Алгоритм Fair match

Беребердина Наталья

211 группа

В работе мы моговорим об алгоритме Fair match основная цель которого максимизировать разнообразие предлагаемых пользователям товаров. Простые рекомендательные системы часто ориентированы на популярные товары и часто одни и те же позиции рекомендуется многим людям, в то время как большинству достаточно хороших позиций не уделяется должного внимания. Это приводит к низкому совокупному разнообразию, что в свою очередь мешает новым продавцам развиваться и лишает новый продукт возможности найти своего потребителя. Все это стопорит торговлю. Для решения этой проблемы авторы статьи "FairMatch: A Graph-based Approach for Improving Aggregate Diversity in Recommender Systems" предлагают использовать алгоритм fairMatch принцип работы которого будет описан далее.

Пусть у нас уже есть алгоритм, который решает насколько товар привлекателен для пользователя. Мы хотим в итоге показать каждом пользователю n товаров. Попросим нашу систему порекомендовать каждому пользователю по t товаров (t » n) так, чтобы последний товар в этом списке все еще оставался достаточно хорошим для пользователя. Теперь построим двудольный граф, где вершины правой доли будут пользователями, а вершины левой доли будут товарами. Ребрами мы будем соединять товар i с пользователем j, если i есть в t-списке у j. Кроме этого добавим в граф две вершины - исток, соединенный со всеми товарами и сток, соединенный со всеми пользователями. Получили граф, где у всех правых вершин степень t + 1, а у левых вершин степень различна. Теперь перейдем к расстановке пропускной способности.

На ребрах между пользователями и товарами будем вычислять пропускные способности по формуле

$$\alpha * degree(item) + (1 - \alpha) * rank_{user}(item)$$

где $degree(item)$ - количество ребер ведущих из конкретного товара в пользователей, $rank_{user}(item)$ - номер товара в t-списке рекомендаций пользователя. То есть, чем меньше ребер из айтема, тем меньше пропускная способность ребра и чем лучше предмет для человека, тем меньше пропускная способность ребра. Коэффициент α регулирует что важнее. Во все вершины в левой доле из истока делаем один вес - суммарная пропускная способность меду левой и правой долями деленная на количество вершин в левой доле ("средняя пропускная способность

товаров"). Во все вершины из правой доли делаем один вес в сток - та же суммарная пропускная способность, только деленная на количество вершин в правой доли ("средняя пропускная способность пользователей").

Мы как раз хотим продвигать непопулярные хорошие предметы, то есть для нас интересны ребра маленького веса. Воспользуемся алгоритмом проталкивания с предпотоком. Давайте кратко сформулируем принцип его работы, чтобы обосновать использование результатов его работы. В проталкивании с предпотоком каждая вершина имеет две характеристики: высота вершины и накопленный поток (входящий минус исходящий). Также мы умеем делать две операции: поднимать высоту вершины и выпускать поток в более низкую точку. Поставим изначальную высоту у истока - количество вершин в графе, у товаров 1, у пользователей и стока 0. Пустим во все вершины одинаковый полный поток из истока в товары. Максимальный поток достигается, когда у всех вершин, кроме стока накопленный поток 0. При этом давайте заметим, что если мы пустим одинаковый поток во все товары, то "недооцененные" товары будут иметь сумарно низкую пропускную способность - у них мало исходящих ребер и эти ребра имеют маленькие пропускные способности. Значит после такого запуска из истока во все товары в этих вершинах образуется излишек. Чтобы поток не задерживался нам надо будет перелить из таких товаров поток обратно в исток, а для этого нам нужно будет поднять высоту. То есть у них вход будет как у всех, а вытекает мало потока, значит их полностью насытим и поднимем по высоте. Найдя максимальный поток выделим вершины на высоте не меньше чем количество ребер. Эти вершины перенасыщались, значит они недооценены. Выкинем теперь эти товары вместе с выходящими ребрами и назовем это множество M. Снова повторим на нашем новом графе алгоритм проталкивания с предпотоком (пересчитывая пропускные способности) и снова найдем недооцененные вершины, добавим их в множество M и удалим из графа. Повторяя такую операцию, пока в M будут добавляться вершины, мы получим в итоге множество всех недооцененных товаров.

Теперь возьмем у каждого пользователя список n самых подходящих ему товаров. Отсортируем его по убыванию количества ребер у товара. Теперь, для товаров из M, рекомендованных пользователю, но не входящих в топ n для него, заменим самые популярные товары в его списке (первые по порядку) на наши товары из M. Теперь в его списке больше недооцененных товаров. А суммарное разнообразие улучшилось.

Следующей частью моей работы было написание алгоритма FairMatch. Рассмотрим его работу на трех простых примерах.

Возьем систему, где 3 пользователя, n = 1, t = 2, со следующими рекомендациями:
$user_0 - item_0, item_1$
$user_1 - item_0, item_2$
$user_2 - item_0, item_2$
В первом случае рассмотрим $\alpha = 1$. Тогда наибольшее влияние для нас будет

играть степень вершин. Посчитаем пропускные способности ребер между пользователями и товарами - они будут равны в точности степеням вершин товаров:

	user_0	user_1	user_2
item_0	3	3	3
item_1	1	0	0
item_2	0	2	2

Пропускные способность ребер выходящи из стока и истока будут одинаковыми:

$$SumCapacity/|U| = SumCapacity/|I| = (3+3+3+2+2+1)/3 = 4.(6)$$

Округляя получим пропускную способность 5. Тогда, когда мы пустим поток 5 во все товары, из вершин $item_1$ - с максимальной пропускной способностью 1 и $item_2$ - с максимальной пропускной способностью 4 поток вернется в исток. Значит их высота станет хотя бы высотой истока и они попадут в множество недооцененных. Выикнем эти вершины и пересчитаем пропускные способности. Между левой и правой долью они не изменятся (так как не изменились степени вершин оставшихся товаров), а между истоком и левой долью будет идти единственное ребро с весом 9. Весь поток идуший в $item_0$ будет расходиться поровну во всех пользователейи дальше в сток. Таким образом, множство недооцененных товаров состоит из $item_1$ и $item_2$.

Теперь сделаем замены в списках. Мы хотим оставить 1 элемент, самый подходящий, а потом заменить его на недооцененный элемент рекомендованный этому пользова- телю. Таким образом, у первого пользоателя мы заменим предмет $item_0$ на $item_1$, у второго и третьего $item_0$ на $item_2$.

Рассмотрим вторй случай на том же наборе рекомендаций, но с $\alpha = 0.5$. Теперь нам одинаково важны редкость и качество для пользователя. Таким образом, $item_1$ окажется достаточно редким, а $item_0$ - достаточно качественным. Можно предположить, что именно они попадут в список недооцененных и тогда рекомендации изменятся только у первого пользователя: с $item_0$ на $item_1$

И в последнем случае положим $\alpha = 0$. Тут редкость элемента не будет играть роли, а значит всем пользователям должен порекомендоваться $item_0$.

Моя программа выдает на этих данных следующие результаты:

$$\alpha = 1, 1, 2, 2$$

$$\alpha = 0.5, 1, 0, 0$$

$$\alpha = 0, 0, 0, 0$$

Это полностью соотвествует нашей ручной проверке первого случая, а также гипотезам во втором и третьем случаях. Таким образом, этот алгоритм дает вполне адекватные результаты даже на маленьких примерах.