О поведении поисковых роботов

А. А. Карташов

1. Введение

В наше время появилось огромное число информационных сетей. Их размеры все время увеличиваются, и найти в них нужную информацию становится очень сложно. Рассмотрим самую большую информационную сеть — Интернет, и представим, что нам нужно найти научные статьи на тему «Нечеткая математика». Эта проблема не столь легкая, так как неизвестно где их нужно искать, не говоря о том, сколько вы потеряете при этом своего времени.

Данная проблема рассматривалась во многих работах, написанных в последнее время. Существует несколько подходов для решения этой проблемы. Первый подход, предложенный в работах [1, 3], основан на идее изменения самой сети, чтобы она поддавалась легкому поиску документов. Иными словами, проиндексировать все документы в сети и ввести правило добавления, удаления и изменения документа в сети. К сожалению, этот подход практически нереализуем в рамках сети Интернет, где нет никаких правил добавления или изменения документов, каждый человек вправе изменить, добавить или удалить ресурс, никого не оповещая. Второй подход, который используют почти все поисковые системы в сети Интернет [2, 6], подразумевает постоянное исследование сети, и таким образом создание индексов, изменяющихся со временем. Находятся новые, удаленные или измененные документы. Этот подход также имеет свои недостатки. Во-первых, таким образом сеть очень сильно перегружается из-за необходимости проводить постоянный мониторинг, во-вторых, в большой сети невозможно быстро отследить все изменения. Третий подход, предложенный в [4, 5], состоит в том, что исследование надо начинать с некоторого документа в сети и попытаться выйти на нужный документ, проходя по ребрам в сети и все время двигаясь в направлении «лучших» документов. Алгоритм, предложенный в этой работе, действует в рамках третьего подхода.

От пользователя требуется только сравнивать документы, все остальное будет делать поисковый робот. Цель данной работы — описать алгоритм работы поискового робота, и показать, что данный алгоритм решает нашу задачу.

2. Постановка проблемы

Задача состоит в следующем:
В некоторой информационной сети, например Интернет, найти наиболее подходящий документ за как можно меньшее время, предполагая, что сравнивать документы мы можем без участия пользователя. Этот аспект проблемы не рассматривается в данной работе, а считается решенным. Решение проблемы сравнения документов можно найти в [8, 9].

Сделаем несколько предположений:

1) Информационную сеть можно рассматривать как граф, в котором вершины — это документы, а ребра — связи между ними.
2) Ребра графа несут информационную нагрузку: они связывают «похожие» документы, то есть имеющие нечто общее между собой в смысловом плане.
3) Граф слабо связный, то есть каждая вершина соединена с малым числом других вершин (например, не более 20).
4) В графе можно выделить много кластеров, которые можно объединить в большие кластеры, и т. д.

В дальнейшем будем говорить о сети Интернет, она наиболее интересна, хотя этот алгоритм можно применять в любых сетях, где верны вышесказанные предположения. Предположения 1–3 немного в другом виде были предложены Ф. Маерсом в его работе [5]. В комментариях нуждается лишь 4 предположение. Кластеры — это некоторые смысловые объединения, такие как Университет, кафедра, сайт магазина, сайт компании, отдел «мониторы» на сайте Sony, и т. д. Кластеры сильно связаны ребрами внутри себя и слабо связаны с другими кластерами. Это предположение можно проиллюстрировать на примере (см. рис. 1).

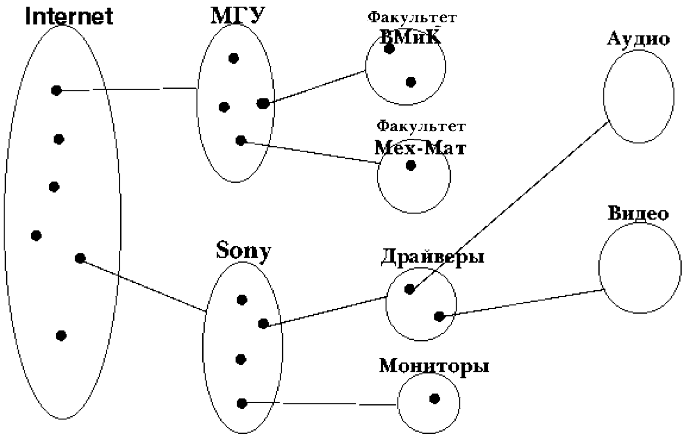

Рис. 1. Часть сети Интернет.

Из предположения 2 следует, что в кластерах примерно одинаковые документы, и чем ближе друг к другу находятся кластеры, тем более похожие документы в них находятся. Если мы знаем, что один документ из огромного кластера нам не подходит, с большой вероятностью не надо исследовать другие документы из данного кластера. Теперь становится очевидным, что для решения проблемы надо уметь выделять такие кластеры, и мы сможем быстро найти нужный документ, так как в процессе построения дерева из кластеров мы будем приближаться к кластеру с хорошими документами и игнорировать большое число ненужных документов, не рассматривая их.

3. Определение дерева кластеров

Для этого алгоритма потребуется граф, представляющий собой ту часть сети, которую мы уже исследовали, и дерево из кластеров.

Кластер — это вершина дерева, которая состоит из нескольких ссылок вниз (на следующие уровни), соединенные между собой по следующему принципу: два кластера соединены столькими ребрами, сколько ребер соединяет два подкластера, один из которых принадлежит первому кластеру, а другой второму. Этот принцип в дальнейшем мы будем называть принципом соединения кластеров (см. рис. 2).

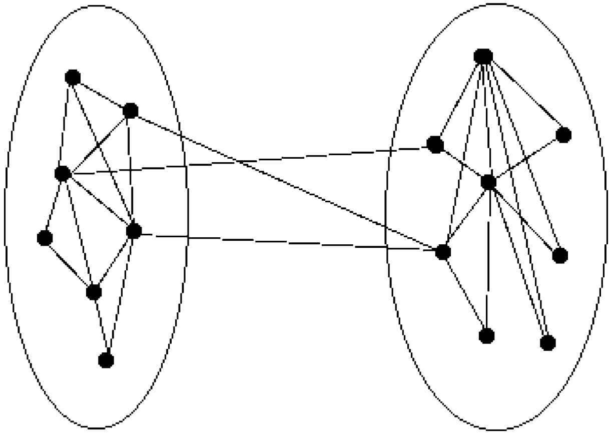

Рис. 2. Два кластера соединены 3-мя ребрами.

Лист дерева — это кластер, который не содержит ссылки вниз, а содержит в себе лишь документы, соединенные ребрами (см. рис. 3).

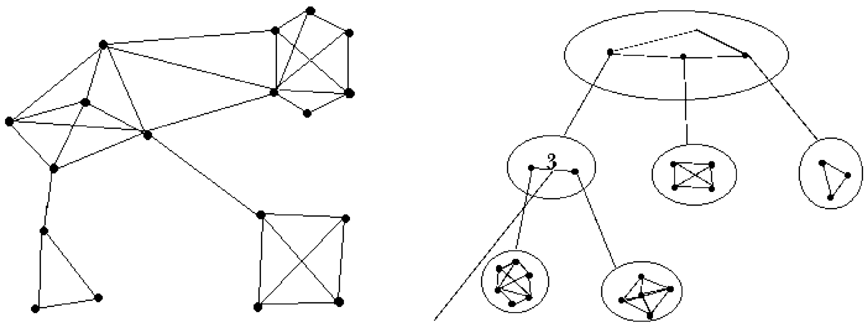

Рис. 3. Граф и дерево, построенное для него.

4. Операции, совершаемые над деревом

4.1. Добавление элемента в дерево

Пусть у нас уже имеется дерево. Новый элемент проходит сверху вниз по дереву и оседает в одном из листьев дерева. Опишем один шаг спуска.

На каждом этапе мы имеем документ и кластер, внутри которого несколько ссылок вниз на дочерние кластеры. Соединим документ с дочерними кластерами по принципу соединения кластеров. Находим дочерний кластер, с которым документ соединен самым большим числом ребер и относим документ к этому дочернему кластеру. Иными словами, мы ищем тот кластер, на который документ больше всего похож. Кластер, в который документ был опущен, надо соединить с другими кластерами, чтобы удовлетворить принципу соединения кластеров (см. рис. 4).

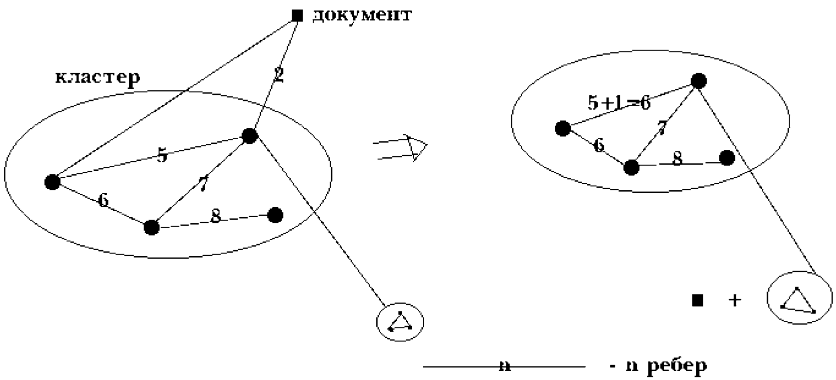

Рис. 4. Добавление документа в дерево.

Замечание. В результате добавления документа структура дерева не меняется, меняется лишь содержание кластеров.

Возникает новая проблема: некоторые кластеры будут увеличиваться в размере, поэтому в некоторые моменты времени надо выделять из них новые подкластеры. Мы также должны учитывать, что в процессе извлечения подкластеров мы можем ошибиться в их выборе из-за того, что на данный момент мы исследовали не всю сеть, а лишь ее часть. На приведенном рисунке (см. рис. 5) видно, что из-за того, что мы не исследовали вершины 5 и 7, мы не смогли правильно выделить подкластеры.

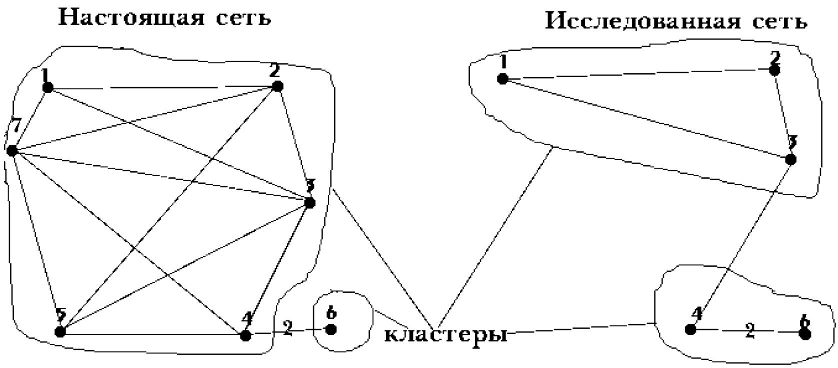

Рис. 5. Неправильное разделение на кластеры.

4.2. Разделение и слияние графов

Как было сказано выше, добавление элемента не меняет структуры дерева. Введем две операции, меняющие структуру дерева — это слияние деревьев и разбиение деревьев.

Слияние деревьев: При слиянии деревьев мы «сливаем» вершины двух деревьев в одну и получаем новое дерево, в вершине которого содержатся все ссылки вниз от первого и второго дерева и соединены они по принципу соединения кластеров. Слияние можно производить не только с целыми деревьями, но и с поддеревьями (см. рис. 6).

Разбиение дерева: При разбиении дерева мы разносим кластеры из вершины дерева в два разных кластера и в итоге получаем два дерева. Разделение также можно производить с поддеревьями (см. рис. 7).

Введем понятия наилучшего слияния и наилучшего разбиения.

Наилучшее слияние: Рассмотрим любую (не лист) вершину графа и ее кластеры, какие-то два из них соединены наибольшим числом ребер. Наилучшее слияние в данной вершине — это слияние этих двух кластеров.

Наилучшее разбиение: Наилучшее разбиение — это разбиение, при

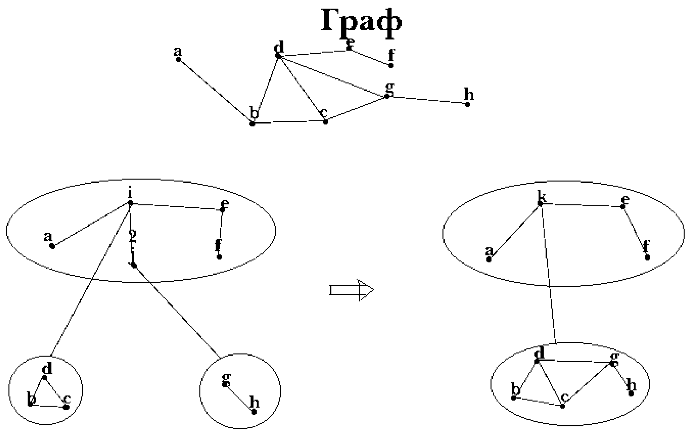

Рис. 6. Слияние двух деревьев.

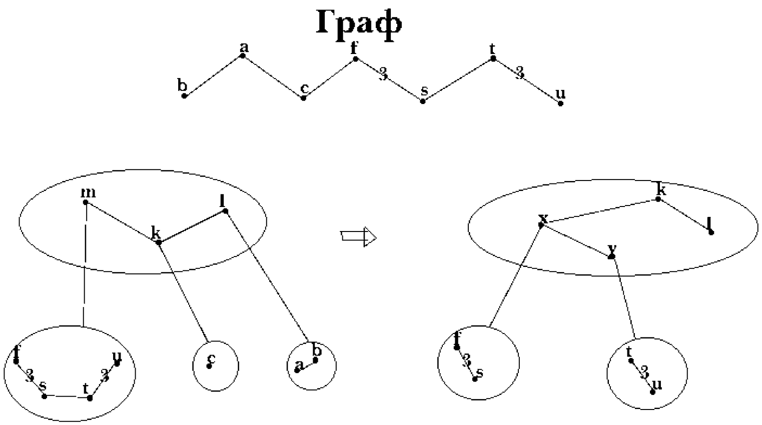

Рис. 7. Разбиение двух деревьев.

котором две получившиеся вершины имеют наименьшее число ребер, соединяющее их.

В этой главе были предложены три основные операции, которые мы будем совершать над деревом. Это операции:

1) добавление элемента в дерево;

2) наилучшее слияние;

3) наилучшее разбиение.

5. Формализация

До данного момента мы объясняли все «на пальцах», теперь мы имеем достаточно информации для более глубокого исследования проблемы.

Обозначения

D — множество всех деревьев кластеров.
G — множество всех графов.
K — множество всех кластеров.
D_g — дерево с элементами (документами), взятыми из графа g.
I — полный граф (тот, что мы хотим исследовать).
D_0 — множество деревьев, кластеры которых сильно соединены внутри и слабо соединены с другими кластерами. Будем называть их идеальными. Формальное определение идеального дерева дано на стр. 422. D_1 — множество деревьев, кластеры которых соединены внутри не хуже, чем соединены с другими кластерами. Будем называть их почти идеальными. Формальное определение почти идеального дерева дано на стр. 427.
Наша цель — построить идеальное или почти идеальное дерево, так как если кластеры сильно соединены внутри и слабо соединены с другими кластерами, то из предположения 2 следует, что это и есть смысловые объединения, которые мы стремимся выделить.

5.1. Определения

$Up(k) : K \to K$. «Родитель» кластера k.
$R(X_1, X_2) : K \times K \to N$. Расстояние между кластерами X_1 и X_2.
$R(X_1, X_2)$ — длина цепи, соединяющей X_1 и X_2 в дереве.
$Dis(d, k) : D \times K \to D \times K \times K$. Наилучшее разделение поддерева d с вершиной в k в исходном дереве. В итоге получаем новое дерево с двумя новыми кластерами.
$Dis_d(d, k) : D \times K \to D$. Взятие дерева из операции $Dis(d, k)$.
$Dis_k(d, k) : D \times K \to K$. Взятие любого кластера из двух кластеров, получившихся в результате применения операции $Dis(d, k)$.
Аналогичные обозначения (J_d, J_k, ...) для нижеследующих операций.

$J(d, k_1, k_2) : D \times K \times K \to D \times K$. k_1 и k_2 — кластеры, содержащиеся в одном кластере как подкластеры. Это операция соединения кластеров k_1 и k_2 в дереве d. В итоге получаем новое дерево с новым кластером.
$Join(d, k) : D \times K \to D \times K \times K$. Это операция наилучшего соединения в кластере k дерева d. В итоге получаем новое дерево с новым кластером.
$Join_k(d, k)$ — кластер, в котором произошло соединение.
$Join_{res}(d, k)$ — кластер, который получился в результате соединения.

Определим основные операции группировки дерева
$DJ(d, k) : D \times K \to D \times K$;
$DJ_d(d, k) \stackrel{\text{def}}{=} Dis_d(Join_d(d, k), Join_{res}(d, k))$;
$DJ_k(d, k) \stackrel{\text{def}}{=} Up(Dis_k(Join_d(d, k), Join_{res}(d, k)))$;
$JD(d, k) : D \times K \to D \times K$;
$JD_d(d, k) \stackrel{\text{def}}{=} Join_d(Dis_d(d, k), Up(Dis_k(d, k)))$;
$JD_k(d, k) \stackrel{\text{def}}{=} Join_k(Dis_d(d, k), Up(Dis_k(d, k)))$.
Впоследствии операции DJ и JD будем обозначать за S.
Определение принадлежности. $\forall x, y \in K$; $x \in y$, если кластер x находится в поддереве с вершиной в y.
Определение равенства деревьев. $d_1, d_2 \in D$. $d_1 = d_2$, если они равны, как деревья (мы не рассматриваем внутреннюю структуру кластеров).
$d_1 \equiv d_2$, если $d_1 = d_2$ и соответствующие кластеры равны между собой (подкластеры соединены одинаково).

5.2. Свойства операций $Join$ и Dis

Посмотрим, что происходит с расстоянием между кластерами при операциях Dis и $Join$.
$Join$**:**
Рассмотрим рис. 6.

$$\forall x \in i, \forall y \in j : R_{new}(x, y) = R(x, y) - 1.$$

Расстояния между другими кластерами не изменились.

Dis:
Рассмотрим рис. 7.

$$\forall v \in x, \forall w \in y : R_{new}(v, w) = R(v, w) + 1.$$

Расстояния между другими кластерами не изменились.

Утверждение 1. *Высота дерева не меняется при применении операций разделения и слияния.*

Доказательство. Из рисунков 6 и 7 видно, что расстояние между кластером, в котором происходила операция, и другими кластерами не изменилось, а значит и высота дерева не изменилась.

Следствие 1. *Высота дерева не меняется при применении операции группировки.*

Теорема 1. *$d_1, d_2 \in D$. Если графы, соответствующие деревьям d_1 и d_2, равны и $\forall x, y \in K \cap d_1 \cap d_2 : R_{d_1}(x, y) = 0 \Leftrightarrow R_{d_2}(x, y) = 0$, тогда $d_1 \equiv d_2$.*

Доказательство. Доказательство будем вести индукцией по высоте дерева (максимальному расстоянию от вершины дерева до кластеров дерева).
База. Если высота равна 0, то дерево состоит из одного листа, значит, так как графы равны, то и деревья равны (дерево состоит из единственного кластера, документы которого соединены как в графе).
1 Шаг. Пусть для деревьев высоты k утверждение верно, докажем его для дерева высоты $k + 1$. Рассмотрим лист дерева d_1, его документы $(a_1, a_2, \ldots, a_n)$ присутствуют в дереве $d_2(b_1, b_2, \ldots, b_n)$, так как графы одинаковы. $\forall i, j : R(a_i, a_j) = 0$ по определению, значит $\forall i, j : R(b_i, b_j) = 0$ (по условию). Это означает, что все они принадлежат одному документу. Получили, что документы из листа дерева d_1 полностью содержатся в одном листе дерева d_2, аналогично доказывается обратное, значит листья содержат одинаковые документы и так как графы деревьев d_1 и d_2 равны, то документы внутри листьев соединены одинаково. Это означает, что эти листы равны между собой. Если провести данное рассуждение для всех листьев,

то получим, что у данных деревьев листья одинаковы. Если листья одинаковы и графы одинаковы, то соединены эти листья одинаково (по принципу соединения кластеров). Составим граф, в котором вершины — листья наших деревьев; он будет одинаков для обоих деревьев. Теперь можно рассматривать листья как документы (за счет этого высота дерева уменьшится на 1), а полученный граф из листьев как исходный граф. Теперь высота дерева стала k, по предположению индукции эти деревья равны, а значит и исходные деревья равны.

6. Алгоритм работы робота

Теперь проанализируем сказанное с точки зрения нашей проблемы.

Пусть в идеальном дереве расстояние между кластерами равно 0. Значит нам надо добиться того, чтобы в нашем дереве расстояние тоже стало равным 0, и деревья окажутся равными. Расстояние увеличивается только при разделении и **только между** кластерами, в которых произошло разделение, значит **единственная** проблема, которая может возникнуть — это проблема неправильного разделения. Заметим также, что при слиянии расстояние уменьшается **только между** кластерами, которые мы сливаем и не затрагивает расстояний между ними и другими кластерами. Значит можно использовать операцию слияния для решения этой проблемы. Операция слияния увеличивает количество кластеров, поэтому разумно использовать ее вместе с операцией разделения.

На данном этапе работа алгоритма полностью определена.

Находим кластер, который содержит самые лучшие документы, и берем тот неисследованный документ, с которым связаны документы из этого кластера; его добавляем в дерево. Когда появляется большой кластер, мы его разделяем. На некоторых этапах работы (через некоторое время) проводим операцию группировки для особо важных кластеров.

Алгоритм описан, все этапы его работы объяснены. Теперь хотелось бы исследовать, к чему может привести операция группировки; приведет ли она к идеальному дереву или к чему-нибудь другому. Об этом пойдет речь в следующем разделе.

7. Анализ алгоритма

Теорема 2.

$$\forall d \in D_0, k \in K, s \in S : s_d(d, k) = d.$$

Доказательство. Для того, чтобы доказать данное утверждение надо рассмотреть два варианта:
1) $S = D'J'$;
2) $S = J'D'$.

Рассмотрим случай 1), случай 2) рассматривается аналогично. Операция J' применяется к кластерам k_1 и k_2, получается k. Так как $d \in D_0$, то кластеры k_1 и k_2 сильно соединены внутри и слабо между собой, значит после применения операции D' мы вновь получим кластеры k_1 и k_2 (по определению операции идеального разделения). Дерево не поменялось, значит $s_d(d, k) = d$.
Вывод: Операция группировки дерева не меняет идеального дерева.

Эту теорему можно использовать как формальное определение идеального дерева.

Определение. $D_0 \stackrel{\text{def}}{=} \{d \in D | \forall S, \forall k \in K : S_d(d, k) \approx d\}$ — множество идеальных деревьев.

Определение. $x, y \in D$. Будем говорить, что x похоже на y и писать $x \cong y$, если $\exists k_1, k_2 \in K; S_1, S_2 \in S : S_{1d}(x, k_1) = y, S_{2d}(y, k_2) = x$.

Легко понять, что это отношение обладает свойствами рефлексивности и симметричности, но оно не обладает свойством транзитивности. Введем новое отношение, в котором будут выполняться все три свойства.

Определение. $d_1, d_2 \in D$. Будем говорить, что d_1 эквивалентно d_2, и писать $d_1 \approx d_2$, если $\exists x_1, x_2, \ldots, x_n \in K : d_1 \cong x_1 \cong x_2 \cong \cdots \cong x_n \cong d_2$.

Это отношение обладает свойствами симметричности, рефлексивности и транзитивности, значит оно разбивает все деревья на классы эквивалентности. Теперь мы найдем инварианты относительно операции S, чтобы описать эти классы.

Определение. $d \in D, k \in K : F(d, k) \stackrel{\text{def}}{=}$ количество ребер в кластере k дерева d.

Утверждение 2. $\sum_{k \in d} F(d,k) =$ *количество ребер в графе* g; g *соответствует дереву* d.

Доказательство. Докажем, что ребру из графа соответствует ребро из дерева кластеров. Рассмотрим два документа x и y из графа g, соединенные ребром. Поставим ему в соответствие ребро из дерева. Найдем самое маленькое поддерево дерева d, содержащее документы x и y. В вершине этого поддерева есть два кластера k_1 и k_2: $x \in k_1$, $y \in k_2$. $k_1 \neq k_2$, так как это поддерево минимальное. k_1 и k_2 соеденены ребром по принципу соединения кластеров (так как $x \in k_1$, $y \in k_2$), и x соединен с y. Получили, что ребру из графа соответствует ребро из дерева.

Осталось доказать, что ребру из дерева кластеров соответствует ребро из графа. Рассмотрим ребро из дерева кластеров, оно соединяет кластеры x и y в кластере k. По принципу соединения кластеров $\exists k_1 \in x, k_2 \in y$, соединенные ребром, далее $\exists k_3 \in k_1, k_4 \in k_2$, соединенные ребром, $\ldots$, до тех пор, пока k_n и k_{n+1} не будут документами. Получили, что документы k_n и k_{n+1} соединены ребром, значит в графе есть ребро между k_n и k_{n+1}.

Мы получили взаимно однозначное соответствие между ребрами из графа и ребрами из дерева кластеров, значит их количество одинаково.

Посмотрим, что может произойти с числом ребер в кластере после совершения операции S.

Утверждение 3. $F(d, S_k(d,k)) \leqslant F(d,k)$. *Другими словами, после группировки количество ребер может только уменьшиться.*

Доказательство. Это утверждение следует из определения операций наилучшего слияния и наилучшего разбиения (см. стр. 416).

Введем классификацию операций группировки относительно данного дерева.

Определение. $d \in D; SF[d] \stackrel{\text{def}}{=} \{s \in S | s_d(d,k) \approx d\}$ — множество фиктивных операций над деревом d.

Определение. $d \in D; SN[d] \stackrel{\text{def}}{=} S \setminus SF[d]$ — множество нефиктивных операций над деревом d.

Определение. $d \in D; SLim[d] \stackrel{\text{def}}{=} \{s \in S | F(d, s_k(d,k)) \neq F(d,k)\}$ — множество приводящих операций над деревом d.

Определим предельное множество.

Определение. $M \stackrel{\text{def}}{=} \{d \in D | \forall S, \forall k \in K : F(d, S_k(d,k)) = F(d,k)\}$ — предельное множество. Другими словами, все операции группировки, совершаемые над деревом из предельного множества, являются неприводящими.

Определим действие операции группировки на всем дереве.

Определение. $d \in D$; $S(d)$ — любая приводящая операция группировки над деревом d.

Из определения следует, что операцию группировки нельзя применять к деревьям из предельного множества M.

Введем определение уровня. Это расстояние от кластера до вершины дерева.

Определение. $k \in K$; $level(k) \stackrel{\text{def}}{=} R(k, k')$, где k' — вершина дерева.

Определение.

$$d \in D;\ n \in \mathbb{N} : F_{level}(d,n) \stackrel{\text{def}}{=} \sum_{k \in K : level(k) = n} F(d,k).$$

Количество всех ребер на уровне n.

Докажем свойства приводящих операций группировки.

Лемма 1. *$d \in D$. Если $s \in SLim[d]$, тогда*

$$\begin{gathered}
F_{level}(0,d) = F_{level}(0, s_d(d,k)); \\
F_{level}(1,d) = F_{level}(1, s_d(d,k)); \cdots ; \\
F_{level}(level(k)-1, d) = F_{level}(level(k)-1, s_d(d,k)); \\
F_{level}(level(k), d) > F_{level}(level(k), s_d(d,k)); \\
F_{level}(level(k)+1, d) < F_{level}(level(k)+1, s_d(d,k)); \\
F_{level}(level(k)+2, d) = F_{level}(level(k)+2, s_d(d,k)); \cdots ; \\
F_{level}(n,d) = F_{level}(n, s_d(d,k)),
\end{gathered}$$

где n — максимальный уровень (высота дерева); k — кластер, в котором происходит группировка.

Доказательство. Так как операция группировки затрагивает только кластер и его подкластеры, то остальные кластеры не поменялись, а значит

$$\begin{gathered}F_{level}(0, d) = F_{level}(0, s_d(d, k));\\ F_{level}(1, d) = F_{level}(1, s_d(d, k)); \cdots ;\\ F_{level}(level(k) - 1, d) = F_{level}(level(k) - 1, s_d(d, k));\\ F_{level}(level(k) + 2, d) = F_{level}(level(k) + 2, s_d(d, k)); \cdots ;\\ F_{level}(n, d) = F_{level}(n, s_d(d, k)).\end{gathered}$$

По утверждению 2

$$\begin{aligned}&F_{level}(level(k), d) + F_{level}(level(k) + 1, d) =\\ &\qquad = F_{level}(level(k), s_d(d, k)) + F_{level}(level(k) + 1, s_d(d, k)).\end{aligned}$$

Остается доказать, что

$$\begin{gathered}F_{level}(level(k), d) > F_{level}(level(k), s_d(d, k));\\ \forall k' \neq k : level(k') = level(k).F(k', d) = F(k', s_d(d, k')),\end{gathered}$$

так как эти кластеры не затрагивались при операции группировки. Осталось рассмотреть $F(d, S_k(d, k))$ и $F(d, k)$. По определению $SLim[d]$

$$F(d, S_k(d, k)) \neq F(d, k),$$

а по утверждению 3

$$F(d, S_k(d, k)) \leqslant F(d, k),$$

значит

$$F(d, S_k(d, k)) < F(d, k).$$

Значит, по определению F_{level}

$$F_{level}(level(k), d) > F_{level}(level(k), s_d(d, k)).$$

Лемма 2 (Основная лемма).

$$\forall d \in D \ \forall S_1, S_2, \ldots, S_k \in SLim[d] : S_1 \circ S_2 \circ \cdots \circ S_k(d) \not\approx d.$$

Доказательство. Рассмотрим S_1, пусть k — кластер, в котором проходит группировка S_1, тогда по предыдущей лемме

$$\sum_{i=0,\ldots,level(k)} F_{level}(i, S_1(d)) < \sum_{i=0,\ldots,level(k)} F_{level}(i, d).$$

Если $S_1 \circ S_2 \circ \cdots \circ S_k(d) \approx d$, то

$$\sum_{i=0,\ldots,level(k)} F_{level}(i, S_1 \circ S_2 \circ \cdots \circ S_k(d)) = \sum_{i=0,\ldots,level(k)} F_{level}(i, d),$$

значит остается доказать, что

$$\forall k \sum_{i=0,\ldots,level(k)} F_{level}(i, S_1 \circ S_2 \circ \cdots \circ S_k(d)) < \sum_{i=0,\ldots,level(k)} F_{level}(i, d).$$

Докажем это утверждение по индукции.
База индукции. Рассуждение проведено выше
Шаг индукции. Пусть для i утверждение верно, докажем для $i+1$. Рассмотрим S_{i+1}. Пусть m — кластер, в котором проходит группировка S_{i+1}.
Если $level(m) = level(k)$, то по предыдущей лемме мы получаем соотношение для $i+1$.
Иначе по утверждению 2 и предыдущей лемме мы получаем, что

$$\sum_{i=0,\ldots,level(k)} F_{level}(i, S_1 \circ S_2 \circ \cdots \circ S_i(d)) = \\ = \sum_{i=0,\ldots,level(k)} F_{level}(i, S_1 \circ S_2 \circ \cdots \circ S_{i+1}(d)),$$

а значит утверждение верно для $i+1$. Доказательство по индукции проведено.

Замечание. Основную лемму можно доказать при более слабых условиях:

$$\forall d \in D, \text{ если } \exists i \in 0, \ldots, k : S_i \in SLim[d] : S_1 \circ S_2 \circ \cdots \circ S_k(d) \not\approx d.$$

План доказательства не меняется.

Впоследствии будем использовать это утверждение как Основную лемму.

Следствие 2. *$SLim[d] \subset SN[d]$.*

Теорема 3. *$\forall d \in D\ \exists k \in \mathbb{N} : S_d^k(d) \in M$. Любое дерево за конечное число операций группировки сводится к дереву из предельного множества.*

Доказательство. Рассмотрим дерево d, пусть его высота равна l. Тогда по Следствию 1 высота дерева $S(d)$ равна l. Докажем, что деревьев высоты l конечное число. Будем строить все такие деревья. Сначала выделим l-тый уровень из графа. Это можно сделать конечным числом способов, далее соединим получившиеся кластеры по принципу соединения кластеров. Теперь выделим $l-1$-ый уровень и т. д... На каждом этапе число деревьев было конечно и этапов конечное число, значит число построенных деревьев конечно. Из построения следует, что мы перебрали все деревья, значит число деревьев высоты l конечно. Если деревьев конечное число, значит число не эквивалентных деревьев конечно. Рассмотрим деревья $S(d), S^2(d), \ldots$ По Основной лемме они все не эквивалентны, а по доказанному не эквивалентных деревьев конечное число, значит эта последовательность закончится, а закончиться она может только тогда, когда применение операции S к очередному дереву будет невозможно, и значит по определению операции S это дерево из предельного множества.

Теорема 4. *Множества D_0 и M совпадают.*

Доказательство. Рассмотрим дерево из множества M и два подкластера, принадлежащие одному кластеру. При совершении операции JD к этим кластерам мы получаем эквивалентное дерево (по определению множества M). Значит число ребер, соединяющих получившиеся кластеры, такое же как и число ребер, соединяющих исходные кластеры. Значит эти кластеры внутри соединены не хуже, чем между собой. То есть это почти идеальное дерево. Обратное включение доказывается аналогично.

Эту теорему можно использовать как формальное определение идеального дерева.

Теорема показывает корректность нашего алгоритма, то есть, применяя операцию группировки, мы всегда за конечное время получим почти идеальное дерево.

Для упрощения объяснений до заключения следует ввести некоторые определения.

Введем упорядочивание по числу ребер в вершине дерева.

Определение. Будем говорить, что дерево d_1 меньше дерева d_2 и писать $d_1 < d_2$, если:
1) графы, соответствующие этим деревьям, одинаковы;
2) количество кластеров в вершине у обоих деревьев одинаково;
3) количество ребер в вершине первого дерева меньше количества ребер в вершине второго дерева.

В данном определении мы рассматриваем только вершину каждого дерева и не берем в рассмотрение все остальные кластеры. Таким образом, мы задали частичный порядок на множестве D.

Определение. $M_0^1 = \{d' \in D | \forall d \in D : d' \leqslant d$ или d не сравнимо с $d'\}$. Это множество, состоящее из минимальных элементов. Назовем его минимальным.

Теперь введем другое упорядочивание. Его назовем сильным упорядочиванием.

Определение. Будем говорить, что дерево d_1 меньше дерева d_2 и писать $d_1 <_s d_2$, если:
1) графы, соответствующие этим деревьям, одинаковы;
2) количество уровней одинаково;
3) количество кластеров на каждом уровне у обоих деревьев одинаково;
4) $\exists k \in 0, \dots, n$:

$$F_{level}(0, d_1) = F_{level}(0, d_2);$$
$$F_{level}(1, d_1) = F_{level}(1, d_2); \cdots ;$$
$$F_{level}(k-1, d_1) = F_{level}(k-1, d_2);$$
$$F_{level}(k, d_1) < F_{level}(k, d_2),$$

где n — высота дерева.

Определение. $M_0^\infty = \{d' \in D | \forall d \in D : d' \leqslant_s d$ или d не сравнимо с $d'\}$. Это множество, состоящее из минимальных элементов в смысле сильного упорядочивания. Назовем его абсолютно минимальным.

Сформулируем очевидное утверждение, следующее из написанных определений.

Утверждение 4. $M_0^\infty \subset M_0^1$ *и* $M_0^\infty \subset D_0 \subset D_1$.

8. Заключение

Рассмотрим, что мы уже сделали для решения поставленной проблемы и что еще предстоит сделать (см. рис. 8).

Рис. 8. Дерево разбора проблемы «Поискового робота».

Комментарии к рисунку:

- «+» — решенная проблема.
- «−» — нерешенная проблема.
- «+−» — частично решенная проблема.
- **Сравнение документов** — Чтобы робот сам ходил по сети без участия пользователя, он должен уметь сравнивать документы сам (какой из них лучше или хуже подходит данной тематике). Эта проблема решена, но не в рамках данной работы. Пользователю предлагается несколько документов на сравнение, а дальше робот рассортировывает остальные документы, причем этот алгоритм самообучающийся. Один из вариантов решения этой проблемы можно посмотреть в [9].

- **Выбор нового документа** — Если мы умеем сравнивать документы, то надо и уметь находить новые хорошие документы в сети, данная задача решена при помощи дерева кластеров.
- **Построение дерева кластеров** — Чтобы иметь дерево кластеров, надо уметь добавлять в него документы, сохранять маленькое количество кластеров в вершинах дерева и добиваться его приближения к идеальному дереву. Все эти аспекты были рассмотрены выше.
- **Выбор документа из дерева кластеров** — В дереве надо найти самый подходящий кластер и взять оттуда документ.
- **Добавление документа** — Добавление документа в дерево кластеров.
- **Применение операции группировки** — Надо выбирать особо важные кластеры и применять к ним операцию группировки, также надо выяснить, к какому дереву приводит применение этой операции.
- **Математический анализ операции группировки** — Задача остается открытой, так как мы знаем, что мы всегда попадем в множество D_1 (см. теорема 3), но хотелось бы попасть в множество D_0, так как оно немного лучше; и совсем маловероятно, что мы попадем в множество M_0^∞. Это множество не только идеально, но оно и самое лучшее из всех возможных.

Исходя из предложенного рисунка можно сделать вывод о том, что **поставленная проблема решена.**

Данный алгоритм может быть применен и к другой проблеме: в графе требуется найти наименьшее число ребер, выкинув которые, мы получим n не связных графов. Данная задача имеет полиномиальную сложность и может быть решена методом Форда и Фалкерсона [7, Глава 2.3].

Несколько видоизмененный алгоритм может быть применен к этой задаче.

Алгоритм: Мы имеем граф. Дерево создаем следующим образом: в начале создаем вершину и в ней n пустых кластеров, далее рассортировав на 5 частей вершины графа, мы добавляем их по одной в соответствующий кластер, используя наш алгоритм. В итоге получим дерево кластеров, в вершине которого ровно n кластеров. Применяя операцию группировки мы будем улучшать наше дерево и в конце концов придем к некоторому пределу (согласно теореме 3). В итоговом дереве, в его вершине мы получим разбиение вершин графа на n кластеров. Это разбиение и можно считать искомым.

На рис. 9 представлено дерево разбора данной проблемы.

Рис. 9. Дерево разбора проблемы «Разделение графа».

Комментарии к рисунку:

- **Проблема разбиения** — Проблема разбиения графа на два графа, при помощи выкидывания ребер.
- **Выбор начального дерева** — В вышенаписанном алгоритме начальное дерево строилось случайным разбиением графа на n частей, но этот алгоритм наверняка можно улучшить, например, если мы уже знаем одно неплохое разбиение.
- **Корректность работы** — Мы выяснили, что данный алгоритм приводит к дереву из множества M, но по условию задачи мы должны получить дерево из M_0^1, остается выяснить, попадем ли мы в множество M_0^1 или нет. В идеале надо попасть в множество M_0^∞, так как тогда мы получим наилучшее разбиение на всех уровнях дерева.
- **Время работы** — Это основная задача в проблеме разбиения. Целью данного алгоритма является нахождение разделения за наименьшение время, и поэтому надо оценить время его работы.

Из дерева разбора для данной задачи видно, что открытых вопросов остается все еще много, но очень многое уже сделано, и намечены основные пути решения других проблем.

Список литературы

[1] Hermans B. Intelligent software agents on the internet. Tilburg University, The Netherlands, the 9th of July 1996.

[2] Yahoo. http://www.yahoo.com.

[3] Maes P. Agents that Reduce Work and Information Overload. MIT Media Laboratory USA, 1997.

[4] Menczer F., Monge A. Scalable Web Search by Adaptive Online Agents: An InfoSpider Case Study. Management Science Department University of Iowa USA, 1997.

[5] Menczer F., Belew R. Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web. Technical report CS98–579. Computer Science and Engineering Department, University of California, San Diego. USA, 1998.

[6] Excite. http://www.excite.com.

[7] Ловас Л., Пламмер М. Прикладные задачи теории графов. М.: Мир, 1998.

[8] Невзоров О. А. Машинное обучение и задачи обработки естественного языка // Новости искусственного интеллекта — 1. М., 1998.

[9] Salton G. Automatic Text Processing: Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley, Readinh MA, 1989.