Approximate search in intrusion detection using dynamic programming

1. Background

In misuse detection systems it is necessary to have a rule/signature for every variant of a single attack, no matter how small the differences between these rules might be. This consumes memory resources in the rules database and slows down the search process during data processing. Therefore, it is of interest to replace many variants of essentially the same attack signature with a single one and employ approximate search instead of the exact search in misuse detection. There exist several methods of approximate search, of which dynamic programming-based methods have been extensively used in general computing practice. These algorithms are based on the algorithm for computation of so-called edit distance, which can be modified to perform both exact and approximate search.

2. The task for the student

1. Implement the classical approximate search algorithm using dynamic programming approach allowing up to k errors in the search string using your favorite programming language (any language can be used – c/c++, Java, Python, c#, Pascal, Matlab, etc.) Read the description of the algorithm in the reference [1], pages 146-148 (these pages can be found in Blackboard).
2. Write a short (up to 3 A4 pages) report about the approximate search algorithm using the classical dynamic programming approach. From this report, it should be clear that you have understood its operation – provide examples etc. Explain how we can implement this algorithm without keeping the whole dynamic programming matrix in the memory (see [1]).
3. Advanced: Explain in which case it is of interest to keep the whole dynamic programming matrix in memory, even though we know that we can implement that algorithm without doing it.
4. Advanced: Read the reference [2] (also available in Blackboard). It is the original Myers' paper from 1999 describing a bit-vector realization of the dynamic programming-based approximate search algorithm. Implement the basic algorithm described in [2] (Section 3, pages 399-406) in the same programming language that you used in 1. Write a report about this algorithm such that it should be clear that you have understood its operation – provide examples etc. (up to 5 A4 pages).

Remark: all the reports can be put in a single document. The allowed format is pdf, odt, or doc.

3. Handing in

This is individual work. Put everything you produce in a single ZIP file yourstudentnumber.zip. The handing in technology is Inspera. The deadline is December 1st, 2020 at 12.00 Oslo time.

4. References

[1] Navarro G. and Raffinot M., Flexible Pattern Matching in Strings, Cambridge University Press, 2002.

[2] Myers G., A fast bit-vector algorithm for approximate string matching based on dynamic programming, Journal of the ACM, Vol. 46, No. 3, May 1999, pp. 395-415.