

Navigation link 1 www.example/WebPage1.ru	
contain	0.0125
example	0.0125
hyperlink	0.0125
page	0.25
static	0.0125
text	0.0125
Web	0.0125

Рис. 2

Подсчитывается частота W_{ij} встречаемости i -го термина из словаря в документе j , т.е. сколько раз i -й терм встретился в тексте, разделенное на общее число термов в документе. Поскольку большинство координат в терм-векторе равны нулю, то для его представления используют разряженную форму, в которой содержатся только значения координат не равных нулю. Для того чтобы различать к какой координате относится значение, их записывают парами <номер координаты, значение>, номер координаты соответствует номеру слова в словаре. Мы предлагаем использовать немного другую форму представления терм-вектора, в которой вместо номера координаты записано само слово. Например, "Navigation link 1" на рис. 2 содержит терм-вектор, соответствующий тексту страницы «Web page 1», изображенной на рис. 1.

Выбор функции релевантности (метрики) играет существенную роль для качества поиска. В данной работе мы рассматриваем одну из наиболее простых метрик. Она никак не учитывает порядок слов и их положение в документе. Так же она никак не учитывает синонимы. Использование более интеллектуальных способов вычисления схожести документов, например, учитывающих семантику [13,14], может привести к существенному увеличению качества поиска.

Как уже говорилось ранее, мы предлагаем строить децентрализованную структуру на уровне данных. Эта структура может рассматриваться как P2P сеть, в которой каждый информационный объект является узлом сети со своей собственной таблицей маршрутизации. Применительно к Web-

страницам, таблице маршрутизации будет соответствовать дополнительный список ссылок на другие страницы. Предполагается, что он будет размещаться внутри страницы, но будет скрыт от обычного пользователя и будет использоваться, только поисковыми алгоритмами.

Запись в таблице маршрутизации мы предлагаем называть навигационной ссылкой (navigation link). Каждая навигационная ссылка, указывающая на страницу x , включает в себя URL адрес и терм вектор d_x соответствующий этой странице (см. рис.2). Стоит отметить, что в навигационную ссылку включать терм вектор страницы, на которую она ссылается, не обязательно. Это лишь способ сокращения времени поиска и уменьшения трафика. То есть возможен обмен между объемом дополнительной информации, которая встраивается в каждую страницу и скоростью поиска.

Совокупность терм-вектора, построенного для данной страницы, и таблицы маршрутизации мы предлагаем называть «навигационный блок». Обратим внимание на то, что навигационный блок необязательно помещать в Web-страницу. Он может находиться в базе сервера, осуществляющего хостинг данной страницы, но при этом должен существовать механизм его получения по запросу. Для этого можно расширить HTTP протокол командой «GET_NAVIGATION_BLOCK».

Пример организации Web-страниц в структуру изображен на рисунке 1. Каждая страница содержит навигационный блок, выделенный на рисунке серым цветом. Стоит заметить, что в отличие от ориентированного графа Web, мы строим не ориентированный граф.

Для того чтобы реализовать данный подход в рамках Web, необходим механизм на уровне Web серверов, поддерживающий для каждой Web-страницы актуальный навигационный блок. Один из возможных путей реализации такого механизма - это реализация дополнительного модуля для одного из популярных Web-серверов. Он может выступать в

```

K-NNSearch(object q, integer: m, k)
1 TreeSet [object] tempRes, candidates, visitedSet, result
2 for (i ← 0; i < m; i++) do:
3   put random entry point in candidates
4   tempRes ← null
5   repeat:
6     get element c closest from candidates to q
7     remove q from candidates
8     //check stop condition:
9     if c is further than k-th element from res
10      than break repeat
11    //update list of candidates:
12    routingTable = c.getRoutingTable()
13    for every element e from routingTable do:
14      if e is not in visitedSet then
15        put e to visitedSet, candidates, tempRes
16
17    end repeat
18    //aggregate the results:
19    add objects from tempRes to result
20 end for
21 return best k elements from result

```

Рис. 3. Алгоритм поиска k ближайших элементов к запросу q , использующий m попыток.

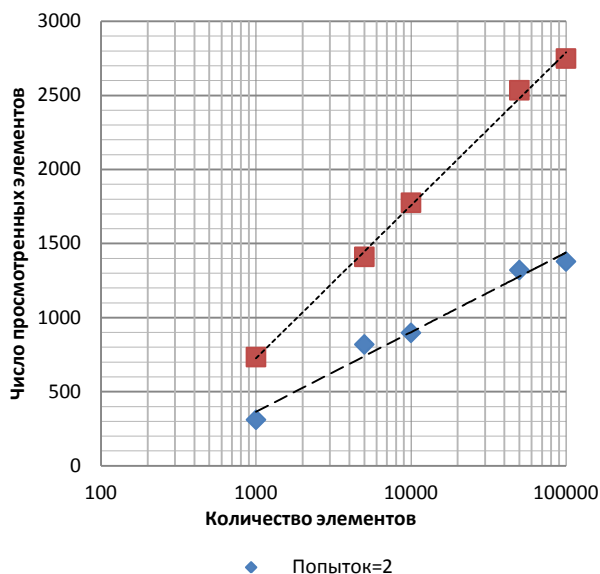


Рис 4. Зависимость числа просмотренных элементов от размера структуры

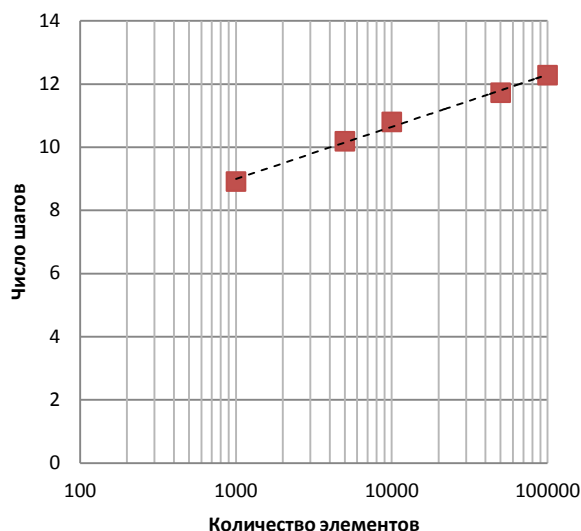


Рис 5. Зависимость среднего числа шагов пройденных поисковым алгоритмом за одну попытку поиска.

качестве прослойки между сервером и пользователем, строя терм-вектора для новых страниц и добавляя их в общую сеть.

Чтобы пользователь мог производить поиск в такой системе, необходимо программное обеспечение, которое по запросу строит терм-вектор и последовательно переходит от одной страницы к другой (см. рис. 3) в направлении увеличения релевантности до запроса, т.е. в направлении увеличения косинуса угла между терм-вектором запроса и вектором рассматриваемой страницы.

Такое программное обеспечение может работать либо на стороне пользователя, и тогда его стоит реализовывать в виде дополнения (плагина) к браузеру, либо оно может быть реализовано в виде внешнего поискового сервиса, развернутого на сторонних серверах. В любом случае, поиск может быть инициирован любым агентом, знающим адрес хотя бы одной страницы участвующей в сети.

Алгоритм поиска k ближайших элементов к запросу q , использующий m попыток поиска, приведен на рис. 3. В качестве запроса в данном случае выступает терм-вектор. Изначально множество кандидатов содержит множество заранее известных страниц. 12-я строчка соответствует получению таблицы маршрутизации для страницы c и эквивалентна одному шагу алгоритма.

4. Результаты моделирования

Чтобы удостовериться, что используя в качестве структуры данных алгоритм MSW, релевантная страница может быть найдена с высокой вероятностью, и при этом число страниц, к которым

произошло обращение, мало по сравнению с общим количеством страниц, мы произвели следующие вычислительные эксперименты.

В качестве тестового набора данных мы использовали коллекцию WikipediaSparse, состоящую из 100,000 векторов, извлеченных из страниц Википедии. Количество уникальных слов в словаре, соответствующих числу координат векторов, составляет 100000.

Собирались структура из 1000, 5000, 10000, 50000 и 100000 элементов. Параметр NN, отвечающий за количество создаваемых ссылок при добавлении нового элемента в структуру, выставлялся равным 20. Для поиска 5 релевантных объектов к запросу использовался алгоритм K-NNSearch с параметрами числа попыток поиска $m=2$, $m=4$ и $k=5$. Замерялось количество вычисленных расстояний и точность поиска (recall), вычисляемая как отношение между количеством правильно найденных элементов и числа запрашиваемых k т.е. в данном случае $k=5$. Зависимость числа вычисленных расстояний от запроса до страниц можно увидеть на рис. 4. Зависимость среднего числа шагов, сделанных поисковым алгоритмом, изображена на рис. 5. Из обоих графиков можно сделать вывод о логарифмической зависимости сложности поиска. Однако с ростом количества элементов в структуре точность поиска незначительно падает (см. рис. 6), но её можно компенсировать путем увеличения либо числа NN, либо увеличением числа попыток поиска (см. рис 7). Эта зависимость требует более детального изучения.

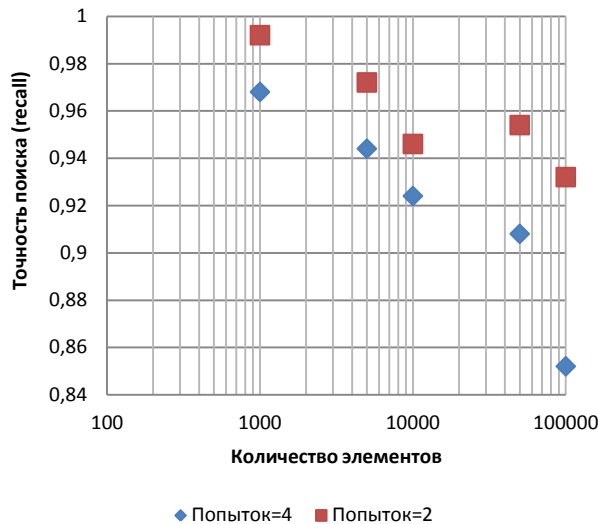


Рис. 6. Зависимость точности поиска от количества элементов в структуре

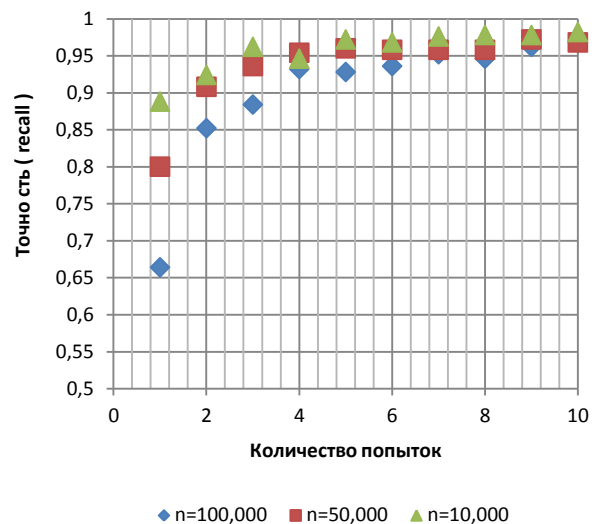


Рис. 7. Зависимость точности поиска от количества попыток поиска

5. Заключение и направление исследований

В этой работе мы предложили подход к организации информации на уровне данных в децентрализованную структуру данных. Описали возможность применения данного подхода для организации полнотекстового поиска в Web без использования центрального механизма индексирования, путем организации информации на уровне данных в децентрализованную структуру. Вычислительные эксперименты продемонстрировали, что несмотря на высокую размерность данных и малое число записей в таблице маршрутов, поиск с высокой вероятностью за логарифмическое число действий от общего размера структуры возвращает к максимально релевантных объектов к запросу. Таким образом, дополнительные издержки, которые влечет данный подход – малы.

В число дальнейших исследований входит конструирование новых функций ранжирования, разработка адаптивного алгоритма добавления, не требующего параметра числа ближайших соседей и поддерживающего заданную точность поиска в структуре, теоретический анализ свойств алгоритма MSW.

Также планируется разработка модуля к одному из популярных Web-серверов (Apache HTTP Server, JBoss), позволяющего поддерживать и автоматически строить для Web-страницы актуальную таблицу маршрутов.

6. Благодарности

Работа выполнена при поддержке Лаборатории алгоритмов и технологий анализа сетевых структур НИУ ВШЭ, грант правительства РФ дог. 11.G34.31.0057

Список литературы

- [1] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., & Raghavan, S. (2001). Searching the web. *ACM Transactions on Internet Technology (TOIT)*, 1(1), 2-43.
- [2] Malkov, Y., Ponomarenko, A., Logvinov, A., & Krylov, V. (2013). Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*.
- [3] Пономаренко А. А., Мальков Ю. А., Логвинов А. А., Крылов В. В. Структура со свойствами тесного мира для решения задачи поиска ближайшего соседа в метрическом пространстве // Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 5. С. 409-415.
- [4] Gnutella. <http://www.gnutella.com/>
- [5] Ritter, J. (2001). Why gnutella can't scale. no, really.
- [6] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware*, 2001.
- [7] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *SIGCOMM*, 2001.
- [8] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. UC Berkeley, Technical Report UCB/CSD-01-1141, 2001.
- [9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *SIGCOMM*, 2001.
- [10] Wong B., Vigfússon Y., Sireer E. G. Hyperspaces for object clustering and approximate matching in peer-to-peer overlays // Proceedings of the 11th USENIX workshop on

- Hot topics in operating systems. – USENIX Association, 2007. – C. 8.
- [11] Salton G., Wong A., Yang C. S. A vector space model for automatic indexing //Communications of the ACM. – 1975. – T. 18. – №. 11. – C. 613-620.
- [12] Singhal A. Modern information retrieval: A brief overview //IEEE Data Eng. Bull. – 2001. – T. 24. – №. 4. – C. 35-43.
- [13] Jiang J. J., Conrath D. W. Semantic similarity based on corpus statistics and lexical taxonomy //arXiv preprint [cmp-lg/9709008](https://arxiv.org/abs/cmp-lg/9709008). – 1997.
- [14] Pirró G., Seco N. Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content //On the Move to Meaningful Internet Systems: OTM 2008. – Springer Berlin Heidelberg, 2008. – C. 1271-1288.
- [15] Ponomarenko A. et al. Approximate Nearest Neighbor Search Small World Approach.
- [16] Beaumont O., Kermarrec A. M., Rivière É. Peer to peer multidimensional overlays: Approximating complex structures //Principles of Distributed Systems. – Springer Berlin Heidelberg, 2007. – C. 315-328.
- [17] Beaumont O. et al. VoroNet: A scalable object network based on Voronoi tessellations. – 2006.
- [18] Datar M. et al. Locality-sensitive hashing scheme based on p-stable distributions //Proceedings of the twentieth annual symposium on Computational geometry. – ACM, 2004. – C. 253-262.
- [19] Boytsov L., Naidan B. Engineering Efficient and Effective Non-metric Space Library //Similarity Search and Applications. – Springer Berlin Heidelberg, 2013. – C. 280-293.
- [20] Figueroa K., Navarro G., Chávez E. Metric spaces library //Online <http://www.sisap.org>. – 2007.
- [21] Jon . "Complex networks and decentralized search algorithms." Proceedings of the International Congress of Mathematicians (ICM). Vol. 3. 2006.