Succinct Data Structures in Information Retrieval: Theory and Practice

Simon Gog
Institute of Theoretical Informatics, Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe, Germany
gog@kit.edu

Rossano Venturini
Department of Computer Science,University of Pisa
Largo Bruno Pontecorvo 3
I-56127 Pisa, Italy
rossano.venturini@unipi.it

ABSTRACT

Succinct data structures are used today in many information retrieval applications, e.g., posting lists representation, language model representation, indexing (social) graphs, query auto-completion, document retrieval and indexing dictionary of strings, just to mention the most recent ones. These new kind of data structures mimic the operations of their classical counterparts within a comparable time complexity but require much less space. With the availability of several libraries for basic succinct structures – like SDSL, Succinct, Facebook's Folly, and Sux – it is relatively easy to directly profit from advances in this field.

In this tutorial we will introduce this field of research by presenting the most important succinct data structures to represent set of integers, set of points, trees, graphs and strings together with their most important applications to Information Retrieval problems. The introduction of the succinct data structures will be sustained with a practical session with programming handouts to solve. This will allow the attendees to directly experiment with implementations of these solutions on real datasets and understand the potential benefits they can bring on their own projects.

1. MOTIVATION

The current growth and availability of massive amounts of data gathered and processed by applications – Web search engines, textual and biological databases, just to cite a few – has changed the algorithmic requirements of basic processing and mining tools and provide ample motivation for a great deal of new theoretical and practical research on algorithms and data structures.

Not surprisingly, the last decades have seen a massive research in the field of the so-called succinct and compressed data structures. These new kind of data structures mimic the operations of their classical counterparts within a comparable time complexity but requiring much less space. These solutions usually resort to a careful combination of ideas born both in algorithmic and data compression fields. As a concrete example consider trees which are probably among the most used data structure in practice. Everybody is familiar with the classical way to represent the structure of a tree in memory: each node is connected to its children by means of pointers. This representation immediately allows simple navigational operations on the tree. One can also augment the structure with auxiliary information in order to answer queries such as: Who is the parent of a given node? How many nodes are in a given subtree? Who is the lowest common ancestor of a given pair of nodes? and so on. The main drawback of this solution is its space occupancy which can be very optimistically estimated in 32 bits per node. Not surprisingly the problem of repesenting trees space efficiently has been one of the first problems to be addressed. The current best succinct tree representation requires *provably* no more than 2 bits per node and is able to answer in constant time a large set of queries [9]. The practical consequence of the use of this succinct data structure is immediate: applications can store in main memory trees whose sizes would be prohibitive with the classical representation.

These efficient tree representations are just an example of succinct data structures and, even if the research field is relatively new, a large fraction of classical data structures have their succinct counterpart [1, 2, 6, 10, 14, 25, 31]. Most of these results are of practical interest and several implementations exist, for example we mention SDSL[1], Succinct[2], Facebook's Folly[3], and Sux[4].

Several authors are successfully applying the ideas at the basis of the succinct data structures to solve problems in other fields of research. Information retrieval community profits a lot from these data structures as there exist several applications in which they play a central role, e.g., posting lists representation [22, 28, 29, 35], language model representation [34], indexing (social) graphs [8], query autocompletion [21], document retrieval [15, 16, 19, 26, 30] and indexing dictionary of strings [11, 18, 36], just to mention the most recent ones.

The tutorial will introduce this field of research by presenting the most important succinct data structures to represent set of integers, set of points, trees, graphs and strings together with their most important applications to Informa-

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2914802

[1] https://github.com/simongog/sdsl-lite
[2] https://github.com/ot/succinct
[3] https://github.com/facebook/folly
[4] http://sux.di.unimi.it

tion Retrieval problems. The introduction of the succinct data structures will be sustained with a practical session with programming handouts to solve. This will allow the attendees to directly experiment with implementations of these solutions on real datasets and understand the potential benefits they can bring on their own projects.

2. OBJECTIVES AND RELEVANCE TO THE IR COMMUNITY

Indexing is the key to efficient query processing in the vast majority of IR applications. The use of inverted indexes in web search engines is probably one of the most prominent examples in Information Retrieval.

While index compression is a relatively old and established method in IR and covered by all relevant text books the new data structures which combine compression and efficient execution of operations – like random access – are a relatively new field of which many IR researchers are not yet aware of. However, this field of research is highly relevant for IR as, for example, we have already seen that techniques from the succinct data structure field are able to improve inverted indexes systematically [29, 35]. Succinct data structures often *extend the functionality* of traditional structures or reduce query time. For example, while inverted indexes are restricted in handling phrase queries, new document retrieval frameworks based on succinct structures allow to efficiently answer top-k queries for phrases of arbitrary length [19], and it is also possible to determine the document frequency of any phrase in constant time by just spending at most 2 bits per element in the collection [32]. Unfortunately, succinct structures per se are not easy to implement from scratch and especially the implementation of the latter two results is very complex. However, as more and more implementations of basic succinct structures become available via libraries, it is possible to combine these pieces to obtain more advanced solutions. While other fields like Bioinformatics already actively profit from using succinct structures after engineering the theoretical proposals to their applications, e.g., the FM-Index is used in DNA read mapping, the IR community does not yet fully exploit the new possibilities of succinct structures. With this tutorial we do not only inform the audience about succinct data structures but also point out engineering challenges and encourage the participants to use the potential of these solutions in IR. Therefore the objectives of this tutorials are (1) to introduce the audience to the world of succinct data structures, (2) to teach the basic techniques of succinct structures, (3) to present the practical impact of the structures to selected applications in IR, and (4) to show the participants how they can implement their own succinct structures on top of existing basic structures provided by the succinct data structure library.

3. DETAILED SCHEDULE

The tutorial is subdivided in two parts. The goal of the first part is to describe the most important succinct data structures. We will present the theoretical achievements in this field of research by focusing our attention to the ones that have demonstrated their relevance in practice. The introduction of each of these solutions will be sustained by presenting a selection of Information Retrieval problems in which they have an immediate application.

The goal of the second part is to present ready-to-use implementations of these data structures. This is done by introducing the Succinct Data Structure Library (SDSL).This library covers all the data structures presented in the previous part and has one of the presenters as its creator and one of its main developers. The speakers will also prepare a set of programming handouts which will allow the attendees to experiment with the learned concepts. The ultimate objective of the tutorial is to demonstrate to the attendees the potential benefit that succinct data structures can bring in their applications at a tiny implementation cost.

Schedule of the tutorial.

- **Part I**
  - Introduction to succinct data structures;
  - Basic operations on binary vectors [25, 31];
  - Elias-Fano representation [4, 5, 8, 27, 28, 29, 35];
  - Succinct representations of trees and range minimum/maximum queries [3, 9, 12, 33];
  - Wavelet trees [2, 13, 17, 24];
  - Compressed Full-Text indexes [7, 25];
  - Structures for IR [8, 11, 14, 20, 21, 23].
- **Part II**
  - Overview about SDSL components and concepts;
  - Detailed presentation of components via small code examples;
  - Exploration of practical performance of structures;
  - Presentation of benchmarking facilities;
  - Programming handouts.

4. ACKNOWLEDGEMENT

This tutorial was partially supported by the EU H2020 Program under the scheme *INFRAIA-1-2014-2015: Research Infrastructures* grant agreement #654024 *SoBigData: Social Mining & Big Data Ecosystem.*

References

[1] J. Barbay. Succinct and compressed data structures for permutations and integer functions. In *Encyclopedia of Algorithms*. 2015.

[2] J. Barbay and J. I. Munro. Succinct encoding of permutations: Applications to text indexing. In *Encyclopedia of Algorithms*. 2008.

[3] D. Benoit, E. Demaine, J. I. Munro, R. Raman, V.Raman, and S. S. Rao. Representing trees of higher degree. *Algorithmica*, 43(4):275–292, 2005.

[4] P. Elias. Efficient storage and retrieval by content and address of static files. *Journal of the ACM*, 21:246–260, 1974.

[5] R. M. Fano. On the number of bits required to implement anassociative memory. *Memorandum 61, Computer Structures Group, Project MAC*, 1971.

[6] P. Ferragina, R. González, G. Navarro, and R. Venturini. Compressed text indexes: From theory to practice. *ACM Journal of Experimental Algorithmics*, 13, 2008.

[7] P. Ferragina and G. Manzini. Indexing compressed text. *Journal of the ACM*, 52(4):552–581, 2005.

[8] P. Ferragina, F. Piccinno, and R. Venturini. Compressed indexes for string-searching in labeled graphs. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages –, 2015.

[9] P. Ferragina and S. S. Rao. Tree compression and indexing. In *Encyclopedia of Algorithms*. 2008.

[10] P. Ferragina and R. Venturini. Indexing compressed text. In *Encyclopedia of Database Systems*, pages 1442–1448. 2009.

[11] P. Ferragina and R. Venturini. The compressed permuterm index. *ACM Transactions on Algorithms*, 7(1):10, 2010.

[12] J. Fischer and V. Heun. Space-efficient preprocessing schemes for range minimum queries on static arrays. *SIAM Journal on Computing*, 40(2):465–492, 2011.

[13] L. Foschini, R. Grossi, A. Gupta, and J. S. Vitter. When indexing equals compression: Experiments with compressing suffix arrays and applications. *ACM Transactions on Algorithms*, 2(4):611–639, 2006.

[14] S. Gog, T. Beller, A. Moffat, and M. Petri. From theory to practice: Plug and play with succinct data structures. In *Proceedings of the 13th International Symposium Experimental Algorithms (SEA)*, pages 326–337, 2014.

[15] S. Gog and G. Navarro. Improved single-term top-k document retrieval. In *Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments, ALENEX*, pages 24–32, 2015.

[16] S. Gog and M. Petri. Compact indexes for flexible top-k retrieval. In *Combinatorial Pattern Matching - 26th Annual Symposium, CPM*, pages 207–218, 2015.

[17] R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 841–850, 2003.

[18] R. Grossi and G. Ottaviano. Fast compressed tries through path decompositions. *ACM Journal of Experimental Algorithmics*, 19(1), 2014.

[19] W. Hon, R. Shah, and J. S. Vitter. Space-efficient framework for top-k string retrieval problems. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 713–722, 2009.

[20] W. Hon, R. Shah, and J. S. Vitter. Space-efficient framework for top-k string retrieval problems. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 713–722, 2009.

[21] B. P. Hsu and G. Ottaviano. Space-efficient data structures for top-k completion. In *Proceedings of the 22nd International World Wide Web Conference (WWW)*, pages 583–594, 2013.

[22] R. Konow, G. Navarro, C. L. A. Clarke, and A. López-Ortiz. Faster and smaller inverted indices with treaps. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 193–202, 2013.

[23] G. Navarro. Spaces, trees and colors: The algorithmic landscape of document retrieval on sequences. *ACM Computing Surveys*, 46(4):article 52, 2014. 47 pages.

[24] G. Navarro. Wavelet trees for all. *Journal Discrete Algorithms*, 25:2–20, 2014.

[25] G. Navarro and V. Mäkinen. Compressed full text indexes. *ACM Computing Surveys*, 39(1), 2007.

[26] G. Navarro and Y. Nekrich. Top-k document retrieval in optimal time and linear space. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1066–1077, 2012.

[27] D. Okanohara and K. Sadakane. Practical entropy-compressed rank/select dictionary. In *Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX*, 2007.

[28] G. Ottaviano, N. Tonellotto, and R. Venturini. Optimal space-time tradeoffs for inverted indexes. In *Proceedings of the 8th Annual International ACM Conference on Web Search and Data Mining (WSDM)*, pages –, 2015.

[29] G. Ottaviano and R. Venturini. Partitioned elias-fano indexes. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 273–282, 2014.

[30] M. Patil, S. V. Thankachan, R. Shah, W. Hon, J. S. Vitter, and S. Chandrasekaran. Inverted indexes for phrases and strings. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*, pages 555–564, 2011.

[31] N. Rahman and R. Raman. Rank and select operations on binary strings. In *Encyclopedia of Algorithms*. 2008.

[32] K. Sadakane. Succinct data structures for flexible text retrieval systems. *J. Discrete Algorithms*, 5(1):12–22, 2007.

[33] K. Sadakane and G. Navarro. Fully-functional succinct trees. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 134–149, 2010.

[34] E. Shareghi, M. Petri, G. Haffari, and T. Cohn. Compact, efficient and unlimited capacity: Language modeling with compressed suffix trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2409–2418, 2015.

[35] S. Vigna. Quasi-succinct indices. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 83–92, 2013.

[36] S. Yata. Marisa trie, https://code.google.com/archive/p/marisa-trie/. 2011.