8. Практическое занятие: Сравнение схожих слов. Лингвистическое и математическое решение

Цель занятия

Изучить различные подходы к измерению близости между словами: от лингвистических метрик (редакторское расстояние) до математических методов векторного сходства.

Теоретический минимум

Сравнение слов может происходить на двух уровнях:

1. **По написанию (Посимвольное сходство):** Насколько близки слова по составу букв. Основная метрика — **расстояние Левенштейна** (количество правок: вставок, удалений и замен для превращения одного слова в другое).
2. **По смыслу (Семантическое сходство):** Насколько близки слова по значению. Используется **косинусное сходство** между векторами слов в многомерном пространстве.

Задание 1. Лингвистическое решение: Расстояние Левенштейна

Этот метод незаменим для задач проверки орфографии (Spell Checking) и поиска нечетких дубликатов.

Инструкция:

Используйте библиотеку Levenshtein (или аналогичную), чтобы вычислить расстояние между парами слов.

```
# Установка: pip install python-Levenshtein
import Levenshtein

pairs = [("кошка", "кошки"), ("программист", "программа"), ("мама",
"папа")]

print(f"{'Пара слов':<25} | {'Расстояние'}")
for s1, s2 in pairs:
    dist = Levenshtein.distance(s1, s2)
    print(f"{s1 + ' - ' + s2:<25} | {dist}")
```

Задание 2. Коэффициент Жаккара (Jaccard Similarity)

Это математическое решение на основе множеств. Оно определяет сходство как отношение размера пересечения множеств символов к размеру их объединения.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Инструкция:

Реализуйте функцию расчета коэффициента Жаккара для сравнения состава букв.

```
def jaccard_sim(str1, str2):
    a = set(str1)
    b = set(str2)
    return len(a.intersection(b)) / len(a.union(b))

print(f"Сходство Жаккара ('интеллект', 'интеллигент'):
{jaccard_sim('интеллект', 'интеллигент'):.2f}")
```

Задание 3. Математическое решение: Косинусное сходство векторов

Когда слова представлены в виде векторов (например, через TF-IDF или Word2Vec), их близость измеряется косинусом угла между ними. Значение 1.0 означает полную идентичность направлений векторов.

Инструкция:

Вычислите косинусное сходство между двумя документами, используя результаты векторизации TF-IDF.

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

docs = [
    "искусственный интеллект это будущее",
    "интеллект и нейросети изменят наш мир",
    "сегодня на улице идет сильный дождь"
]

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(docs)

# Сравнение первого документа со всеми остальными
similarity = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)

print("Матрица сходства первого документа с остальными:")
for i, sim in enumerate(similarity[0]):
    print(f"Док 1 и Док {i+1}: {sim:.2f}")
```

Задание 4. Анализ результатов: Смысл vs Написание

Сравните пары слов: «замок» (строение) и «замок» (дверной).

1. Какое будет расстояние Левенштейна?

2. Будет ли оно отличаться от расстояния между словами «замок» и «замок» с разным смыслом?
3. **Вывод:** Объясните, почему посимвольное сходство бессильно в задачах омонимии.

Задание 5. Практический кейс: Поиск похожих товаров

Представьте, что пользователь ищет «iPhone 15», а в базе есть «iPhone 15 Pro».

1. Рассчитайте расстояние Левенштейна.
2. Рассчитайте коэффициент Жаккара.
3. Какая метрика, по вашему мнению, лучше подходит для реализации функции «Возможно, вы имели в виду...»?

Контрольные вопросы

1. В чем разница между $distance=0$ в Левенштейне и $similarity=1.0$ в косинусном сходстве?
2. Почему для сравнения длинных текстов косинусное сходство эффективнее расстояния Левенштейна?
3. Как нормализация (лемматизация) влияет на точность косинусного сходства?

Итог работы

Обучающиеся освоили инструментарий для сравнения текстовых единиц. Теперь они могут выбирать между «лингвистической» точностью правки символов и «математической» мощью векторных пространств в зависимости от бизнес-задачи.