ABSTRACT

String kernels which compare the set of all common substrings between two given strings have recently been proposed by Vishwanathan & Smola (2004). Surprisingly, these kernels can be computed in linear time and linear space using annotated suffix trees. Even though, in theory, the suffix tree based algorithm requires $O(n)$ space for an $n$ length string, in practice at least $40n$ bytes are required -- $20n$ bytes for storing the suffix tree, and an additional $20n$ bytes for the annotation. This large memory requirement coupled with poor locality of memory access, inherent due to the use of suffix trees, means that the performance of the suffix tree based algorithm deteriorates on large strings. In this paper, we describe a new linear time yet space efficient and scalable algorithm for computing string kernels, based on suffix arrays. Our algorithm is a) faster and easier to implement, b) on the average requires only $19n$ bytes of storage, and c) exhibits strong locality of memory access. We show that our algorithm can be extended to perform linear time prediction on a test string, and present experiments to validate our claims.

REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.


applications. Algorithmica, 12(4/5), 327--344.


↑ Collaborative Colleagues:

- Choon Hui Teo: S. V. N. Vishwanathan
- S. V. N. Vishwanathan: Karsten M. Borgwardt, Alex J. Smola, Terry Caelli, Choon Hui Teo
- Hans-Peter Kriegel: Li Cheng, Shaojun Wang
- Kevin P. Murphy: Cheng Soon Ong, Stefan Schönauer