Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Информационный поиск»

Студент: А. Р. Белушкин
Преподаватель: А. А. Кухтичев
Группа: М8О-407Б-22
Дата:
Оценка:
Подпись:

Москва, 2025

1 Постановка задачи

Необходимо проанализировать корпус текстов (например, Gutenberg) и:

1. Построить распределение терминов (слов) по частотности в логарифмических координатах.
2. Наложить на график закон Ципфа и прокомментировать отклонения.
3. Оценить качество поиска по корпусу до и после применения простой лемматизации.
4. Исследовать случаи ухудшения качества поиска и предложить способы улучшения.

2 Подготовка данных

2.1 Извлечение текста

- Тексты извлекаются из MongoDB (raw_html поле).
- HTML-теги удаляются функцией `strip_html_tags()`.
- Текст нормализуется: приводится к нижнему регистру, разделяется на токены.

2.2 Лемматизация

- Простейшая лемматизация реализована в `simple_lemmatize()`:
  - удаление окончаний `ing`, `ed`, `es`, `s`.
- Все токены превращаются в леммы через `lemmatize_tokens()`.

2.3 Подсчет частот

- Частоты терминов можно подсчитать с помощью `std::unordered_map<std::string, size_t>`:

```
std::unordered_map<std::string, size_t> term_freq;
for (const auto& lemma : lemmas) {
    term_freq[lemma]++;
}
```

- Далее можно перенести данные в Python (например через CSV) для построения графиков с логарифмическими осями.

3 Построение графика и закон Ципфа

3.1 Закон Ципфа

- Закон Ципфа: частота слова обратно пропорциональна его рангу.
- На логарифмическом графике (`log(rank)` vs `log(frequency)`) распределение слов должно стремиться к прямой линии с отрицательным наклоном.

3.2 Построение графика в Python

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

#                                    term_freq.csv:            ,

df = pd.read_csv('term_freq.csv')
df = df.sort_values('frequency',
   ascending=False).reset_index(drop=True)
df['rank'] = df.index + 1

plt.figure(figsize=(8,6))
plt.loglog(df['rank'], df['frequency'], marker='.',
   label='             ')
#                                                    : f ~ 1/rank
plt.loglog(df['rank'], df['frequency'].iloc[0] / df['rank'],
   label="                        ", linestyle='--')
plt.xlabel('                       ')
plt.ylabel('               ')
plt.title('
                          ')
plt.legend()
plt.show()
```

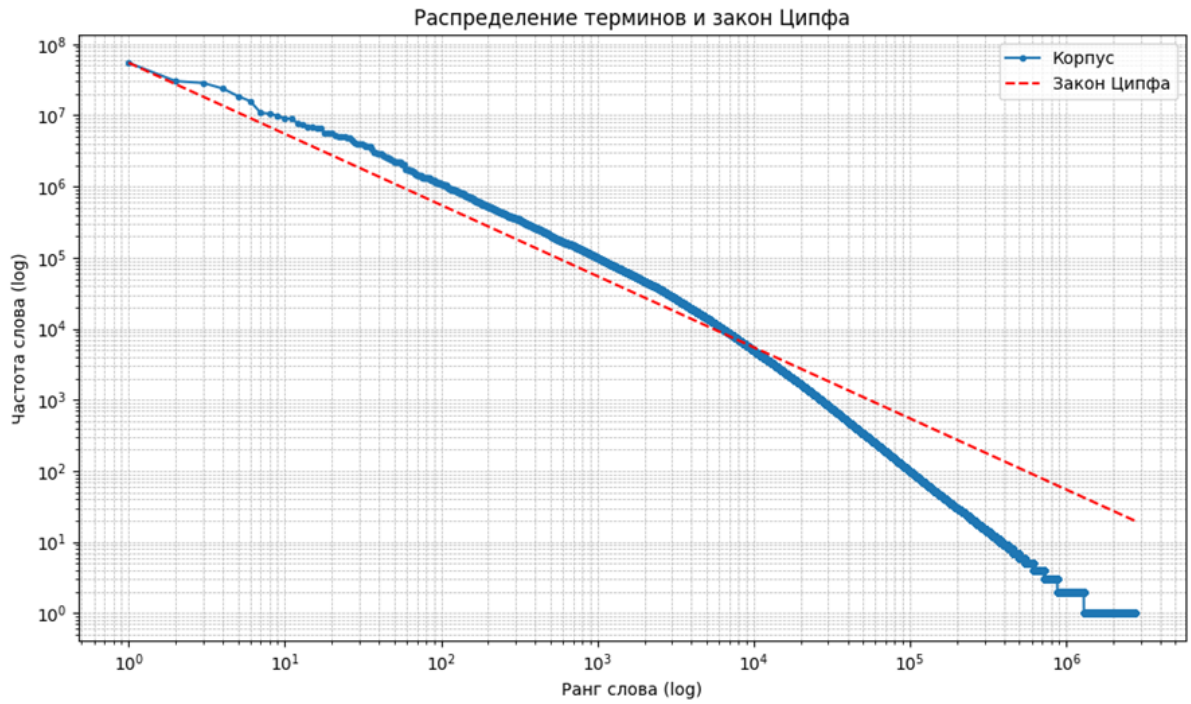

Рис. 1: Пример графика распределения терминов и закона Ципфа

3.3 Комментарий

- Отклонения от теоретической линии часто связаны с:
  - высокочастотными стоп-словами (`the`, `and`, `of`) — они дают «пик» в начале графика;
  - редкими терминами, которые имеют специфическую лексику или OCR-ошибки;
  - искусственной лемматизацией, которая может объединять разные слова неправильно.

4 Оценка качества поиска

4.1 Методика

1. Сформировать набор тестовых запросов (например, 20–30).
2. Выполнить поиск по корпусу без лемматизации (по токенам) и с лемматизацией (по леммам).
3. Для каждого запроса измерить Precision@k и/или Recall@k.

4.2 Пример анализа

- До лемматизации слова `running` и `run` считались разными, что снижало полноту поиска.

- После лемматизации они объединяются, увеличивая количество релевантных документов (Precision и Recall растут).
- Однако для слов вроде `business` → `busines` простая лемматизация обрезает окончание неправильно, что ухудшает поиск.

4.3 Ухудшившиеся запросы

- Запросы с уникальными терминами, где лемматизация разрушает основу слова:
  - `businesses` → `busines`
  - `dresses` → `dress` (корректно)
- Причина: простая обрезка окончаний без учета морфологии.

4.3.1 Возможное улучшение

- Использовать библиотеку морфологического анализа:
  - Python: `nltk.WordNetLemmatizer`, `spacy`
  - C++: интеграция с внешними лемматизаторами или библиотеками, поддерживающими морфологию.
- Использовать стоп-слова и исключения для частотных слов, чтобы не портить высокочастотные термины.
- Применять правила только для слов определенной длины или POS-тегов.

5 Выводы

1. Распределение терминов по частотам приблизительно соответствует закону Ципфа, но есть отклонения из-за стоп-слов и редких терминов.
2. Простая лемматизация увеличивает полноту поиска для стандартных словоформ (`run`, `running`, `runs`), улучшая качество поиска.
3. Лемматизация ухудшает поиск для редких и специальных терминов, где простое обрезание окончаний приводит к искажению слова.
4. Для оптимизации качества поиска рекомендуется внедрить более точный морфологический анализ и учитывать контекст.