Применение инструментов анализа семантики языков к поиску неточных повторов в документации ПО

Соловьев Александр
группа 344
научный руководитель: к.ф.-м.н., ст. преп., Д.В. Луцив

СПбГУ
кафедра системного программирования

01 октября 2018 г.

Введение

- Сложность поддержания документации растет с увеличением сложности программного обеспечения;
- Одной из проблем является дублирующаяся информация;
- Существуют различные решения для нахождения точных копий;
- Существует алгоритм поиска неточных повторов, основанный на N-граммной модели;
- Не учитывается семантика языка.

Постановка задачи

Цель: изучение применимости инструментов анализа семантики языков к алгоритму поиска неточных повторов на основе N-граммной модели.

- реализация алгоритма поиска неточных повторов с использованием информации о семантических связях в языке;
- оптимизация полученного решения;
- проведение экспериментов для оценки эффективности решения.

Алгоритм поиска неточных повторов на основе N-граммной модели

- N-граммная модель
- Построение множества триграмм для предложений
- Критерий принадлежности дубликата к группе — наличие половины триграммов из предложения в группе
- Постепенное разбиение на группы
- Реализован, как часть Duplicate Finder
- Не учитывается семантика

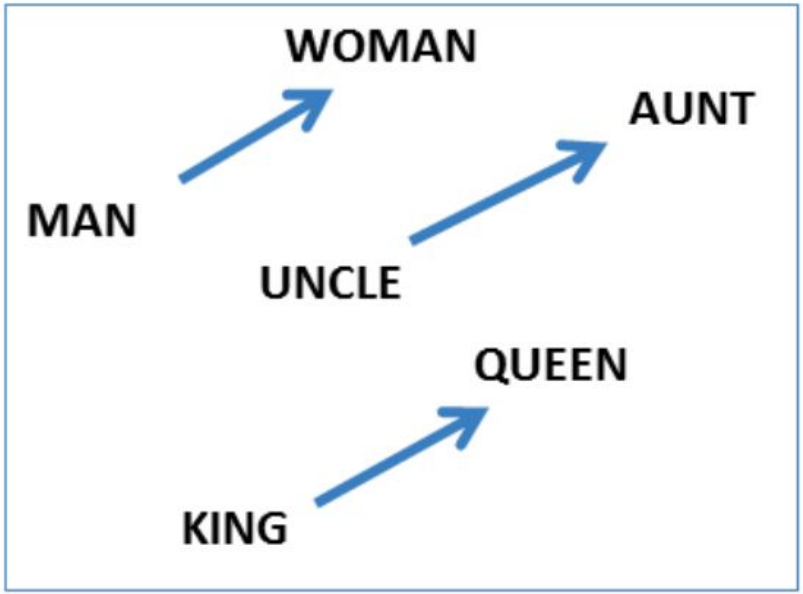
Описание алгоритма

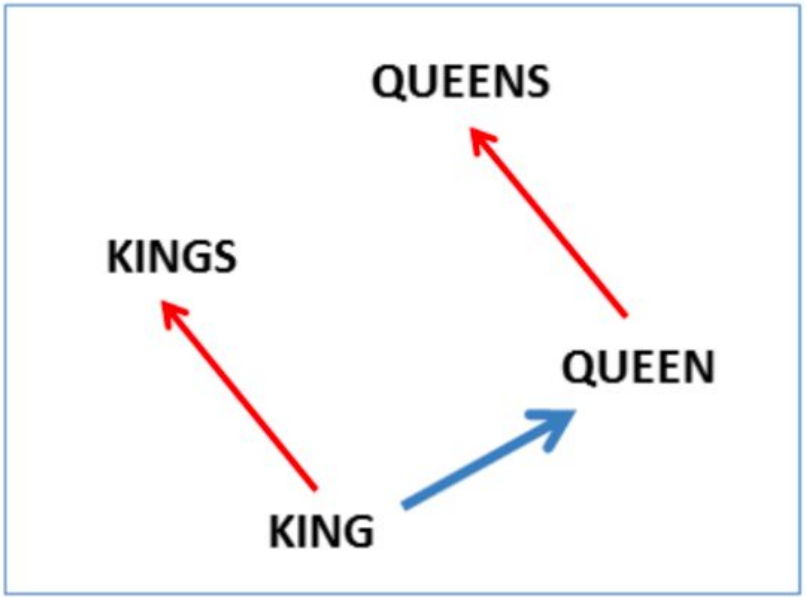
1. Текст разбивается на предложения, по предложениям строятся триграммы
2. Для каждого предложения подыскивается наиболее подходящая группа
   a. Для этого вычисляется пересечение множества триграмм предложения и множества триграмм группы
   b. После этого вычисляется отношение количества элементов полученного пересечения к общему количеству триграмм в группе
   c. Наиболее подходящей группой является группа, в которой значение данного отношения максимально
   d. Если максимальное значение меньше 0.5, создаётся новая группа, которая и становится наиболее подходящей
   e. В наиболее подходящую группу добавляются предложение и его триграммы
3. Группы, содержащие по крайней мере два предложения, и считаются содержащими дубликаты

Векторное представление слов

- Соотнесение каждому слову некоторый вектор в N-мерном пространстве
- Для построения векторного представления часто используются нейронные сети
- Инструмент word2vec
  - Высокая скорость обучения
  - Высокая точность
  - Полученные представления хранят информацию о семантическом различии, что позволяет отвечать на вопросы соотношения между словами
  - Работа с последними возможна при помощи простых операций над векторами
  - Полезен косинусный коэффициент

Сохранение семантических свойств





Рисунки взяты из статьи “**Linguistic Regularities in Continuous Space Word Representations**”

Решение

- Для того, чтобы алгоритм учитывал семантику, было решено использовать векторное представление слов, полученное при использовании word2vec
- Был введен оператор семантического подобия:
  - Пара слов считалась семантически подобной, если косинусный коэффициент был больше некоторого ε
  - Пара триграмм считалась семантически подобной, если пары слов, находящихся на одинаковых позициях были семантически подобными
- Была модифицирована функция intersect — теперь она возвращала множество триграмм предложения, для которых в группе нашлись семантически близкие триграммы

Реализация

- Реализация была разбита на две основные фазы:
  - Создание первоначального решения;
  - Дальнейшая его оптимизация.
- Некоторые решения были обусловлены тем, что работа велась на языке Python.

Наивная реализация

- Быстрое получение векторов для сравнения
  - Необходимость хранить вектора в памяти;
  - Разделение структуры на массив с векторами и словарь, сопоставляющий словам индексы для доступа к их векторному представлению;
  - Расширение структуры, представляющей триграммы, для хранения индексов слов.
- Оператора семантического подобия
  - Неочевидность сравнения специфичных для документации конструкций;
  - При отсутствии векторного представления происходит посимвольное сравнение;
  - ***NumPy*** для более быстрых операций над векторами.

Необходимость оптимизации

- Для оценки эффективности первоначального решения использовалась
  - документация SVN;
  - Векторное представление примерно 71 тысячи слов на 200-мерном пространстве.
- 1.7 миллиардов обращений для расчета косинусного коэффициента
- Из ~7.5 тысяч предложений за 14 часов было обработано лишь 60

Оптимизация

- Формирование матрицы семантики, содержащей семантические отношения между каждой парой слов, имеющих векторное представление;
  - Для хранения используется ***bitarray***
    - Для выбора было произведено сравнение между шестью реализациями
  - Достаточно хранить лишь нижнюю треугольную матрицу, ввиду симметричности оператора семантического подобия
- Вычислить матрицу для заданного векторного представления достаточно лишь один раз

Сравнение различных реализаций bitarray

Структура	Время доступа, сек. 10^6 запросов
bitarray	0.286
bitstring	1.767
intbitset	83.02
Структура на bytes	0.339
Структура на bytearray	0.367
Структура на списке 64-битных целых чисел	0.364

Эксперименты

Эксперименты были основаны на GQM. Были сформулированы следующие вопросы:

1. Как изменилось количество найденных групп, действительно содержащих дубликаты, а также количество ошибок первого рода?
2. Как изменилась производительность алгоритма?

Эксперименты

Для экспериментов были использованы следующие документы:

- DocBook 4 Definitive Guide (DocBook);
- Linux Kernel Documentation (LKD);
- Version Control with Subversion(SVN);
- Zend Framework Documentation (Zend).

Эффективность обнаружения дубликатов

Документ	Оригинальный алгоритм	Предложенное решение	Количество новых групп
DocBook	32	33	1
LKD	74	76	2
SVN	182	186	4
ZEND	452	459	7

Количество групп, в которых содержались дубликаты.

Эффективность обнаружения дубликатов

Документ	Оригинальный алгоритм	Предложенное решение	Количество новых групп
DocBook	24	24	0
LKD	41	43	2
SVN	84	92	8
ZEND	239	246	7

Количество групп с осмысленным текстом, не содержащие дубликаты

Производительность

Документ	Размер, Кб	Время работы алгоритма 1	Время работы алгоритма 2	Отношение времени работы
DocBook	463	1м. 4с.	20м. 33с.	19,2
LKD	595	50с.	14м. 6с.	16,9
SVN	1101	3м. 56с.	69м. 50с.	17,8
Zend	1646	15м. 13с.	312м. 4с.	20,5

Сравнение времени работы оригинального алгоритма и полученного решения.

Заключение

Достигнутые результаты:

- реализован алгоритм поиска неточных повторов с использованием информации о семантических связях в языке;
- полученное решение было оптимизировано;
- были проведены эксперименты для оценки эффективности решения.

Дальнейшее развитие

- Разрешение проблемы ошибок первого рода;
- Дальнейшая оптимизация решения.