Обзор методов кластеризации больших объемов текстовых записей

Ринат Хайруллин

Московский физико-технический институт (ГУ)

Постановка задачи

Имеется база данных с большим числом (21000000) текстовых файлов, среди которых встречаются похожие. Задача, в первую очередь, заключается в том, чтобы отнести к одному кластеру дубликаты. При этом желательно уменьшить число попарных сравнений объектов. Можно попытаться ускорить работу двумя способами.

Во-первых, предкластеризацией объектов можно уменьшить число попарных сравнений, сравнивая их только в пределах групп. Однако, в таком случае увеличивается суммарная ошибка кластеризации Во-вторых, сложность можно уменьшить, применив алгоритм, который работает быстрее, чем за $O(n^2)$.

Проблема также заключается в том, что данные плохо организованы: у двух дубликатов авторы, названия, аннотации могут быть записаны разными способами. Кроме того, некоторые значения могут быть нулевыми, и у одного текста может быть несколько авторов, они могут быть записаны в разном порядке.

Обработка текстовых записей

Мы хотим перевести текстовые объекты в векторное пространство признаков с известной меры, чтобы сравнивать не сами объекты, а их короткие заменители. Рассмотрим алгоритмы, которые позволяют это сделать (будем считать, что текст уже разбит на токены - слова, биграммы, триграммы и т. д.):

1) Simhash:

- 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.
- Мы посредством хеш-функций f_i получили для каждого токена отображение.
- Для каждого объекта d мы получаем вектор $f(d)$: суммируем проекции всех токенов, входящих в него.
- Признаковый вектор: если $f(d)[i] >\geq 0$, то i-й элемент 1, иначе 0.

Мера схожести d и d' - число битов, в которых их принаковые векторы совпадают. Два элемента называются почти одинаковыми, если их мера схожести превосходит некоторого порога T.

2) Broder's algortm: Пусть у каждый объект представлен набором токенов длины n.

- Каждое подмножество длины k заменяется своим цифровым отпечатком (предлагается использовать 64-битные отпечатки Рабина)
- 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.

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

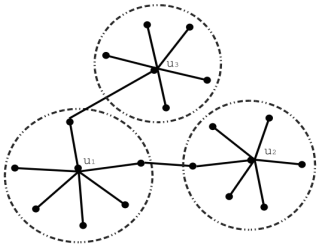
$$\frac{|S(d) \cap S(d')|}{|S(d) \cup S(d')|}$$

Бродер показал, что количество одинаковых элементов в векторах, соответствующих d и d' равно числу различных совпадающих шинглов в $S(d)$ и $S(d')$.

Алгоритмы кластеризации

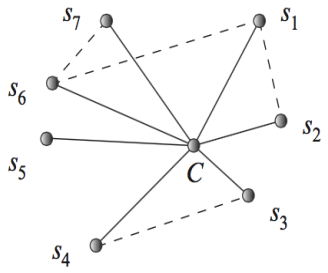
1) Center: Пусть у нас есть граф смежности объектов в базе: объекты u и v связаны, если из схожесть $sim()$ превосходит некоторый порог t. Задача сводится к кластеризации вершин графа.

Алгоритм CENTER проводит кластеризацию за один проход по отсортированным парам (u, v) похожих вершин. 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.



2) Star clustering:

- 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



3) Ricochet family of algoritms: Рассмотрим один из алгоритмов семейства (Sequential rippling). 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.

4) 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 не бисвязные компоненты. Эти копоненты и становятся кластерами (могут пересекаться).

Меры схожести текстовых документов

1) 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.

2) 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

3) 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

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

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

tf_r - term frequency; idf - inverse document frequency

Использованная литература

[1] D. T. Wijaya and S. Bressan. Ricochet: A Family of Unconstrained Algorithms for Graph Clustering.

[2] A. Broder, Identifying and filtering near-duplicate documents, Proc. Annual Symposium on Combinatorial Pattern Matching, Lecture Notes in Computer Science (eds. R. Giancarlo and D. Sankoff) 1848, 2000, pp.(1–10)

[3] Charikar's M., 2002. "Similarity estimation techniques from rounding algorithms", In Proc. 34th Annual Symposium on Theory of Computing (STOC 2002), pp. 380-388.

[4] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable Techniques for Clustering the Web. In Proc. of the Int'l Workshop on the Web and Databases (WebDB), pages 129–134, Dallas, Texas, USA, 2000.

[5] J. A. Aslam, E. Pelekhov, and D. Rus. The Star Clustering Algorithm For Static And Dynamic Information Organization. Journal of Graph Algorithms and Applications, 8(1):95–129, 2004.