Information Tags Maintenance: Anchoring

Karol RÁSTOČNÝ*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova, 842 16 Bratislava, Slovakia
rastocny@fiit.stuba.sk

Current content processing and presenting systems create a lot of different metadata that contain valuable information, for example logs about users' behavior or derived concepts. These metadata are closely related to their resources – data in repositories of information spaces. But these data are not static and all their modifications affect validity of metadata, so metadata have to be maintained. Because several types of metadata exist and probably each type needs specialized maintenance approach, we have aimed to information tags (descriptive metadata with semantic relations to a tagged content) and we are working on a proposition of automatic information tags maintenance approach and information tags representation which is suitable for effective maintenance.

Due to structural similarity of information tags to annotations, we based information tags model on widely accepted Open Annotation model. Open Annotation model allows complex structures and it is proposed for RDF repositories, so we have lightened the Open Annotation model and we have redesigned it to an object model which can be stored to fast and scalable MongoDB repository.

Problem of metadata maintenance has not any sufficient solution. But this problem can be divided to two partly indifferent sub-problems. The first is maintenance of anchoring, which can be solved by accurate robust position descriptor [2]. The second problem is maintenance of bodies of metadata. This problem is not solved in current approaches of metadata maintenance.

In case of our research project PerConIK we utilize information tags mainly for tagging parts of source code by behaviour information about programmers and information about source code's features. To deal with problem of anchoring in source code we proposed an approach, which works with source code files like with sequences of textual elements (lines or words). Our approach uses location descriptor within a target (source file or an AST element of a source file), which consists of two partial descriptors with different scope of use cases:

* Supervisor: Mária Bieliková, Institute of Informatics and Software Engineering

Spring 2013 PeWe Workshop, April 5, 2013, pp. 87–88.

1. *Index-based location descriptor* – the index-based location descriptor contains indexes of the first and the last letter from a target text in a target.
2. *Context-based location descriptor* – the context-based location descriptor is used as the robust location descriptor. A context-based location descriptor contains information about:
   a. *Tagged text* – sequence of textual elements, that has been tagged in a target;
   b. *Context before and after tagged text* – minimal unique sequence of textual elements that are directly before and after tagged text;

We interpret context-based location descriptors as sequences of textual elements. The approach gives us opportunity to break up problem of time and memory complexity of approximate string matching [1] to two smaller parts – comparing textual elements and local sequence alignment that are processed separately:

1. *Compare textual elements* – unique textual elements of the source code are compared to unique textual elements of a context-based location descriptor;
2. *Local sequence alignment* – locations of a tagged text are searched as a sub-sequence of the source code via the Smith-Waterman local sequence alignment algorithm. In this step we also calculate scores for each matched location.
3. *Calculate scores of matches of contexts* – scores of contexts before and after tagged text are calculated for each possible tagged text's alignment. Confidences are calculated via the Smith-Waterman algorithm whose scoring function uses as aligned sequences connected to alignment of tagged texts.
4. *Calculate confidences of matched locations* – confidences of matched locations are calculated as linear combination of scores for each possible alignment.

We have processed preliminary evaluations of proposed anchoring approach. In the evaluations, we proved that the approach is useable for real-time processing and that it is enough accurate for anchoring information tags in source code. We performed these evaluations with several configurations, while optimal configuration (from tested values) seem to be usage of the Jaro-Winkler string similarity algorithm for comparing textual elements, while if our anchoring approach has to be used with a longer source code, usage of lines as textual elements ensures usability for real-time anchoring.

Extended version was published in Proc. of the 9th Student Research Conference in Informatics and Information Technologies (IIT.SRC 2013), STU Bratislava, 82-89.

Acknowledgement. This contribution is the partial result of the Research & Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

[1] Fredriksson, K., Navarro, G.: Average-optimal Single and Multiple Approximate string matching. *ACM J. of Experimental Alg.*, (2005), vol. 9, no. 1.4, pp. 1–47.
[2] Phelps, T.A., Wilensky, R.: Robust Intra-document Locations. *Computer Networks*, (2000), vol. 33, no. 1-6, pp. 105–118.