Обзор методов кластеризации больших объемов текстовых записей

Фельдман Даниил

Московский физико-технический институт

Постановка задачи

Дано: 30 млн. библирграфических записей.

Цель: необходимо найти дубликаты произведений.

Проблемы:
1. Поиск дубликатов при помощи попарного сравнения занимает много времени.
2. Данные плохо организованы: у двух дубликатов авторы, названия, аннотации могут быть записаны разными способами.
3. Много пустых записей.

Задача:
1. Придумать механизм предкластеризации объектов, чтобы дубликаты искать уже внутри сформированных групп.
2. Попытаться придумать/найти алгоритм, который бы позволил произвести кластеризацию быстрее, чем за $O(n^2)$.

План презентации

- Способы отображения объектов в векторное пространство с известной мерой
  1. Broder's algoritm
  2. Charikar's algoritm (simhash)
- Некоторые алгоритмы кластеризации
  1. Center & Merge-cente
  2. Star clustering algoritm
  3. Ricochet algoritms
  4. Articulation Point Clustering
  5. Обзор результатов работы алгоритмов
- Результаты работы некоторых алгоритмов

Broder's algoritm

Описание алгоритма:
Пусть каждый объект представлен набором токенов длины n.

- Каждое подмножество длины k заменяется своим цифровым отпечатком
- Мы получаем последовательность из $n - k + 1$ отпечатков - шинглы (shingles)
- Берется семейство хеш-функций f_i, $1 \leq i \leq m$
- Для каждой хеш-функции f_i находятся $n - k + 1$ значений, соответствующим шинглам. Для каждого i выбирается наименьшее из них - minvalue.
- Мы получили для объекта признаковый вектор длины m.

Broder's algoritm

В алгоритме предполагается, что схожесть объектов можно измерять числом шинглов, совпадающих у них:

$$\frac{|S(d) \cap S(d')|}{|S(d) \cup S(d')|}$$

Бродер показал, что количество одинаковых элементов в векторах, соответствующих d и d' равно числу различных совпадающих шинглов в $S(d)$ и $S(d')$.
То есть в качестве меры схожести двух объектов можно рассматривать число соответствующих элементов в векторах minvalue.
Параметры, которые задают: k, m.
Существует ускоренная версия алгоритма (из minvalue семейством хеш-функций получаем supershingle).

Charikar's algoritm

Опишем алгоритм:

- Выберем константу b
- Каждый токен отображается в b-мерное пространство случайным выбором b элементов из множества $\{-1, 1\}$. Важно, что отображение будет одинаковым для всех документов.
- Для каждого объекта d мы суммируем проекции на выбранное b-мерное пространство всех токенов, входящих в него $\Rightarrow$ получаем вектор $f(d)$.
- Признаковый вектор: если $f(d)[i] \geq 0$, то i-й элемент 1, иначе 0.

Charikar's algoritm

Мера схожести d и d' - число битов, в которых их принаковые векторы совпадают. Она совпадает с мерой Жаккара для этих объектов.
Два элемента называются почти одинаковыми, если их мера схожести превосходит некоторого порога.

Разница Charikar's algoritm и Broder's algoritm:
Первый учитывает число вхождений токена, но не учитывает их порядок, второй наоборот.

T - общее число токенов во всех документах. Тогда алгоритм Бродера - за $O(Tm)$, Чарикара - за $O(Tb)$.

Предразбиение на группы

Мы можем составить предварительный список дубликатов.

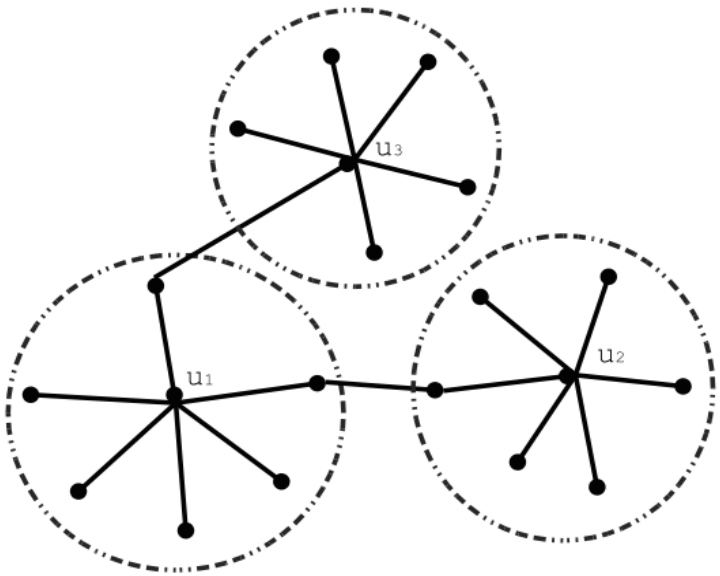
Пусть каждому объекты сопоставлен вектор длины n из minvalue или simhash. Тогда кандидатами в дубликаты мы можем считать те, у которых некоторое подмножество элементов (например только $\lfloor \frac{n}{2} \rfloor$) совпадает.

Мы получим некоторое число пересекающихся групп, и последующую кластеризацию будем производить только в их пределах.

CENTER

Пусть $G = (V, E)$ - граф смежности объектов в базе: объекты u и v связаны, если их схожесть $sim()$ превосходит некоторый порог t. Задача сводится к кластеризации вершин графа.

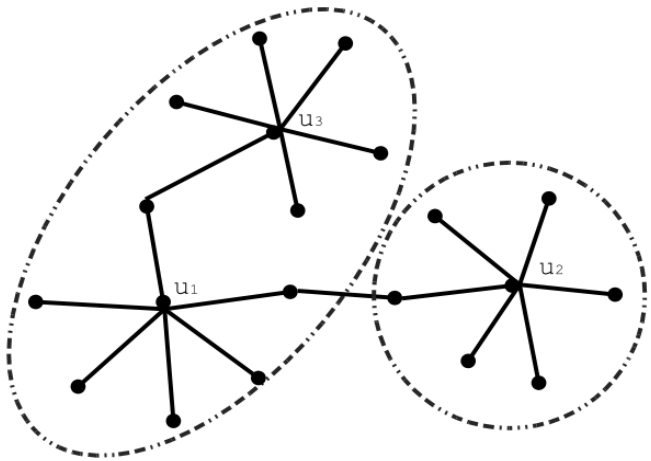
Пусть P - множество пар похожих вершин (u, v), отсортированных по убыванию $sim(u, v)$ Алгоритм CENTER проводит кластеризацию за один проход P. Первый раз, когда встречается вершина u, она становится центром кластера. Все вершины v из пар (u, v) относятся к тому же кластеру и больше не рассматриваются. Алгоритм работает за $O(|E| \log |E|)$.



MERGE-CENTER

Похож на CENTER, но если в кластере c_j есть вершина, похожая на центр кластера c_i, кластеры сливаются. Первый раз, когда встречается вершина u, она становится центром кластера. Все вершины v из пар (u, v), которые не принадлежат никакому кластеру, переходят в него и более не рассматриваются. Если встречается пара (u, v'), и v' уже принадлежит некоторому кластеру, то они сливаются. Алгоритм работает за $O(|E| \log |E|)$.

- При правильном пороге дубликаты будут отслежены
- Создает непересекающиеся кластеры
- Существует расширение для пересекающихся кластеров

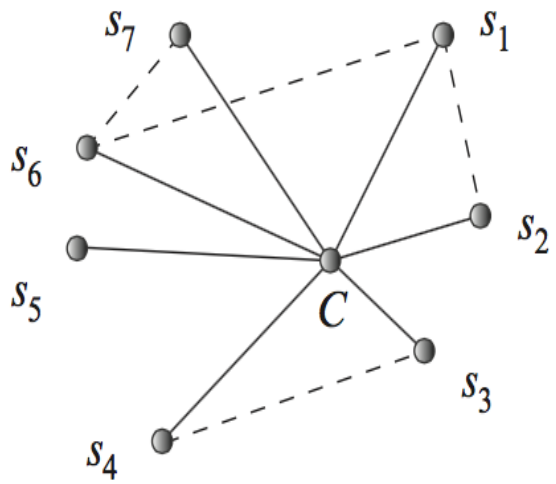


(c) MERGE-CENTER (MC)

Star clustering algoritm

Смысл алгоритма: Убираем из графа ребра с величиной меньше порога Θ; Строим минимальную клику на вершинах графа. Это обеспечивает то, что в каждом кластере похожесть объектов превосходит установленный порог. Однако задача получается *NPC*, что не подходит. Однако мы можем позволить покрытие графа подграфами звездной формы. Сперва кажется, что между листьями не будет обеспечен нужный порог схожести, однако можно, сдвинув порог, добиться нужной точности. Положительные моменты:

- Элементы в кластере с установленной точностью
- Кластеры могут пересекаться



Star clustering algoritm

Приведем сам алгоритм:

- Для установленного порога Θ находим граф $G_\Theta = (V, E_\Theta)$, где $E_\Theta = \{e \in E | \ w(e) \geq \Theta\}$
- Пусть каждая вершина в графе изначально не помечена
- Считаем степень каждой вершины $v \in V$
- Непомеченная вершина с наибольшей степенью становится центром кластера-звезды, в которую входят смежные вершины. Вершины из нового кластера помечаются
- Повторяем предыдущий шаг, пока все вершины не помечены
- Получаем множество кластеров

В статье "The Star Clustering Algorithm for Information Organization by J.A. Aslam, E. Pelekhov, and D. Rus" подробно изложены теоретические выкладки и модификации алгоритма.

Ricochet family of algorithms

Семейство алгоритмов, которые работают в два этапа с некоторыми альтернативами:
1. Определяются ядра (что-то, напоминающее определение центров кластеров-звезд)
2. Вершины добавляются в кластеры, связанные с ядрами (что-то, напоминающее алгоритм k-средних)

Создатели алгоритма предложили 4 варианта (в отличаются выбором ядер - одно за другим, либо конкурентно; приводит к пересекающимся, либо непересекающимся кластерам). Рассмотрим планы этих алгоритмов.

Ricochet family of algorithms

1) Sequential Rippling (SR):
Сначала вершины сортируются по убыванию их весов (средний вес их ребер). Новые ядра выбираются по одному из этого списка. Когда добавляется новое ядро, вершины перераспределяются в новый кластер, если они ближе к новому ядру, чем к нынешнему. Если никаких перемен не произошло, то новый кластер не создается. Если кластер вырождается в одну вершину, он примыкает к ближайшему ядру. Алгоритм заканчивает работу, когда не осталось вершин без кластера.

2) Balanced Sequential Rippling (BSR):
Похож на SR в выборе первого ядра в во второй фазе алгоритма. Новое ядро выбирается таким образом, чтобы максимизировать отношение его веса к сумме схожести с существующими ядрами. Смысл заключается в том, чтобы выбирать новое ядро не только с большим весом, но и достаточно далеко от существующих.

Ricochet family of algorithms

3) Concurrent Rippling (CR):
Изначально все вершины помечены как ядра. На каждой итерации алгоритм выбирает для каждого ядра ребро с наибольшим весом. Если это ребро соединяет соединяет ядро с некоторой вершиной, которая не является ядром, то вершину относят к кластеру того ядра. Если вершина является ядром, то ее относят к кластеру, только если ее вес меньше веса ядра. Итерация проводится для всех ядер с равным темпом.

4) Ordered Concurrent Rippling (CR):
Похоже на CR, за исключением требования, что для всех ядер итерация проводится с равным темпом.

Articulation Point Clustering

Алгоримт основан на нахождении точек связности и бисвязных компонент.
Вершина называется точкой связности, если ее удаление (вместе с ребрами) превращает граф в несвязный. Компонента называется бисвязной, если в ней нет точек связности.

Нахождение бисвязных компонент в графе - известная задача, которая решается за линейное время. Удаление точек связи разделяет граф на бисвязные компоненты. Эти копоненты и становятся кластерами (могут пересекаться).

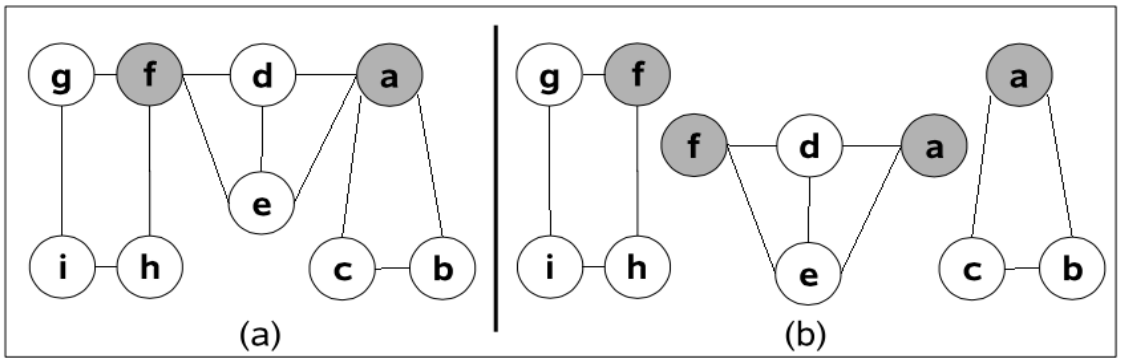


Figure 4: (a) Articulation points are shaded, (b) Biconnected components.

Меры схожести текстовых документов

Рассмотрим некоторые "меры" схожести, которые используются для оценки схожести текстовых документов.

- Jaccard and weighted Jaccard
- Edit similarity
- Cosine

Jaccard and weighted Jaccard

Пусть текстовые файлы уже токенизированы. Рассмотрим два набора токенов r_1 и r_2.

$$sim_{wJaccard}(r_1, r_2) = \frac{\sum\limits_{t \in r_1 \cap r_2} w(t)}{\sum\limits_{t \in r_1 \cup r_2} w(t)}$$

Здесь t - токен, а $w(t)$ - вес токена.

В случае, если $w(t) = 1 \; \forall t$ имеем обычную меру Jaccard.

Edit similarity

Эта мера часто используется в совокупности с другими. Пусть мы имеем два набора токенов r_1 и r_2. Смысла меры Edit similarity заключается в стоимоти (количестве операций) трансформации r_1 в r_2.

Под операциями имеется в виду замена символа, удаление или добавление (может быть разная стоимость).

$$sim_{edit}(r_1, r_2) = 1 - \frac{tc(r_1, r_2)}{max(|r_1|, |r_2|)}$$

здесь $tc(r_1, r_2)$ - transformation cost

Cosine

Смысл косинусной меры в том, что сначала объекты переводятся в вектора, а затем между ними находится угол. Пусть r_1, r_2 - наборы токенов

$$sim_{Cosine}(r_1, r_2) = \sum_{t \in r_1 \cap r_2} w_{r_1}(t) \cdot w_{r_2}(t)$$

Здесь $w_{r_1}(t)$ и $w_{r_2}(t)$ - веса токена.
Веса определяют как

$$w_r(t) = \frac{w'(t)}{\sqrt{\sum_{t' \in r} w'_r(t')^2}}$$

$$w'_r(t) = tf_r(t) \cdot idf(t)$$

tf_r - term frequency; idf - inverse document frequency

Данные для тестирования

Рассмотрены три датасета. В каждом из них некоторая часть записей намеренно испорчена (заменой некоторых токенов местами).

Использованные датасеты

описание	название	доля испорченных дубликатов	доля ошибок в записи
высокая ошибка	HE	90	30
средняя ошибка	ME	50	10
низкая ошибка	LE	10	10

Размер и число кластеров в датасетах

название	источник	размер	число кластеров
HE	названия компаний	5000	500
ME	названия компаний	5000	500
LE	названия компаний	5000	500

Метрики качества

Пусть истинные кластеры - $G = \{g_1, ..., g_n\}$, найденные кластеры - $C = \{c_1, ..., c_n\}$. Отнесем каждому c_j кластер g_i так, что у них наибольшее число совпадающих элементов, тогда $f(g_i) = argmax_{c \in C}\{|g_i \cap c|\}$.

$$Pr_i = \frac{|f(g_i) \cap g_i|}{|f(g_i)|} \qquad Re_i = \frac{|f(g_i) \cap g_i|}{|g_i|}$$

Precision, Recall и F-мера:

$$Pr = \sum_i \frac{|g_i|}{|R|} Pr_i \qquad Re = \sum_i \frac{|g_i|}{|R|} Re_i \qquad F_1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$$

Также для оценки качества самой кластеризации рассмотрим показатель CP_{r_i}, который выражает способность алгоритма относить элементы их одного кластеру в единый кластер. CP_r среднее по CP_{r_i}.

$$CP_{r_i} = \frac{|(t,s) \in c_i \times c_i|\ t \neq s\ \wedge \exists j \in 1, ..., k,\ (t,s) \in g_j \times g_j|}{C^2_{k'}}$$

Результаты Center и Merge-Center

результаты работы

	Pr	Re	F_1	CP_r
Center	0,971	0,805	0,877	0,692
Merge-Center	0,958	0,885	0,918	0,795

Мы видим, что для нашей задачи лучше подходит Merge-Center, так как Center выдает хороший precision и recall, но менее точно определяет число кластеров.

Результаты Star clustering

Будем рассматривать результаты алгоритма при разной величине порога θ

результаты работы

	Pr	Re	F_1	CP_r
$\theta = 0,2$	0,588	0,730	0,644	0,726
$\theta = 0,3$	0,778	0,842	0,805	0,801
$\theta = 0,4$	0,900	0,870	0,884	0,781

При более высоком пороге алгоритм показывает лучшие результаты в плане точности.

Спасибо за внимание.