**CONCLUSION**

To locate a bug, developers use not only the content of the bug report but also domain knowledge relevant to the software project. We introduced a learning-to-rank approach that emulates the bug finding process employed by developers. The ranking model characterizes useful relationships between a bug report and source code files by leveraging domain knowledge, such as API specifications, the syntactic structure of code, or issue tracking data. Experimental evaluations on six Java projects show that our approach can locate the relevant files within the top 10 recommendations for over 70 percent of the bug reports in Eclipse Platform and Tomcat. Furthermore, the proposed ranking model outperforms three recent state-of-the-art approaches. Feature evaluation experiments employing greedy backward feature elimination demonstrate that all features are useful. When coupled with runtime analysis, the feature evaluation results can be utilized to select a subset of features in order to achieve a target trade-off between system accuracy and runtime complexity. The proposed adaptive ranking approach is generally applicable to software projects for which there exists a sufficient amount of project specific knowledge, such as a comprehensive API documentation (Section 3.1.2) and an initial number of previously fixed bug reports (Section 6.1). Furthermore, the ranking performance can benefit from informative bug reports and well documented code leading to a better lexical similarity (Section 3.1.1), and from source code files that already have a bug-fixing history (Section 3.2). In future work, we will leverage additional types of domain knowledge, such as the stack traces submitted with bug reports and the file change history, as well as features previously used in defect prediction systems. We also plan to use the ranking SVM with nonlinear kernels and further evaluate the approach on projects in other programming languages.