Поиск и кластеризация нечетких повторов в документации программного обеспечения

Коновалова Ирина Михайловна

Науч.руководитель: к. ф.-м. н., старший преподаватель
Д.В. Луцив

Рецензент: менеджер проектов
ООО «Системы компьютерного зрения»
Н.А. Пенкрат

Санкт-Петербургский государственный университет
Санкт-Петербург

10 июня 2020 г.

Добрый день! Меня зовут Коновалова Ирина Михайловна. Тема моей выпускной квалификационной работы «Поиск и кластеризация нечетких повторов в документации программного обеспечения.

Документация ПО: Нечеткие повторы

Пример java-doc нечеткого повтора из библиотеки JUnit

Повтор 1

Method public void validatePublicVoidNoArg (FrameworkMethod.java)

Adds to errors this method:

is not public, or takes parameters, or returns something other than void, or is static (given isStatic is false), or is not static (given isStatic is true).

Повтор 2

Method protected void validatePublicVoidNoArgMethods (ParentRunner.java)

Adds to errors if any method in this class is annotated with annotation, but:

is not public, or takes parameters, or returns something other than void, or is static (given isStatic is false), or is not static (given isStatic is true).

Документация играет важную роль при создании программных продуктов. Копирование и лишь частичное изменение фрагментов текста приводит к накоплению ошибок и затрудняет дальнейшее использование документации. Поэтому задача автоматического поиска нечетких повторов для дальнейшего переиспользования и редактирования является актуальной. Результаты работы призваны облегчить разработку и оценку качества новых подходов к обнаружению и анализу дублирующихся данных в документации программного обеспечения, а также обеспечить автоматизацию тестирования создаваемых алгоритмов на тестовых коллекциях документов.

На слайде представлен пример неточного повтора, синим выделены различающиеся фрагменты в тексте. Посомотев на данные фрагменты человек может заключить, что они являются повтором, а значит должны находиться в одной группе. На практике алгоритмы кластеризации используются для автоматизации процесса поиска. Несмотря на то, что люди могут осуществлять поиск достаточно хорошо, этот процесс происходит медленно по сравнению со скоростью компьютера. Ожидается, что алгоритм кластеризации способен так же объеденять фрагменты текста в кластера как это сделал бы человек.

Цель и задачи

Целью работы является исследование возможности применения различных методов обработки текста и алгоритмов кластеризации для задачи поиска неточных повторов в документации программного обеспечения.

- Выбрать методы обработки текста на естественном языке и метрики для сравнения их применения на эталонной коллекции
- Выбрать алгоритмы кластеризации и метрики оценки их качества
- Создать программную среду для апробации создаваемых алгоритмов и разработать методику проведения экспериментов

Целью работы является исследование возможности применения различных методов обработки текста и алгоритмов кластеризации. Для этого были поставлены следующие задачи: выбрать и оценить методы обработки текста, выбрать и оценить алгоритмы кластеризации, создать программную среду для апробации создаваемых алгоритмов и разработать методику проведения экспериментов.

Вопросы исследования

Вопрос исследования 1. (векторное представление)

Какой способ векторного представления позволяет лучше всего находить неточные повторы в Javadoc документации?

Вопрос исследования 2. (оценка качества)

Какие из метрик оценки качества целесообразно применять для сравнения качества алгоритмов кластеризации?

Вопрос исследования 3. (алгоритм кластеризации)

Какой из алгоритмов кластеризации наиболее применим к задаче поиска неточных повторов?

Были поставлены следующие вопросы исследования.

1. Важной метрикой при нахождении дубликатов является семантическое сходство фрагментов текста. Повторы в Javadoc документации представляют собой короткие текстовые фрагменты, поэтому сравнение их с помощью лексического сопоставления может показать хорошие результаты. Но, возможно, это не самый эффективный метод.

2. В случае поиска повторов в документации программного обеспечения не все текстовые фрагменты являются дубликатами, поэтому при кластеризации образуется большое количество одноэлементных кластеров. А так же большинство кластеров, не являющихся одноэлементными, состоят из малого количества элементов. Это вносит существенные ограничения на использование большинства общепринятых внешних метрик оценки качества кластеризации.

3. Несмотря на то, что существует множество методов кластеризации, нет единого мнения о том, какие методы больше подходят для существующего набора данных.

Векторное представление текста

- Bag of Words (BoW)
- Sentence-BERT (SBert)
- Sentence-RoBERTa (SRoBERTa)
- Universal sentence encoder (USE)

Для того, чтобы иметь возможность проанализировать схожесть текста, необходимо представить строки в числовом формате, перевести их в вектора. Одним из подходов к преобразованию текста в числовой вектор может быть представление Bag-of-Words, где каждый термин / слово подсчитывается по частоте появления в тексте. В последние годы широкое распространение получили различные векторные представления, полученные с помощью обучения нейронных сетей на больших корпусах документов. Были рассмотрены векторные представления SBert, SRoberta, USE, которые обучены и оптимизированы для текста, превышающего длину слова (предложения, фразы или короткие абзацы).

Документы эталонной коллекции

	Slf4j	Mockito	GSON	JUnit
Количество символов	97421	290177	174228	207454
Количество слов	15574	40651	25639	29287
Общее количество текстовых фрагментов	225	594	442	223
Количество повторов	133	134	71	223
Количество групп	22	29	18	78

Таблица 1: Набор данных

Для сравнения векторных представлений текста использовался эталонный датасет, состоящий из четырех документов. Разметка документов осуществлялась экспертами предметной области. На слайде представлены характеристики каждого документа. Задачей алгоритма было найти все фрагменты текста, которые похожи между собой. Для каждой пары фрагментов текста в датасете было известно являются они повторами или нет.

Метрики качества для векторных представлений

Косинусное сходство:

$$cos(i,j) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

F- мера:

$$F = 2\frac{Precision \times Recall}{Precision + Recall}$$

Выбор порогового значения:

$$F_{measure} = \underset{h \in [0.5:1]}{\text{argmax}}[F(h)]$$

Для определения схожести двух фрагментов текста использовалось косинусное сходство. Косинусное сходство измеряет косинус угла между векторами. На слайде представлено как вычисляется косинусное сходство для двух векторов x и y, каждый из которых имеет длину n.

Метрикой оценки качества поиска повторов была выбрана F-мера, которая является средним гармоническим между точностью (precision) и полнотой (recall).

Для поиска дубликатов использовался подход с определением пороговой функции. То есть, если значение косинусной метрики между двумя фрагментами текста превышает пороговое значение,то эти фрагменты определяются как повторы. Порог h, выбирался таким образом, чтобы максимизировать значение F-меры. Он равен 0.8

Апробация векторных представлений (1)

Данные - 2638 пар JavaDoc-комментариев, которые были сгенерированы из групп комментариев к документам Slf4J, Mockito, GSON, JUnit

	BoW [1]	USE [2]	S-Bert [3]	S-Roberta [4]
F-мера	0.884 $\pm$0.059	0.875 $\pm$0.064	0.861 $\pm$0.069	0.881 $\pm$0.064

Таблица 2: Среднее значение F-меры, доверительный интервал (CI = 95)

[1] Bag of Words
[2] Universal sentence encoder
[3] Sentence-BERT
[4] Sentence-RoBERTa

Данные это 2638 пар JavaDoc-комментариев, сгенерированных из групп комментариев к представленным раннее документам Slf4J, Mockito, GSON, JUnit. В таблице приведены результаты апробации различных векторных представлений. Как видно из таблицы, значение F-меры для векторного представления BoW является высоким и превосходит значения F-меры для других векторных представлений.

Апробация векторных представлений (2)

	Slf4j (133 повтора)	Mockito (134 повтора)	GSON (71 повтор)	JUnit (223 повтора)
BoW [5]	0.892	0.880	0.901	0.881
USE [6]	0.887	0.878	0.895	0.877
S-Bert [7]	0.836	0.842	0.885	0.871
S-Roberta [8]	0.885	0.878	0.897	0.879

Таблица 3: Результаты апробации для документов: F-мера

На данном слайде представлена апробация векторных представлений для каждого из документов в отдельности. Эти данные показывают насколько хорошо векторные представления позволяют искать повторы в имеющимся эталонном датасете. Вероятная причина того, что Bow показывает лучший результат по сравнению с другими моделями, заключается в том, что повторы в Javadoc имеют небольшой объем и насыщены специальной терминологией, в связи с этим модели, обученные на других корпусах, могут не содержать в словаре необходимых слов. Тем не менее, применение предобученных векторных представлений может быть использовано в дальнейших работах при рассмотрении других видов документаций. Рассмотрим на примерах почему это может показать хороший результат по сравнению с BoW.

[5] Bag of Words
[6] Universal sentence encoder
[7] Sentence-BERT
[8] Sentence-RoBERTa

Пример из DITA-документации

Adding work packages to an iteration.Iterations can group work packages with the same due dates. At least one iteration must exist. Group all work packages due for the same milestone into one iteration. In My Products, expand the desired product. Click the desired release. Click Iterations. Select the box around the iteration to which the work package belongs. Work packages that are not yet added to an iteration are shown in Unscheduled. All work packages in the selected iteration appear. Drag and drop the work package into the desired iteration. The work package is now part of the selected iteration.

Adding an iteration. Iterations can group your work packages by due dates. Know the due date for your iteration. Use iterations to easily track a group of work packages with similar due dates. In My Products, expand the desired product. Click the desired release. Click Iterations. Click Create new iteration. Enter a name and description for the iteration. Set the start and end dates. Click Save. The iteration is set and ready for work packages. If you need to edit an iteration, click the iteration name. The iteration details window opens for editing.

	BoW	USE	S-Bert	S-Roberta
cosine measure	0.771	0.711	0.901	0.881

Таблица 4: Результаты апробации на DITA-документации

На слайде представлены два фрагмента текста, а также таблица, в которой указано значение косинусной метрики между этими фрагментами при использовании различных векторных представлений. Как видно из данного примера, для DITA-документации предобученные модели S-Bert и S-Roberta показываютрезультат лучше, чем BoW. Это связано с тем, что количество общих словв двух фрагментах текста невелико. При этом, даже не смотря на наличие технических терминов, эти модели смогли распознать семантическое сходство фрагментов.

Пример из Javadoc документации

"Adds to errors if this method: is not public, or takes parameters, or returns something other than void, or is static (given isStatic is false), or is not static (given isStatic is true)"

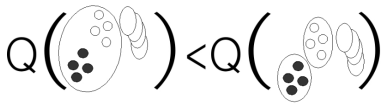
"Adds to errors if any method in this class is annotated with annotation, but: is not public, or takes parameters, or returns something other than void, or is static (given isStatic is false), or is not static (given isStatic is true)"

	BoW	USE	S-Bert	S-Roberta
cosine measure	0.957	0.896	0.915	0.910

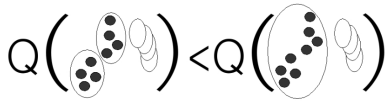
Таблица 5: Результаты апробации на Javadoc-документации

На слайде представлен пример неточного повтора из Javadoc документации. Большинство повторов в рассматриваемой Javadoc документации имеют подобный вид. Из-за того, что лишь малая часть повтора является вариативной, BoW хорошо определяет неточный повтор. Основываясь на полученных результатах, для кластеризации неточных повторов в Javadoc документации было выбрано векторное представление текста в виде BoW.

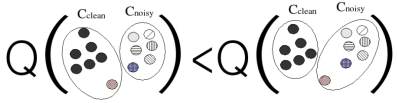
Оценка качества кластеризации: свойства метрик



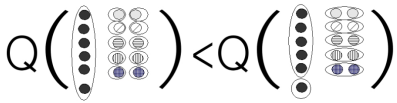
1) Однородность



2) Полнота



3) Rag Bag



4) Size vs Quantity

Для оценки кластеризации необходимо выбрать метрику оценки качества. На слайде представлены свойства, которым должна удовлетворять эта метрика.

Первое свойство означает, что значение метрики качества должно уменьшаться при объединении в один кластер двух эталонных.

Второе свойство означает, что значение метрики качества должно уменьшаться при разделении эталонного кластера на части.

Третье свойство показывает, что поместить новый нерелевантный обоим кластерам элемент в разупорядоченный кластер менее вредно, чем поместить этот элемент в чистый кластер.

В задаче кластеризации повторов в документации программного обеспечения большинство кластеров содержат малое количество элементов, таким образом, особенно важным является выполнение свойства Size vs Quantity. То есть значительное ухудшение кластеризации большого числа небольших кластеров должно обходиться дороже небольшого ухудшения кластеризации в крупном кластере.

Метрики качества кластеризации

Внешняя метрика

- BCubed F-measure

Внутренняя метрика

- Коэффициент силуэта

Существуют внутренние и внешние метрики для оценки качества работы алгоритмов кластеризации. Внутренние позволяют сравнивать результаты кластеризации в отсутствие дополнительной информации об истинных классах объектов, т.е. исходя только из имеющихся данных об объектах выборки и построенной кластеризации. Внешние оценки качества опираются на информацию об истинных классах объектов, т.е. на эталонную коллекцию.

В качестве внешней метрики была выбрана BCubed F-мера, которая удовлетворяет всем четырем свойствам метрик. Группа метрик BCubed предлагает взглянуть на кластеризацию с точки зрения одного отдельного элемента. Точность элемента показывает, сколько элементов в одном алгоритмическом кластере находится так же и в одном эталонном. Полнота элемента показывает сколько элементов из такого же эталонного кластера находятся в алгоритмическом кластере. F-мера является их средним гармоническим.

В качестве внешней метрики был выбран коэффициент силуэта. Эта метрика показывает, насколько среднее расстояние до объектов своего кластера отличается от среднего расстояния до объектов других кластеров.

Кластеризация

- K-means++
- Спектральная кластеризация
- Affinity Propagation
- Агломеративная кластеризация с количеством кластеров
- Агломеративная кластеризация с пороговым значением
- DBSCAN

В работе были рассмотрены алгоритмы кластеризации, представленные на слайде. Они основаны на различных подходах. K-means является алгоритмом, основанным на разбиении, спектральный алоритм использует матрицу подобия, affinity propagation основан на идее "передачи сообщений"между объектами выборки, агломеративная кластеризация является иерархическим алгоритмом, dbscan это алгоритмом на основе плотности. Приведенные алгоритмы являются одними из самых часто используемых в различных исследованиях.

Результаты кластеризации (1)

	kmeans++	Спектральная	Агломеративная
FBCubed	0.88 ±0.04	0.86 ±0.03	0.91 ±0.03

Таблица 6: Максимальное значение FBCubed

	kmeans++	Спектральная	Агломеративная (k)
FBCubed	0.86 ±0.03	0.85 ±0.04	0.88 ±0.03

Таблица 7: FBCubed при подборе параметров с помощью коэффициента силуэта

Апробация алгоритмов кластеризации проводилась на четырех документах эталонной коллекции (SLF, Mockito, GSON, JUnit). Каждому фрагменту текста экспертами был присвоен номер группы, к которой относится данный фрагмент.

Для алгоритмов кластеризации, которые требуют указания количествакластеров перед началом своей работы (k-means, спектральная, агломеративная), была реализована возможность оценки экспертом результатов работы алгоритма при различных параметрах, с последующим выбором оптимального значения. Для осуществления этого процесса строится график, показывающий значение коэффициента силуэта при разном заданном количестве кластеров.

В верхней таблице приведены усредненные максимальные значения метрики качества FBCubed для имеющегося набора данных. Эти данные были полученыпутем перебора параметра k(количества кластеров), а также подбором оптимальных параметров для каждого алгоритма. Данные значения можно считать потенциальным максимумом для данных алгоритмов.

В таблице, расположенной внизу слайда, приведены усредненные значения метрики FBCubed, полученные при подборе параметров с помощью оптимизации коэффициента силуэта.

Результаты кластеризации (2)

	Affinity	Агломеративная (порог)	DBSCAN
FBCubed	0.91 $\pm$0.05	0.92 $\pm$0.04	0.90 $\pm$0.04
Силуэт	0.31 $\pm$0.05	0.37 $\pm$0.06	0.37 $\pm$0.05

Таблица 8: FBCubed для алгоритмов, не требующих задания количества кластеров

Алгоритмы Affinity Propagation, агломеративный с указанием порогового значения и DBSCAN определяют количество кластеров автоматически. В таблице представлено среднее максимальное значение метрики FBCubed и среднее значение коэффициента силуэта при кластеризации.

Приведенное сравнение демонстрирует превосходство алгоритмов, которые при запуске не требуют указания количества кластеров. При этом удобство их использования так же значительно выше, так как нет необходимости несколько раз запускать алгоритмы для поиска лучшего разбиения.

Исследование полученных разбиений повторов показало, что большинство из повторов, которые были неправильно кластеризованы алгоритмами являются неоднозначными и их сложно отнести к какому-то одному классу. В связи с этим, несмотря на то, что полученные разбиения удовлетворяют внешней метрике качества, в последующих работах предполагается рассмотреть алгоритмы нечеткой кластеризации, в которых одному объекту может быть назначено несколько кластеров.

Время работы алгоритмов

	Apach	SWT
Количество символов	1520997	342807
Количество слов	198035	53030
Общее количество текстовых фрагментов	3648	1454

Таблица 9: Документы для тестирования времени работы алгоритмов

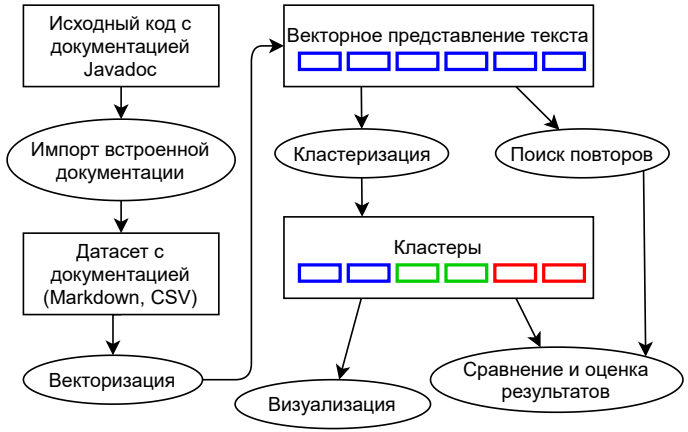
	Apach	SWT
Affinity	905.31 ±20.14	7.12 ±.93
Агломеративная	30.27 ±1.47	3.72 ±0.29
DBSCAN	4.12 ±0.13	0.44 ±0.06

Таблица 10: Время работы (в секундах)

Для алгоритмов, показавших наибольшее значение метрики FBCubed (это алгоритмы Afinity Propagation, агломеративная кластеризация по пороговому значению и DBSCAN) была проведена оценка времени их работы на документах проекта Apache Commons Collections и проекта Eclipse SWT. Данные документы не являются размеченными, они были выбраны, так как по объему превышают имеющиеся в эталонной коллекции. На слайде представлено описание документов и время работы (в секундах) алгоритмов кластеризации.

Для автоматической кластеризации повторов в рамках текущей работы лучшими алгоритмами являются алгоритм агломеративной кластеризации по пороговому значению, который показал лучшее значение FBCubed метрики на эталонной коллекции, и алгоритм DBSCAN, время работы которого существенно ниже, чем у других алгоритмов, что является актуальным при обработке документаций крупных проектов. В ходе экспериментов было установлено, что в качестве параметров для агломеративного алгоритма косинусное сходство между двумя векторами должно быть больше 0.8 (косинусное расстояниев таком случае будет 0.2), а в качестве метода вычисления расстояния между кластерами должен быть использован метод попарного среднего.

Программная среда



Для реализации и проведения экспериментов был выбран язык Python, популярный при решении задач в областях data science и анализа текстов на естественных языках. Разработка модели эксперимента проводилась в Jupyter Notebook. Формат файлов-блокнотов .ipynb позволяет хранить в одном файле программный код, комментарии, графики и изображения, что заметно облегчает анализ полученных результатов. При этом так же доступна возможность обращаться к коду во внешних модулях Python.

Исходный Javadoc файл преобразуется в формат .csv, где каждый текстовый фрагмент представлен отдельной строкой. После этого файл обрабатывается в одно из векторных представлений (BoW, USE, S-Bert, S-Roberta) и осуществляется поиск повторов по косинусному сходству, либо кластеризацияих на группы, с последующей оценкой результатов работы алгоритмов. Визуализация осуществляется с использованием инструмента DuplicateFinder, что помогает наглядно оценить разбиения, полученные при кластеризации и проанализировать возможные причины возникновения ошибок в работе алгоритмов. Данную программную среду так же можно использоватьдля поиска неточных повторов в неразмеченных данных.

Результаты

- Произведена оценка различных методов векторного представления текста на естественном языке применительно к задаче поиска повторов в документации ПО
- Произведено сравнение алгоритмов кластеризации на тестовой коллекции документов
- Создана программная среда[9] и разработана методика проведения эксперимента, позволяющая проводить апробацию создаваемых алгоритмов

[9] https://github.com/ikonovalova/clustering-algorithms

В работе были достигнуты слудующие результаты: проведена оценка различных верторных представлений, произведено сравнение алгоритмов кластеризации, разработана методика проведения экспериментов и создана программная среда, позволяющая проводить апробацию создаваемых алгоритмов.

Замечание рецензента: В качестве недостатка можно выделить низкое внимание, уделенное реализации среды для апробации алгоритмов, несмотря на то, что эта задача была сформулирована в целях работы. Ответ: Согласна с замечанием рецензента, стоило рассказать подробнее