5. Практическое занятие: Упрощение формы слов. Стемминг и лемматизация

Цель занятия

Освоить методы приведения слов к их базовым формам, изучить различия между алгоритмическим отсечением окончаний (стемминг) и морфологическим анализом (лемматизация), а также научиться применять их для снижения размерности данных в NLP.

Теоретический минимум

В естественных языках одно и то же слово может иметь десятки форм (например: идти, шёл, идущий). Для компьютера это разные токены. Чтобы объединить их смысл, используют:

1. **Стемминг (Stemming):** грубое отсечение суффиксов и окончаний по правилам языка. Результат (стем) не всегда является реальным словом.
2. **Лемматизация (Lemmatization):** приведение слова к канонической форме (лемме) с использованием словаря и учетом части речи.

Задание 1. Реализация стемминга на Для русского языка классическим алгоритмом является стеммер Портера (Snowball Stemmer).

Инструкция:

Используйте библиотеку nltk для обработки списка слов.

```
from nltk.stem import SnowballStemmer

# Инициализация стеммера для русского языка
stemmer = SnowballStemmer("russian")

words = ["бежать", "бегу", "бежал", "бегущий", "красивый", "красивого",
"красивыми"]

# Применение стемминга
stems = [stemmer.stem(word) for word in words]

for word, stem in zip(words, stems):
    print(f"{word} -> {stem}")
```

Задание 2. Лемматизация с учетом морфологии

Для качественной лемматизации в русском языке часто используется библиотека pymorphy2, которая возвращает начальную форму (именительный падеж, единственное число или инфинитив).

Инструкция:

Преобразуйте те же слова в леммы и сравните результат со стеммингом.

```
import pymorphy2

morph = pymorphy2.MorphAnalyzer()

# Функция получения леммы
def lemmatize(word):
    p = morph.parse(word)[0]
    return p.normal_form

lemmas = [lemmatize(word) for word in words]

for word, lemma in zip(words, lemmas):
    print(f"{word} -> {lemma}")
```

Задание 3. Сравнительный анализ: Стемминг vs Лемматизация

Заполните таблицу в тетради, проанализировав результаты выполнения Заданий 1 и 2.

Исходное слово	**Результат стемминга**	**Результат лемматизации**	**Качество (осмысленность)**
Бегущий	бегущ	бежать	
Красивыми	красив	красивый	
Организация	организ	организация	

Задание 4. Проблема «сверхстемминга» (Overstemming)

Стеммеры работают по правилам, а не по смыслу. Иногда они могут свести разные слова к одной основе, теряя смысл.

Инструкция:

Проверьте, к каким основам стеммер приведет слова «организация» и «орган». Объясните, почему для поисковых систем это может стать проблемой.

```
test_words = ["организация", "орган"]
print([stemmer.stem(w) for w in test_words])
```

Задание 5. Обработка целого предложения

Напишите скрипт, который принимает предложение, очищает его от знаков препинания и возвращает строку из лемм (нормализованный текст).

```
def normalize_sentence(text):
    # Очистка и токенизация
    words = re.sub(r'[^а-яё\s]', '', text.lower()).split()
    # Лемматизация каждого слова
    res = [lemmatize(w) for w in words]
    return " ".join(res)

sentence = "Мама мыла раму, а студенты изучали программирование."
print("Нормализованная строка:", normalize_sentence(sentence))
```

Контрольные вопросы

1. В каком случае стемминг предпочтительнее лемматизации? (Подсказка: скорость обработки).
2. Почему лемматизация слова «стекло» может быть неоднозначной без контекста?
3. Какую роль играет нормализация слов в задачах классификации текстов (например, определение спама)?

Итог работы

Студенты научились приводить слова к базовым формам, понимая разницу между механическим отсечением окончаний и полноценным лингвистическим анализом. Эти навыки критически важны для уменьшения «проклятия размерности» в векторных моделях текста.