АЛГОРИТМ КРАУЛИНГА S.C.E.N.T.

С.Л. Сотник

Тех. директор Iveonik Systems, Днепродзержинск

В процессе деятельности компании Iveonik Systems, возникла задача извлечения текстовой информации из произвольного набора страниц сайтов. Данный процесс известен также под названием краулинга (crawling). Программный компонент, осуществляющий обход сайта, далее будет называть краулером (crawler). Краулер является весьма важной составляющей частью таких сложных и известных программных комплексов, как поисковые сервисы Google, Yahoo, Bing, Яндекс.

Одним из требований к краулеру является требование о поддержании информации о сайте в максимально релевантном состоянии. Это несложно для относительно небольших сайтов. Но реалии таковы, что зачастую краулеру не удается обойти сайт «в один присест». Это может происходить по разным причинам - небольшой канал у сайта, перегрузка сайта посетителями, огромные размеры сайта. Наконец, есть сайты, которые настолько «живые», что даже за несколько минут их содержимое (на некоторых страницах) может существенно измениться. Все это приводит к тому, что полученная при акте краулинга сайта информация оказывается неполной, либо неактуальной.

Рассмотрим задачу краулинга как задачу получения максимального количества информации при ограничении ресурсов. Интуитивно понятно, что краулер может закачать и обработать за определенный промежуток времени конечное количество информации. Это и есть тот самый ограниченный ресурс. Однако вопрос, как измерить практическую ценность добытой информации, является уже не столь интуитивным. Будем считать, что информационная ценность страницы зависит от количества символов в её текстовом представлении. Чем больше символов, тем более информативна страница. С другой стороны, для человека зачастую очень длинные страницы имеют ценность, не пропорциональную их длине. Попытаемся отразить это, используя логарифмическую меру:

|  |  |
| --- | --- |
|  | (1) |

Здесь ActualSize – размер актуального контента (который был в версии, сохраненной в индексе и имеется в существующем контенте), DeletedSize – суммарный размер контента, который отсутствует в текущей версии, но был в индексированной, либо наоборот. Суммарную же информационную емкость индекса мы можем оценить как:

|  |  |
| --- | --- |
|  | (2) |

Здесь Q(j) – суммарная стоимость (цена) индекса для сайта j. N(j) – множество документов сайта (в том числе, удаленных сейчас). Примем, что на проверку каждой страницы нам необходимо время:

|  |  |
| --- | --- |
|  | (3) |

Здесь A – коэффициент временных расходов, зависящих от размера документа (страницы), а B – накладные расходы на установление соединения и другие операции, не зависящие от размера.

Теперь мы готовы описать задачу краулинга, как задачу максимизации информационной стоимости индекса при ограниченных временных ресурсах:

|  |  |
| --- | --- |
|  | (4) |

Здесь мы также положили, что время между запусками краулера представляет собой константу. Т.е., расходы на индексацию сайта гораздо меньше, чем расходы на индексацию всего множества сайтов, обрабатываемых краулером.

Предлагаемый алгоритм оптимизации обхода (названный S.C.E.N.T.) немного навеян муравьиными алгоритмами и является самоорганизующимся.

Каждая страница наделяется свойством "запаха" (scent). Проверка страницы уменьшает запах в 2 раза. При этом, если "паучком"-краулером был собран "урожай" (информация), то запах страницы увеличивается на величину, пропорциональную "урожаю". На каждом шаге выбираются страницы, у которых "запах" был самым сильным. После каждого шага, каждой странице (не только проверенные) добавляется DeltaS единиц запаха. На псевдокоде это можно описать следующим образом:

void CrawlerIteration() {

List<Page> pages = GetPagesFromDB().SortByScent();

for(int i=0; i<PagesForProcessing; ++i) {

pages[i].Scent /= 2;

pages[i].Scent += CalcInfo(CheckPage(pages[i]));

}

foreach(Page page in pages) {

page.Scent += DeltaS;

}

}

double CalcInfo(int differenceInBytes) {

return Math.Log(page.Size + 1);

}

Основной вопрос, который сейчас исследуется - характер оценочной функции (CalcInfo) и значение константы DeltaS, от которой зависит характер обхода в случае, если сайт невозможно обойти за один "присест".

DeltaS=0 - Страницы, которые несколько раз оказались неизмененными, перестанут обходиться. DeltaS→∞ - Сайт будет обходиться равномерно, невзирая на то, что часть страниц изменяется чаще, часть - реже.

Оптимальное значение DeltaS, разумеется, находится где-то посредине. Эксперименты показывают, что при хорошо подобранных настройках, данный алгоритм дает Q(j) до 98% от максимально возможной (для сайтов со статической и динамической составляющей).