УДК 004.421.2:512.54.05 DOI: 10.18698/0236-3933-2022-4-20-35

ВЫЯВЛЕНИЕ И КЛАСТЕРИЗАЦИЯ ШАБЛОННЫХ ТЕКСТОВ В БОЛЬШИХ МАССИВАХ СООБЩЕНИЙ

И.Э. Вишняков vishnyakov@bmstu.ru
И.П. Иванов ivanov@bmstu.ru
И.А. Каркин karkinia@yandex.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Многие сервисы используют короткие сообщения для различных целей, например, магазины рассылают акционные предложения, МЧС России информирует население при угрозе возникновения чрезвычайных ситуаций природного и техногенного характера. Выделение из общего трафика коротких текстов шаблонных сообщений можно использовать для фильтрации спама и рассылок, чтобы уберечь пользователей от мошеннических действий. Зачастую такие массивы сообщений достигают настолько больших размеров, что их хранение и обработка на одном выделенном персональном компьютере или сервере попросту невозможны. Разработаны методы эффективного выявления и кластеризации шаблонных текстов из больших массивов коротких сообщений с применением фреймворка для реализации распределенной обработки неструктурированных данных. Рассмотрены методы, позволяющие проводить кластеризацию на больших массивах сообщений с применением распределенных вычислений без предварительного получения векторных представлений текстов. Приведены алгоритмы для эффективного выявления шаблонных сообщений из больших массивов коротких текстов. Выполнено сравнение алгоритмов по производительности и качеству выявления шаблонов

Ключевые слова

Выявление шаблонов, кластеризация текстов, большие данные

Поступила 16.06.2022
Принята 27.06.2022

Введение. Задача выявления групп шаблонных сообщений относится к задачам кластеризации текстовой информации. Существует большое число методов решения данной задачи, однако часть из них не адаптирована для применения на кластере серверов, а другая часть не применима

на практике из-за больших затрат ресурсов, обусловленных объемом массива коротких сообщений.

Цель настоящей работы — разработка методов эффективного выявления и кластеризации шаблонных текстов из больших массивов коротких сообщений с применением фреймворка для реализации распределенной обработки неструктурированных данных Apache Spark. Общий обзор методов кластеризации текстовой информации приведен в работе [1].

В силу специфики, связанной с большим объемом данных и использованием кластера Spark [2] для хранения и обработки этих данных, при решении поставленной задачи необходимо рассматривать те методы, которые допускают распределенную реализацию и имеют высокую скорость работы.

Все методы кластеризации текстов можно разделить на числовые (или метрические) и нечисловые. Числовые методы в основном работают с векторами признаков, полученными с помощью нейронных сетей глубокого обучения таких, как Word2Vec [3]. Нечисловые методы работают непосредственно со словами. Зачастую нечисловые методы требуют адаптации под особенности конкретной задачи кластеризации текстов. К нечисловым методам можно отнести методы на основе деревьев и алгоритмы выравнивания последовательностей, которые применяются в биоинформатике [4].

При использовании числовых методов кластеризации каждый текст представляется в виде точки в пространстве $\mathbb{R}^n$ с помощью отображения $\varphi: T \to \mathbb{R}^n$, которое каждому объекту данных $t \in T$ сопоставляет его векторное представление $\varphi(t) \in \mathbb{R}^n$. Задача получения такого отображения обычно рассматривается отдельно.

Числовые методы кластеризации могут быть разделены на методы разбиения, иерархические, плотностные, сетевые и модельные. В настоящей работе особый интерес представляют алгоритм *k*-средних [5], относящийся к методам разбиения, и алгоритм DBSCAN (Density-Based Spatial Clustering of Applications with Noise — основанная на плотности пространственная кластеризация для приложений с шумами) [6], относящийся к плотностным методам.

В основе алгоритма *k*-средних лежит идея разбиения всего пространства на кластеры так, чтобы расстояние между объектами внутри одного кластера было минимально, а между объектами различных кластеров — максимально. Объект относится к тому кластеру, расстояние до центроида c_i которого минимально. Центроид кластера вычисляется по формуле

$$c_i = \frac{\sum\limits_{k \in K_i} \varphi(k)}{\left|K_i\right|} \in K_i, \qquad (1)$$

где K_i — множество текстов, принадлежащих одному кластеру.

В плотностных методах кластеры рассматриваются как регионы пространства с высокой плотностью объектов, а разделяющие их регионы являются пространством с низкой плотностью объектов. В алгоритме DBSCAN локальная плотность объекта определяется как число объектов из его эпсилон-окрестности $U_\varepsilon(t_0) = \left\{t \in T \mid d(\varphi(t), \varphi(t_0)) < \varepsilon\right\}$.

Алгоритмы нечисловой кластеризации используют суффиксные деревья или *FP*-деревья (*Frequent-Pattern Tree* — дерево частых шаблонов) [7]. Один из алгоритмов построения суффиксного дерева описан в [8]. Однако эти методы требуют построения дерева на одном узле кластера, что снижает скорость работы алгоритма. Тем не менее алгоритм кластеризации с применением *FP*-деревьев можно модифицировать и добиться значительного прироста производительности благодаря распределенным вычислениям.

Под шаблоном понимается непустая строка $a_0 b_0 a_1 b_1, ..., a_{n-1} b_{n-1} a_n$ символов из алфавита *S*, где $\forall i : a_i,\ b_i \in S^*$. При этом кортеж $a_0, a_1, ..., a_n$ называют постоянной частью шаблона, а кортеж $b_0, b_1, ..., b_{n-1}$ — переменной частью шаблона.

Дерево *FP* представляет собой дерево, в корне которого содержится пустая вершина. Потомками пустой вершины являются статистически самые часто встречаемые шаблоны среди всех шаблонов из представленной выборки текстов. Далее построение дерева продолжается рекурсивно с тем условием, что если вершина x_i непосредственно доминирует над вершиной x_j, то шаблон, сопоставленный вершине x_j, не может встречаться в текстах, в которых нет шаблона, сопоставленного вершине x_i. Построение ветви дерева прекращается на вершине x_i, если вершине x_j не может быть сопоставлен никакой шаблон. Обзор применения *FP*-деревьев в задаче выявления шаблонов приведен в [9]. Модификация алгоритма построения *FP*-дерева для выявления шаблонов в массиве коротких сообщений приведена в [10].

Для решения задачи кластеризации текстов также можно использовать методы выравнивания последовательностей, которые активно применяются в биоинформатике для работы с белковыми и нуклеотидными последовательностями [4, 7].

Все методы выравнивания последовательностей базируются на общей идее: поиск расстояния Левенштейна между двумя последовательностями. Расстояние Левенштейна определяется как минимальное число операций вставки, удаления и замены символов, необходимых для преобразования первой последовательности во вторую [11]. Наиболее часто в алгоритмах биоинформатики применяется модифицированная метрика Левенштейна, у которой цена операции зависит от самого вида операции и от участвующих в ней символов [7]. Одним из классических решений поставленной задачи является алгоритм Смита — Уотермана [4].

Для поиска шаблонных сообщений основным недостатком расстояния Левенштейна является то, что расстояние между произвольными короткими текстами оказывается небольшим, при этом расстояние между похожими длинными текстами может оказаться достаточно большим. Это означает, что короткие тексты будут чаще образовывать кластеры с другими произвольными короткими текстами, чем длинные тексты с длинными похожего содержания. Модификация алгоритмов получения редакционного предписания для поиска непрерывных участков совпадающих последовательностей значительной длины приведена в [12]. Это помогает уменьшить ранее упомянутый недостаток расстояния Левенштейна при обработке текстов коротких сообщений.

В результате предлагается использовать два числовых алгоритма кластеризации: *k*-средних и DBSCAN, в качестве исходных объектов будут выступать не числовые векторы, а строки исходных текстов с применением идеи выравнивания последовательностей. Это должно обеспечить значительное увеличение скорости кластеризации за счет пропуска шага получения векторных представлений текстов. Предлагается также использовать нечисловой метод на основе *FP*-деревьев.

Модификация алгоритма DBSCAN. Алгоритм DBSCAN предполагается использовать на метрическом пространстве строк S^* с введенной на нем нормированной метрикой Левенштейна:

$$d(x,\ y) = \frac{2GLD(x,\ y)}{\|x\| + \|y\| + GLD(x,\ y)}, \tag{2}$$

где $GLD(x, y)$ — расстояние Левенштейна; $\|a\|$ — длина слова a. Нормированное расстояние Левенштейна рассмотрено в [13], авторы которой доказывают, что это расстояние на самом деле является метрикой. Нормированная метрика Левенштейна поможет избежать случая, когда короткие тексты будут постоянно образовывать кластеры с другими произвольными

короткими текстами, а длинные тексты похожего содержания практически не будут образовывать кластеры.

Пропуская операцию векторизации, можно определить эпсилон-окрестность текста:

$$U_\varepsilon(t_0) = \left\{ t \in T \,\middle|\, d\,(t, t_0) < \varepsilon \right\}.$$

Для описания схемы модифицированного алгоритма кластеризации DBSCAN необходимо ввести еще несколько определений. Объект $t_0 \in T$ называется корневым, если $\left|U_\varepsilon(t_0)\right| \geq m$ ($m \in \mathbb{N}$ и $\varepsilon \in \mathbb{R}^+$ являются параметрами алгоритма). Объект $p \in T$ непосредственно плотно достижим из объекта, если $p \in U_\varepsilon(t_0)$ и t_0 является корневым. Отметим, что любой корневой объект непосредственно плотно достижим из самого себя. Объект $p \in T$ плотно достижим из объекта $t_0 \in T$, если $\exists p_0, p_1, ..., p_n \in T$ такие, что $p_0 = t_0$, $p_n = p$ и $\forall i \in \left[0, n\right] p_{i+1}$ непосредственно плотно достижим из p_i. Кластеры $K_1, K_2 \subset T$ плотно достижимы, если все корневые объекты K_1 плотно достижимы из корневых объектов K_2 и все корневые объекты K_2 плотно достижимы из корневых объектов K_1. Объект $p \in T$ (непосредственно) плотно достижим из кластера, если p (непосредственно) плотно достижим из некоторого корневого объекта кластера.

Схема модифицированного алгоритма кластеризации DBSCAN следующая.

1. Для всех корневых сообщений $r \in T$ создается новый кластер K_r, состоящий из всех непосредственно плотно достижимых из r сообщений, где расстояние считается по нормированной метрике Левенштейна (2).

2. Пока не произойдет стабилизация кластеров (т. е. значения кластеров на текущем и предыдущем шаге совпадут), повторяются следующие пункты:

– если существуют плотно достижимые по нормированной метрике Левенштейна кластеры, то их необходимо объединить в один;

– для всех кластеров K_i добавить в K_i все непосредственно плотно достижимые по нормированной метрике Левенштейна из K_i объекты.

Модификация алгоритма *k*-средних. Для корректной работы алгоритма *k*-средних на пространстве строк с нормированной метрикой Левенштейна помимо метрики необходимо определять центры кластеров. Центрированием строк $s_1, s_2, ..., s_k \in S^*$ назовем симметричную n-арную операцию

$$c(s_1, s_2, ..., s_k) = d_E(s_{i_1}, d_E(s_{i_2}, ..., d_E(s_{i_{k-1}}, s_{i_k}) ...)), \qquad (3)$$

где $d_E(x, y)$ — общая часть редакционного предписания строк x и y; строки $s_1, s_2, ..., s_k$ в правой части выражения расположены в лексикографическом порядке: $s_{i_1} \le s_{i_2} \le ... \le s_{i_k}$.

Схема модифицированного алгоритма кластеризации k-средних имеет следующий вид.

1. Выбираются случайные $1 \le i_1 < i_2 < ... < i_k \le |T|$.

2. В качестве центроидов $c_1, c_2, ..., c_k$ кластеров выбираются короткие сообщения $t_{i_1}, t_{i_2}, ..., t_{i_k}$.

3. Пока не произойдет стабилизация кластеров (т. е. значения кластеров на текущем и предыдущем шагах совпадут) повторяются следующие пункты:

– для всех $t \in T$ отнести t к кластеру K_i, где $i = \arg\min\limits_{j \in [1, k]}(d(t, c_j))$, $d(t, c_j)$ — нормированное расстояние Левенштейна между сообщением t и текстом c_j, вычисляемое по формуле (2);

– пересчитать значения центроидов c_i всех кластеров в соответствии с формулой центрирования (3) для строк $t_1, t_2, ..., t_{|K_i|} \in K_i$ по формуле $c_i = c(t_1, t_2, ..., t_{|K_i|})$.

Кроме того, что в алгоритме k-средних используется нормированное расстояние Левенштейна (2), его отличие от классического заключается в том, что для подсчета центроидов кластеров используется формула (3), а не (1). Это позволит избавиться от необходимости проведения промежуточного получения векторных представлений текстов.

Отметим, что редакционное предписание не всегда единственно, однако d_E не зависит от выбора конкретного редакционного предписания. Поскольку расчет d_E происходит за $O(n^2)$, а реализация алгоритма k-средних предполагает многократное вычисление d_E от одних и тех же параметров при пересчете центроидов после добавления новых строк в кластер, то при реализации алгоритма имеет смысл хэшировать результаты вычисления d_E.

Модификация алгоритма кластеризации с применением *FP*-дерева. Для кластеризации с применением *FP*-деревьев предварительно необходимо провести нормализацию слов для того, чтобы при подсчете частоты употребления слова учитывать различные его написания. Необходимо использовать распределенный алгоритм нормализации слов. Поскольку алгоритмы кластеризации с применением *FP*-деревьев основаны не на использовании метрик, а на подсчете статистики слов, то в данном случае

в качестве способа нормализации можно использовать хеширование по сигнатуре [14]. Для каждого слова задается его класс эквивалентности посредством введения хеш-функции, которая составлена таким образом, что получившиеся в результате классы устойчивы относительно произвольного числа перестановок и повторений букв для слов, являющихся представителями этих классов. В дальнейшем можно считать, что все слова, относящиеся к одному классу, являются различными написаниями одного слова. Однако хеширование по сигнатуре имеет один серьезный недостаток — в ряде случаев совершенно разные по смыслу слова могут быть отнесены к одному классу. При эффективной реализации сбор статистики по массиву сообщений может иметь временную сложность $O(L\cdot|T|)$, где $|T|$ — число сообщений в массиве данных; L — среднее значение длины одного сообщения. В данном случае число сообщений значительно больше их длины, поэтому при оценке временной сложности алгоритма средней длиной сообщения можно пренебречь.

Как правило, все короткие сообщения, содержащие общий шаблон, имеют общего отправителя или получателя. Другие сообщения с общим шаблоном не представляют интереса, поскольку не являются спамом. Это позволяет применить еще одну оптимизацию, суть которой заключается в перегруппировке сообщений по номеру отправителя или получателя с последующим распределенным сбором статистики о словах (внутри каждой такой группы статистика собирается на одном узле кластера). Во-первых, значительно упрощается распределенная реализация алгоритма благодаря изолированию групп сообщений и отсутствию необходимости использовать распределенную память внутри таких групп. Во-вторых, подсчет статистики внутри таких групп выполняется локально на одном узле кластера, что увеличивает производительность вычислений, а также позволяет добиться линейной зависимости не от общего числа сообщений в массиве данных, а от числа сообщений в самой большой группе. Однако данный метод стоит использовать лишь в тех случаях, когда группировка сообщений по отправителю или получателю оправдана (т. е. кластеры, в сообщениях которых все отправители и получатели различны, не представляют интереса).

После нормализации необходимо выполнить частичное построение *FP*-дерева на одном узле кластера по N шаблонам, наиболее часто употребляемым в выборке (в данном случае в качестве шаблонов выступают нормализованные слова или их хеши). Когда очередное наиболее часто употребляемое слово либо имеет частоту ν меньше наперед заданного значения, либо встречается во всех текстах суммарно менее m раз, то по-

строение той ветви *FP*-дерева, к которой относится данное слово, необходимо остановить. Построение дерева завершается при добавлении в него N наиболее часто употребляемых слов. Такой метод с частичным построением дерева снижает нагрузку на том узле кластера, где выполняется построение дерева, и одновременно с этим увеличивает максимальный размер входных данных.

Добавление одного слова в дерево осуществляется за $O(\log P)$, где P — текущее число вершин в дереве. Поскольку максимальное число вершин в дереве не может превышать N, то оценка худшего времени построения дерева равна $O(N \log N)$, где N — параметр модели.

После построения *FP*-дерева необходимо провести кластеризацию. Кластером K_i будем считать множество всех текстов, в которых встречается слово, соответствующее *i*-й вершине кроны построенного *FP*-дерева.

Оценка качества кластеризации разработанных алгоритмов. Метрическая кластеризация на основе строковых представлений текстов проводится с применением распределенных реализаций алгоритмов *k*-средних [15] и DBSCAN [16] на метрическом пространстве строк с введенной на нем нормированной метрикой Левенштейна. Кластеризация с применением *FP*-деревьев проводится посредством применения распределенной реализации алгоритма построения *FP*-дерева.

Проблема оценки качества кластеризации заключается в необходимости заранее знать число кластеров и то, какое сообщение, к какому кластеру должно относиться. Поэтому при оценке качества кластеризации разработанных методов будет оцениваться не качество разделения на кластеры уже выделенных шаблонов, а качество выделения шаблонов в сообщениях как таковых. Это позволяет объединить все шаблонные сообщения в один кластер и при оценке качества рассматривать лишь принадлежность сообщения к этому кластеру.

В данном случае для оценки качества работы алгоритма используются метрики точности и полноты, вычисляемые по формулам:

$$P = \frac{T_P}{T_P + F_P}; \; R = \frac{T_P}{T_P + F_N},$$

где T_P — число шаблонных сообщений, в которых модель распознала шаблон; F_P, F_N — ошибки первого и второго рода.

Точность и полнота не зависят от баланса классов, что также делает их устойчивыми при несбалансированных выборках. Для одновременного учета в одной метрике точности и полноты рассматривалась F_β-мера:

$$F_\beta = \frac{(1 + \beta^2)PR}{\beta^2 P + R},$$

где β определяет значимость точности перед полнотой.

Для оценки качества кластеризации использовался размеченный вручную массив из 1000 сообщений. Каждое сообщение отнесено к одному из двух классов. Сообщения первого класса содержат шаблон (не обязательно общий) и должны содержаться в одном кластере. Сообщения второго класса не содержат шаблонов и не должны содержаться ни в одном кластере.

Оценка качества результатов кластеризации проводилась для трех модификаций алгоритмов:

1) метрическая кластеризация на основе строковых представлений текстов алгоритмом *k*-средних;

2) метрическая кластеризация на основе строковых представлений текстов алгоритмом DBSCAN;

3) кластеризация с применением *FP*-деревьев.

Для сравнения со стандартным методом проводилась метрическая кластеризация алгоритмом DBSCAN [17] на основе векторных представлений текстов. В качестве векторизатора использовалась универсальная предварительно обученная модель Word2Vec [3], которая в дальнейшем дообучалась на тех же текстовых данных, на которых необходимо провести кластеризацию. В качестве данных для предварительного обучения во всех случаях предполагалось использовать очищенные тексты русских страниц из [18] и набор коротких текстов на русском языке из [19]. Использование предварительно обученных моделей позволяет получать значительный прирост производительности во время кластеризации, поскольку большая часть работы проведена заранее. При оценке производительности учитывалось только время дообучения модели Word2Vec на текстовых данных, на которых необходимо провести кластеризацию. Результаты тестирования приведены в табл. 1.

Согласно данным из табл. 1, кластеризация с применением *FP*-деревьев и метрическая кластеризация на основе строковых представлений текстов позволяют добиться повышения точности путем потери в полноте. Из пятого столбца (F_1-мера) табл. 1 следует, что кластеризация с промежуточной векторизацией может показать лучшие результаты в задачах с одинаковой значимостью точности и полноты. Методы на основе *FP*-дерева и алгоритма DBSCAN на метрическом пространстве строк с введенной на нем нормированной метрикой Левенштейна показали высокие значения точности. Из четвертого столбца ($F_{0,5}$-меры) следует, что такие методы могут

иметь преимущество по сравнению с методами промежуточной векторизации в задачах с повышенной значимостью точности перед полнотой. Так, при решении задачи обнаружения спама и рассылок точность имеет большее значение, поскольку пропустить рассылку лучше, чем отнести важное письмо к спаму. Однако при решении задачи обнаружения мошенничества значимость полноты выше, поскольку куда важнее обнаружить как можно больше мошеннических текстов, чем бояться выносить ложные обвинения.

Таблица 1

Оценка качества кластеризации алгоритмов

Метод	Точность	Полнота	$F_{0,5}$-мера	F_1-мера	F_2-мера
Модифицированный алгоритм с построением *FP*-дерева	0,99	0,47	0,81	0,64	0,53
Модифицированный алгоритм DBSCAN	0,87	0,59	0,79	0,70	0,63
Модифицированный алгоритм *k*-средних	0,80	0,65	0,76	0,72	0,68
DBSCAN + Word2Vec	0,72	0,88	0,75	0,79	0,84

Метод на основе *k*-средних на метрическом пространстве строк с введенной на нем нормированной метрикой Левенштейна не имеет большого преимущества перед кластеризацией с промежуточной векторизацией в задачах с повышенной значимостью точности, но при этом сильно уступает по показателю F_2-меры. При этом методы на основе строковых представлений тестов имеют соизмеримые результаты F_1-меры, но метод на основе алгоритма *k*-средних является более сбалансированным в плане точности и полноты. Это позволяет рассматривать метод на основе алгоритма *k*-средних в качестве альтернативы кластеризации с промежуточной векторизацией лишь в случаях задач с одинаковой значимостью точности и полноты для увеличения производительности.

Исследование производительности разработанных алгоритмов. При исследовании производительности использовался кластер Spark с 12 узлами со следующими характеристиками каждого узла:

– Ubuntu 16.04.7 LTS (ОС);
– Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz (ЦП);
– 128GB DDR3, 1866 MHz (ОЗУ);
– 20TB HDD (ПЗУ).

Исследование проводилось на реальных исторических данных. Полный размер выборки включает в себя 27,6 млн коротких сообщений, хранящихся в HDFS (Hadoop Distributed File System — распределенная файловая система Hadoop) тестового кластера [20]. Средняя длина одного сообщения составляет 14,5 слов, медиана — 11 слов. В процессе тестирования последовательно запущено несколько задач Spark для различных объемов тестовой выборки. Каждая задача Spark выполняет кластеризацию заданного набора сообщений с применением одного разработанного метода кластеризации. Результаты тестирования приведены в табл. 2. Все разработанные методы позволяют добиться значительного повышения производительности благодаря пропуску этапа обучения модели векторизации слов.

Таблица 2

Исследование производительности (времени, с) алгоритмов

Метод	Число сообщений, 10^3			
	65	100	5000	27 600
Модифицированный алгоритм с построением *FP*-дерева	530	790	38 000	210 000
Модифицированный алгоритм DBSCAN	580	610	46 000	290 000
Модифицированный алгоритм *k*-средних	600	1 200	62 000	> 360 000
DBSCAN + Word2Vec	3 500	3 700	300 000	> 360 000

Отметим, что предложенный алгоритм кластеризации на основе *FP*-дерева требует построения дерева на одном узле кластера. Это означает, что при достаточно больших массивах текстов построение дерева может занять значительное время. Ограничение на максимальный размер дерева поможет в решении этой проблемы, однако это ограничение неизбежно скажется на качестве кластеризации. Метрическая кластеризация на основе векторных представлений текстов имеет производительность меньше остальных алгоритмов, поскольку подавляющая часть времени векторной кластеризации приходится на обучение модели Word2Vec. Следовательно, пожертвовав качеством векторных представлений, скорость работы этого метода можно увеличить.

Заключение. Выполнен обзор основных методов, которые можно применить при кластеризации шаблонных текстов в больших массивах данных с использованием распределенных вычислений. Предложены метод, сочетающий в себе метрическую кластеризацию с алгоритмами выравнивания

биоинформатики, и модификация *FP*-дерева, позволяющая проводить кластеризацию с использованием распределенных систем. Выполнено исследование производительности разработанных алгоритмов и оценено качество кластеризации. Показано, что предложенные алгоритмы могут иметь преимущество по качеству лишь в задачах с повышенной значимостью точности перед полнотой, но при этом имеют более высокую производительность при использовании на кластере.

Повышенная значимость метрики точности перед метрикой полноты в задаче выделения спама из общего трафика позволяет с помощью предложенных методов добиться лучших результатов на больших объемах коротких текстов.

В зависимости от постановки задачи и данных результаты кластеризации могут отличаться, поскольку понимание того, какие тексты считать шаблонными, может различаться. Поэтому окончательный выбор того или иного алгоритма не может быть основан только на приведенном в настоящей работе результате исследования производительности и требует дополнительного анализа эффективности выбранного метода при решении поставленной задачи.

ЛИТЕРАТУРА

[1] Кириченко К.М., Герасимов М.Б. Обзор методов кластеризации текстовой информации. *Dialogue 2001*. М., 2001.
URL: https://www.dialog-21.ru/digest/2001/articles/kirichenko (дата обращения: 20.12.2021).

[2] *Apache Spark: веб-сайт*. URL: https://spark.apache.org (дата обращения: 20.12.2021).

[3] Mikolov T., Chen K., Corrado G., et al. Distributed representations of words and phrases and their compositionality. *Proc. 26th NIPS*, 2013, vol. 2, pp. 3111–3119.

[4] Огурцов А.Н. Основы биоинформатики. Харьков, ХПИ, 2013.

[5] Hartigan J.A., Wong M.A. A *K*-means clustering algorithm. *J. R. Stat. Soc. Ser. C Appl. Stat.*, 1979, vol. 28, pp. 100–108. DOI: https://doi.org/10.2307/2346830

[6] Tan P.N., Steinbach M.M., Kumar V. Introduction to data mining. New York, Addison Wesley, 2006.

[7] Гасфилд Д. Строки, деревья и последовательности в алгоритмах. СПб., Невский Диалект, 2003.

[8] Ukkonen E. On-line construction of suffix trees. *Algorithmica*, 1995, vol. 14, no. 3, pp. 249–260. DOI: https://doi.org/10.1007/BF01206331

[9] Sabbir A., Mohammad R.R., Motaher H., et al. An improved frequent pattern mining algorithm using suffix tree & suffix automata. Bangladesh, Dhaka, Univ. of Asia Pacific, 2014.

[10] Вирцева Н.С., Вишняков И.Э. Выявление и выделение шаблонов в массиве коротких сообщений. *Наука и образование: научное издание МГТУ им. Н.Э. Баумана*, 2016, № 10. DOI: 10.7463/1016.0848929

[11] Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов. *Доклады АН СССР*, 1965, т. 163, № 4, с. 845–848.

[12] Дубанов А.В. Сравнение исходных текстов программ путем выравнивания последовательностей токенов. *Инженерный журнал: наука и инновации*, 2014, № 9. DOI: https://doi.org/10.18698/2308-6033-2014-9-1318

[13] Yujian L., Bo L.A. Normalized Levenshtein distance metric. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2007, vol. 29, no. 6, pp. 1091–1095.
DOI: https://doi.org/10.1109/TPAMI.2007.1078

[14] Бойцов Л.М. Использование хеширования по сигнатуре для поиска по сходству. *Прикладная математика и информатика*, 2000, № 7, с. 135–153.

[15] Bahmani B., Moseley B., Vattani A., et al. Scalable K-Means++. *Proc. VLDB Endow.*, 2012, vol. 5, no. 7, pp. 622–633. DOI: http://dx.doi.org/10.14778/2180912.2180915

[16] Wang Y., Gu Y., Shun J. Theoretically-efficient and practical parallel. *Proc. SIGMOD'20*, 2020, pp. 2555–2571. DOI: https://doi.org/10.1145/3318464.3380582

[17] Apache ML Clustering — DBSCANCluster. *commons.apache.org: веб-сайт.*
URL: https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/ml/clustering/DBSCANClusterer.html (дата обращения 20.12.2021).

[18] Index of /ruwiki/. *dumps.wikimedia.org: веб-сайт*.
URL: https://dumps.wikimedia.org/ruwiki (дата обращения: 20.12.2021).

[19] Russian troll tweets. *kaggle.com: веб-сайт*.
URL: https://www.kaggle.com/vikasg/russian-troll-tweets?select=tweets.csv (дата обращения: 20.12.2021).

[20] *Apache Hadoop. hadoop.apache.org: веб-сайт*. URL: https://hadoop.apache.org (дата обращения: 20.12.2021).

Вишняков Игорь Эдуардович — старший преподаватель кафедры «Теоретическая информатика и компьютерные технологии» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Иванов Игорь Потапович — д-р техн. наук, доцент, заведующий кафедрой «Теоретическая информатика и компьютерные технологии» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Каркин Илья Александрович — бакалавр, кафедра «Теоретическая информатика и компьютерные технологии» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Просьба ссылаться на эту статью следующим образом:

Вишняков И.Э., Иванов И.П., Каркин И.А. Выявление и кластеризация шаблонных текстов в больших массивах сообщений. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2022, № 4 (141), с. 20–35.
DOI: https://doi.org/10.18698/0236-3933-2022-4-20-35

IDENTIFICATION AND CLUSTERING OF TEMPLATE TEXTS IN THE LARGE ARRAYS OF MESSAGES

I.E. Vishnyakov vishnyakov@bmstu.ru
I.P. Ivanov ivanov@bmstu.ru
I.A. Karkin karkinia@yandex.ru

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

A lot of services are using short messages for various purposes today, for example, stores are sending promotional offers, and EMERCOM of Russia informs population in the event of a threat of natural and technogenic emergencies. Selecting short texts of the template messages from general traffic could be used to filter spam and mailings, as well as to protect users from fraudulent activities. Such arrays of messages are often reaching such a large size that their storage and processing on a single dedicated personal computer or server becomes practically impossible. This work aims at developing approaches to the efficient identification and clustering of the template texts from large arrays of the short messages using the Apache Spark framework for distributed processing of the unstructured data. Main approaches to identifying templates and clustering textual information are considered. Approaches were developed making it possible to cluster in large arrays of messages using distributed computation without preliminary acquisition of the text vector representations. Algorithms are provided for efficient identification of the template messages from large arrays of short texts. Algorithms were compared in terms of performance and quality of pattern identification

Keywords

Pattern identification, text clustering, big data

Received 16.06.2022
Accepted 27.06.2022

REFERENCES

[1] Kirichenko K.M., Gerasimov M.B. [Review on text information clustering methods]. *Dialogue 2001*. Moscow, 2001 (in Russ.).
Available at: https://www.dialog-21.ru/digest/2001/articles/kirichenko (accessed: 20.12.2021).

[2] *Apache Spark*: *website*. Available at: https://spark.apache.org (accessed: 20.12.2021).

[3] Mikolov T., Chen K., Corrado G., et al. Distributed representations of words and phrases and their compositionality. *Proc. 26th NIPS*, 2013, vol. 2, pp. 3111–3119.

[4] Ogurtsov A.N. Osnovy bioinformatiki [Basics of bioinformatics]. Kharkov, KhPI Publ., 2013.

[5] Hartigan J.A., Wong M.A. A *K*-means clustering algorithm. *J. R. Stat. Soc. Ser. C Appl. Stat*., 1979, vol. 28, pp. 100–108. DOI: https://doi.org/10.2307/2346830

[6] Tan P.N., Steinbach M.M., Kumar V. Introduction to data mining. New York, Addison Wesley, 2006.

[7] Gusfield D. Algorithms on strings, trees, and sequences. Cambridge, Cambridge University Press, 1997.

[8] Ukkonen E. On-line construction of suffix trees. *Algorithmica*, 1995, vol. 14, no. 3, pp. 249–260. DOI: https://doi.org/10.1007/BF01206331

[9] Sabbir A., Mohammad R.R., Motaher H., et al. An improved frequent pattern mining algorithm using suffix tree & suffix automata. Bangladesh, Dhaka, Univ. of Asia Pacific, 2014.

[10] Virtseva N.S., Vishnyakov I.E. Pattern identification and retrieval in short text messages. *Nauka i obrazovanie: nauchnoe izdanie MGTU im. N.E. Baumana* [Science and Education: Scientific Publication], 2016, no. 10 (in Russ.). DOI: 10.7463/1016.0848929

[11] Levenshteyn V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady AN SSSR*, 1965, vol. 163, no. 4, pp. 845–848 (in Russ.).

[12] Dubanov A.V. The comparison of program sources using the sequence alignment of tokens. *Inzhenernyy zhurnal: nauka i innovatsii* [Engineering Journal: Science and Innovation], 2014, no. 9 (in Russ.).
DOI: https://doi.org/10.18698/2308-6033-2014-9-1318

[13] Yujian L., Bo L.A. Normalized Levenshtein distance metric. *IEEE Trans. Pattern Anal. Mach. Intell*., 2007, vol. 29, no. 6, pp. 1091–1095.
DOI: https://doi.org/10.1109/TPAMI.2007.1078

[14] Boytsov L.M. Using signature hashing for fuzzy string search. *Prikladnaya matematika i informatika*, 2000, no. 7, pp. 135–153 (in Russ.).

[15] Bahmani B., Moseley B., Vattani A., et al. Scalable K-Means++. *Proc. VLDB Endow*., 2012, vol. 5, no. 7, pp. 622–633. DOI: http://dx.doi.org/10.14778/2180912.2180915

[16] Wang Y., Gu Y., Shun J. Theoretically-efficient and practical parallel. *Proc. SIGMOD'20*, 2020, pp. 2555–2571. DOI: https://doi.org/10.1145/3318464.3380582

[17] Apache ML Clustering — DBSCANCluster. *commons.apache.org: website.*
Available at: https://commons.apache.org/proper/commonsmath/javadocs/api-3.6.1/org/apache/commons/math3/ml/clustering/DBSCANClusterer.html
(accessed: 20.12.2021).

[18] Index of /ruwiki/. *dumps.wikimedia.org: website*.
Available at: https://dumps.wikimedia.org/ruwiki (accessed: 20.12.2021).

[19] Russian troll tweets. *kaggle.com: website.*
Available at: https://www.kaggle.com/vikasg/russian-troll-tweets?select=tweets.csv
(accessed: 20.12.2021).

[20] *Apache Hadoop: website*. Available at: https://hadoop.apache.org (accessed: 20.12.2021).

Vishnyakov I.E. — Senior Lecturer, Department of Theoretical Informatics and Computer Technologies, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Ivanov I.P. — Dr. Sc. (Eng.), Assoc. Professor, Head of the Department of Theoretical Informatics and Computer Technologies, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Karkin I.A. — Bachelor, Department of Theoretical Informatics and Computer Technologies, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Please cite this article in English as:

Vishnyakov I.E., Ivanov I.P., Karkin I.A. Identification and clustering of template texts in the large arrays of messages. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2022, no. 4 (141), pp. 20–35 (in Russ.).
DOI: https://doi.org/10.18698/0236-3933-2022-4-20-35