые свойства токенов.

Инструкция:

Напишите скрипт, который для каждого слова в предложении вычисляет его длину и частоту появления.

```
from collections import Counter

text = "программирование это интересно программирование это будущее"
words = text.split()
```

```
# Свойства слов
word_counts = Counter(words)

print(f"{'Слово':<15} | {'Длина':<10} | {'Частота':<10}")
for word in set(words):
    print(f"{word:<15} | {len(word):<10} | {word_counts[word]:<10}")
```

Задание 2. Ручной расчет TF

Представьте, что у вас есть документ: "данные это нефть данные это сила".

Инструкция:

Рассчитайте TF для слова «данные».

1. Количество слов в документе: 6.
2. Количество вхождений слова «данные»: 2.
3. TF = 2 / 6 = 0.33.

Задание 3. Реализация TF-IDF с помощью Scikit-Learn

В реальных задачах для расчета TF-IDF используются оптимизированные библиотеки.

Инструкция:

Используйте TfidfVectorizer для анализа небольшой группы текстов.

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Корпус документов
corpus = [
    "алгоритмы машинного обучения очень эффективны",
    "машинное обучение это будущее технологий",
    "алгоритмы и структуры данных важны для программиста"
]

# Инициализация векторайзера
vectorizer = TfidfVectorizer()

# Расчет TF-IDF
tfidf_matrix = vectorizer.fit_transform(corpus)

# Преобразование в таблицу для наглядности
feature_names = vectorizer.get_feature_names_out()
df_tfidf = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)

print(df_tfidf)
```

Задание 4. Анализ результатов

Посмотрите на полученную таблицу из Задания 3.

1. Найдите слово «алгоритмы». Сравните его вес в первом и третьем документах.
2. Почему слова, которые встречаются во всех документах (если бы такие были), имеют низкий показатель TF-IDF?

Задание 5. Применение TF-IDF для поиска ключевых слов

Напишите функцию, которая принимает документ из корпуса и выводит топ-3 самых важных слова на основе весов TF-IDF.

```
def get_top_keywords(df, doc_index, top_n=3):
    row = df.iloc[doc_index]
    return row.sort_values(ascending=False).head(top_n)

print("Топ-3 слова для первого документа:")
print(get_top_keywords(df_tfidf, 0))
```

Задание 6. Визуализация важности слов с помощью WordCloud Одним из самых эффективных способов быстро оценить тематику корпуса или важность слов является «Облако слов» (WordCloud), где размер слова пропорционален его весу.

Инструкция: Используйте библиотеку `wordcloud` для создания визуализации на основе вычисленных ранее весов TF-IDF.

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Генерируем словарь "слово: вес" на основе средних значений TF-IDF по всем документам
weights = df_tfidf.mean(axis=0).to_dict()

# Создание объекта облака слов
wordcloud = WordCloud(width=800, height=400, background_color='white',
                      colormap='viridis').generate_from_frequencies(weights)

# Отображение
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off') # Отключаем оси
plt.title("Облако слов на основе весов TF-IDF")
plt.show()
```

Задание 7. Анализ «свойств» через распределение весов Изучите, как длина слова (свойство, изученное в Задании 1) соотносится с его значимостью (TF-IDF).

Инструкция: Постройте диаграмму рассеяния (Scatter Plot), где по оси X — длина слова, а по оси Y — его средний вес TF-IDF.

```
import seaborn as sns

# Подготовка данных
words_data = pd.DataFrame({
    'word': feature_names,
    'length': [len(w) for w in feature_names],
    'importance': df_tfidf.mean(axis=0).values
})

# Визуализация
plt.figure(figsize=(8, 6))
sns.scatterplot(data=words_data, x='length', y='importance')
for i in range(words_data.shape[0]):
    plt.text(words_data.length[i]+0.2, words_data.importance[i],
words_data.word[i])

plt.title("Связь длины слова и его значимости")
plt.xlabel("Длина слова (количество символов)")
plt.ylabel("Средний вес TF-IDF")
plt.grid(True)
plt.show()
```

Задание 8. Сравнение свойств: Частотность vs TF-IDF Часто случается так, что самое частое слово в тексте (высокий TF) имеет нулевую значимость (низкий TF-IDF) из-за своей распространенности во всем корпусе.

Инструкция: Найдите в вашем корпусе слово, которое встречается в 2-х и более документах. Сравните его позицию в рейтинге по частоте (Count) и по весу TF-IDF. Запишите вывод: почему его «важность» снизилась?

Контрольные вопросы

1. В чем главное преимущество TF-IDF перед обычным подсчетом слов (Bag of Words)?
2. Как изменится значение IDF для слова, которое встречается в каждом документе библиотеки?
3. Почему при расчете TF-IDF часто используют логарифмирование в формуле IDF?

Итог работы

Студенты изучили математическую основу и программную реализацию модели TF-IDF. Этот метод позволяет переходить от текста к числам (векторизация), выделяя наиболее значимые признаки для задач поиска, классификации и суммаризации текстов.