4. Практическое занятие: Модели N-грамм и Метрика Perplexity

Цель занятия

Изучить принцип работы статистических языковых моделей на основе N-грамм, научиться рассчитывать вероятности последовательностей слов и оценивать качество модели с помощью метрики Perplexity (перплексия).

Задание 1. Построение униграмм и биграмм

N-грамма — это последовательность из n соседних элементов (слов или символов). Модель N-грамм предсказывает следующее слово, основываясь на $n-1$ предыдущих словах.

Шаги выполнения:

1. Возьмите очищенный текст.
2. Разбейте его на токены.
3. Сформируйте список биграмм.

```
text = "мама мыла раму мама мыла окно"
tokens = text.split()

# Создание биграмм
bigrams = [(tokens[i], tokens[i+1]) for i in range(len(tokens)-1)]

print("Токены:", tokens)
print("Биграммы:", bigrams)
```

Задание 2. Расчет вероятностей биграмм

Вероятность слова w_2 при условии предыдущего слова w_1 рассчитывается по формуле:

$$P(w_2|w_1) = \frac{Count(w_1, w_2)}{Count(w_1)}$$

Инструкция:

Вычислите вероятность того, что после слова "мама" идет слово "мыла".

```
from collections import Counter
```

```
word_counts = Counter(tokens)
bigram_counts = Counter(bigrams)

w1, w2 = "мама", "мыла"
probability = bigram_counts[(w1, w2)] / word_counts[w1]

print(f"P({w2} | {w1}) = {probability}")
```

Задание 3. Понятие Perplexity (Перплексия)

Перплексия — это метрика, которая показывает, насколько хорошо языковая модель предсказывает выборку. Чем ниже значение перплексии, тем лучше модель «понимает» структуру текста и тем меньше она «запутана» при выборе следующего слова.

Теоретическая справка:

Если у нас есть последовательность $W = w_1, w_2, \dots, w_N$, то перплексия вычисляется как:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Задание:

Представьте, что у вас есть две модели. Модель А предсказывает следующее слово в тесте с вероятностью 0.5, а Модель Б — с вероятностью 0.1. Вычислите, у какой модели перплексия будет ниже, и объясните почему.

Задание 4. Проблема нулевых вероятностей (Smoothing)

Если в тестовом наборе данных встречается биграмма, которой не было в обучающем наборе, вероятность всей фразы станет равной нулю, а перплексия уйдет в бесконечность.

Инструкция:

Изучите концепцию сглаживания Лапласа (Add-one smoothing). Суть метода — добавить единицу к числителю и размер словаря к знаменателю, чтобы избежать деления на ноль.

```
# Формула со сглаживанием:
# P = (count(w1, w2) + 1) / (count(w1) + V)
# где V — размер уникального словаря
```

Задание 5. Программный расчет вероятности предложения

Вероятность целого предложения в модели биграмм — это произведение условных вероятностей всех его пар слов. Чтобы избежать исчезающе малых чисел при умножении вероятностей, на практике часто используют логарифмирование.

Инструкция:

Напишите функцию, которая вычисляет вероятность короткой фразы на основе частотного словаря, созданного в Задании 2.

Python
```
def sentence_probability(sentence, bigram_counts, word_counts):
    tokens = sentence.split()
    prob = 1.0
    for i in range(len(tokens) - 1):
        w1, w2 = tokens[i], tokens[i+1]
        # Используем простейшее сглаживание, если пара не найдена
        count_w1_w2 = bigram_counts.get((w1, w2), 0) + 1
        count_w1 = word_counts.get(w1, 0) + len(word_counts)
        prob *= (count_w1_w2 / count_w1)
    return prob

test_phrase = "мама мыла окно"
print(f"Вероятность фразы '{test_phrase}': {sentence_probability(test_phrase,
bigram_counts, word_counts)}")
```

Задание 6. Вычисление Perplexity (Перплексии)

Перплексия — это обратная вероятность тестового набора, нормированная на количество слов. Она показывает, насколько эффективно модель сжимает данные.

Инструкция:

Реализуйте расчет перплексии для тестовой фразы. Помните: $PP(W) = \sqrt[N]{\frac{1}{P(w_1...w_N)}}$, где N — количество слов.

Python
```
import math

def calculate_perplexity(sentence, bigram_counts, word_counts):
    tokens = sentence.split()
    N = len(tokens)
    prob = sentence_probability(sentence, bigram_counts, word_counts)

    # Расчет через корень N-ой степени
    perplexity = pow(1/prob, 1/N)
    return perplexity

pp = calculate_perplexity("мама мыла окно", bigram_counts, word_counts)
print(f"Perplexity модели на тестовой фразе: {pp:.2f}")
```

Задание 7. Анализ зависимости N и Perplexity

Проведите эксперимент: сравните значения перплексии для двух предложений:

1. Типичного для вашей обучающей выборки (например, "мама мыла раму").

2. Нетипичного или случайного (например, "окно мыла мама").

Инструкция:

Сравните результаты. Почему во втором случае перплексия выше? Сделайте вывод о том, как порядок слов влияет на оценку модели.

Контрольные вопросы

1. Как изменится количество N-грамм, если мы перейдем от биграмм (n=2) к триграммам (n=3)?
2. Почему модель с перплексией 10 лучше, чем модель с перплексией 100?
3. Какую роль играет специальный токен `<s>` (начало предложения) в расчетах вероятностей N-грамм?

Итог работы

В результате выполнения задания обучающиеся осваивают математический аппарат статистических моделей текста, учатся работать с частотными словарями и оценивать предсказательную способность модели через метрику перплексии.