Affix-based Metadata Search for Self-describing Data Formats

CS5352 Course Project, Spring 2021

[This project accepts two students.]

**Contact**

Wei Zhang (X-Spirit.zhang @ttu.edu)

**Description**

MIQS is a Metadata Indexing and Querying Service built for self-describing data formats [1]. MIQS addresses metadata querying for string metadata attributes and numeric metadata attributes by building index for these attributes. However, in MIQS, we have not implemented:

1. the affix-based keyword search for string metadata attributes.
2. the range query against numeric attribute values.

In this project, you are asked to implement:

1. Affix-based metadata search on attribute names. For example:
   1. prefix queries against attribute names like “list the IDs of all objects featuring attribute name starting with ‘EXPID-’ ”,
   2. suffix queries against attribute names like “list the IDs of all objects featuring attribute name like ‘\*-01’ ”
   3. infix queries against attribute names like “list the IDs of all objects featuring attribute name containing ‘-ID-’ ”.
2. Affix-based metadata search on string attribute values. For example:
   1. prefix queries against string attribute values like “list the IDs of all objects with ‘name’ attribute matching ‘abc\*’ ”,
   2. suffix queries against string attribute values like “list the IDs of all objects with ‘name’ attribute matching ‘\*.abc’ ”
   3. infix queries against string attribute values like “list the IDs of all objects with name attribute matching ‘\*bcc\*’ ”.
3. Range query against numeric attribute values, such as “list the IDs of all objects featuring attribute ‘EXPID’ whose value is in between the range of [2, 32)”
   1. Note that the range can be inclusive or exclusive on both sides. Also, considering the case where left or right boundary is negative infinity or positive infinity.
4. Affix-based metadata search on both attribute name and string attribute values. For example:
   1. “list the IDs of all objects featuring attribute names matching ‘\*abc’ and values matching ‘\*bcc\*’ ”, etc. Hint: for implementing this function, you may jointly invoke the two functions designed for 1 and 2 together.
   2. “list the IDs of all objects featuring attribute names matching ‘\*abc’ and values in between the range of [2, +INF)

For implementing efficient infix search, you need to construct suffix tree using our adaptive radix tree implementation. You may refer to <https://www.geeksforgeeks.org/pattern-searching-using-suffix-tree/> to get a sense of what a suffix tree really is.

Your final deliverable should include the following:

1. A functional code base: you can take the in-memory index code out of our MIQS code base and build your own code base so that you will not get bothered by other features of our MIQS implementation. Make sure your code base includes all the features required, all the necessary documentation guiding people to compile, test and run your program. (35%)
2. Detailed the test cases: consider various situations that you should validate for you final implementation. (15%)
3. Make sure your program prints out the performance metrics, such as: average time/throughput for indexing, average time/throughput for querying. You need to plot the performance metrics in your final project report. (25%)
4. A detailed project report, including introduction talking about the overview of the project, background talking about necessary background and motivation of the project, methodology section talking about the design of your implementation, evaluation section talking about the performance metrics and other metrics you’ve ever tested and collected, conclusion section recalling what you have done and how meaningful it is in this project. (25%)

**Requirements**

* Proficient in C programming
* Knowledge of indexing data structures

**References**

[1] W. Zhang, S. Byna, H. Tang, B. Williams and Y. Chen. MIQS: Metadata Indexing and Querying Service for Self-describing File Formats. Accepted to appear in The Proceedings of The 31st ACM/IEEE Supercomputing Conference (SC’19), Denver, CO, 2019. (first-around acceptance rate: 72/344=21%, another 15 papers being asked for major revisions per SC'19)

[2] Leis, V., Kemper, A., & Neumann, T. (2013, April). The adaptive radix tree: ARTful indexing for main-memory databases. In 2013 IEEE 29th International Conference on Data Engineering (ICDE) (pp. 38-49). IEEE.

[3] Suffix Tree: <https://www.geeksforgeeks.org/pattern-searching-using-suffix-tree/>

[4] MIQS software release: <https://bitbucket.org/berkeleylab/miqs/src/ad94c4c35fe6737424f0a1659500a798fca2ac63/?at=release%2Fv0.6>