3.8 Influence of the Similarity Metrics

A number of similarity metrics have been proposed to assess name similarity [16] (details are presented in Section 6.3). To assess to what extent the conclusions drawn in previous sections hold when different similarity metrics are employed, we repeat the analysis with *Levenshtein distance-based* and *JaroWinkler-based* similarity metrics. On one hand, the evaluation results may provide hints for the selection of similarity metrics in name-based argument analyses tasks if different similarity metrics lead to essentially different properties of the argument and parameter names. On the other hand, if there is no essential difference, the evaluation results may reveal that the conclusions drawn in previous sections do not depend on specific similarity metrics, which generalizes the conclusions.

3.8.1 Alternative Metrics

For the Levenshtein distance-based similarity metrics, we compute the similarity of two names as follows. First, we calculate the Levenshtein distance $LevDistance(arg, par)$ between the argument name arg and the parameter name par. Levenshtein distance is a way to quantify how dissimilar two strings are by counting the minimum number of character operations (e.g, insertion, deletion, substitution) required to transform one string into the other. For example, $LevDistance(\text{“}name\text{”}, \text{“}getName\text{”}) = 3$. Second, we compute the Levenshtein distance-based similarity between the argument name arg and the parameter name par as follows:

$$LevSim(arg, par) = \frac{maxLength(arg, par) - LevDistance(arg, par)}{maxLength(arg, par)} \quad (8)$$

$maxLength(arg, par)$ stands for the max length of arg and par. *LevSim* gives the fraction of the longer name that matches the shorter one and does not have to be edited. For example, $LevSim(\text{“}name\text{”}, \text{“}getName\text{”}) = \frac{7-3}{7} = 57\%$.

For the JaroWinkler-based similarity metrics (called *jwSim* for short), we compute the similarity $jwSim$ between the argument name arg and the parameter name par, which is based on the number and order of the common characters between the two names, as follows:

$$jwSim(arg, par) = jaroSim(arg, par) + len * p(1 - jaroSim(arg, par)) \quad (9)$$

where p is a constant scaling factor for how much the score is adjusted upwards for having common prefixes, len is the length of common prefix at the start of the string, and $jaroSim$ is computed as follows:

$$jaroSim(arg, par) = \frac{1}{3}(\frac{m}{|arg|} + \frac{m}{|par|} + \frac{m-r}{m})$$

where m is the number of matching characters, and r is the number of matching (but different sequence order) characters.

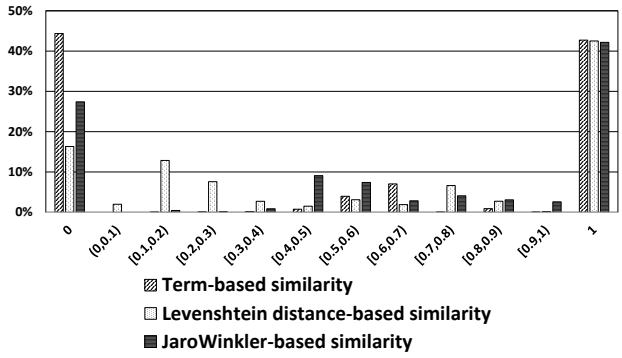

Fig. 11: Distribution of similarity with different metrics.

3.8.2 Results

3.8.2.1 Distribution of Similarity: The distribution of lexical similarity measured with different metrics is compared in Fig. 11. From this figure we observe that the distribution of the lexical similarity remains a U-shape when different similarity metrics are employed. However, we also observe some difference. A significant difference is that Levenshtein distance-based similarity metrics greatly reduce the number of arguments whose similarity with their corresponding parameters is zero (completely different). It is nature in that Levenshtein distance-based similarity metrics take two terms as completely different only if they share no common characters at all. In contrast, term-based similarity metrics take them as completely different whenever they are not identical (but may share some common characters).

3.8.2.2 Length of Names: The correlation between length of names and the lexical similarity measured with different metrics is presented in Fig. 12. From this figure, we observe that switching from one metric to another has little influence on the correlation. We observe that no matter which similarity metrics are used, the similarity increases when the length of parameter names increases, whereas it decreases when the length of argument names increases.

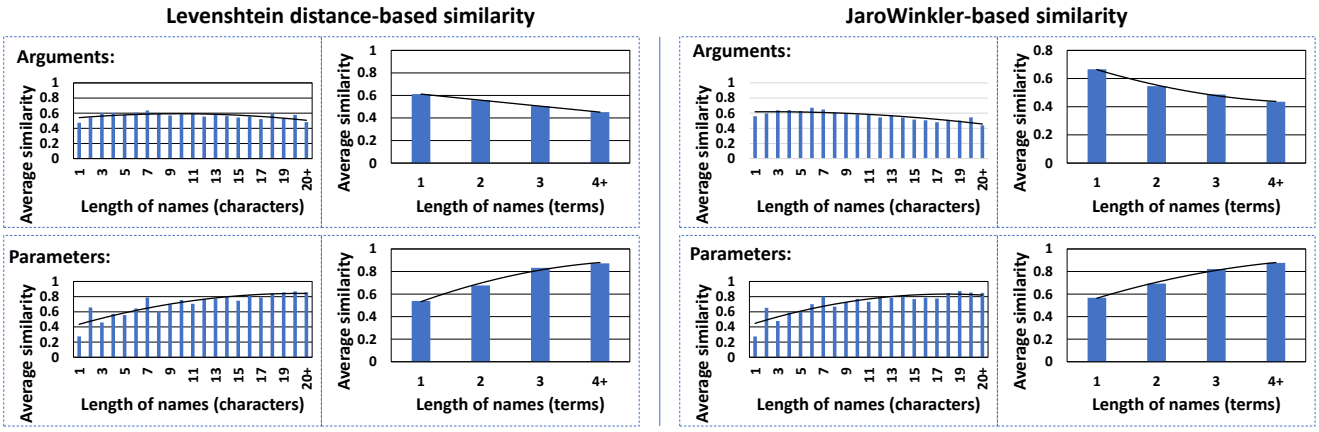

Fig. 12: Correlation between length of names and average similarity (different metrics)

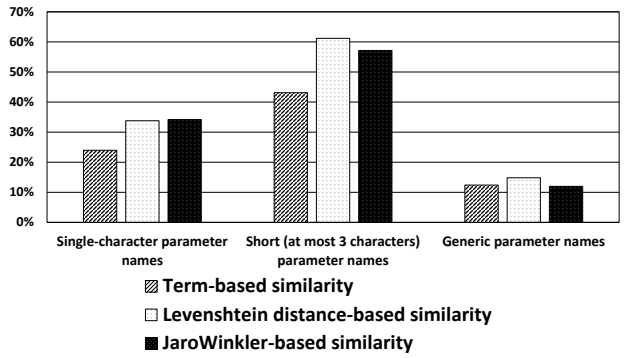

Fig. 13: Reasons for dissimilarity between arguments and parameters

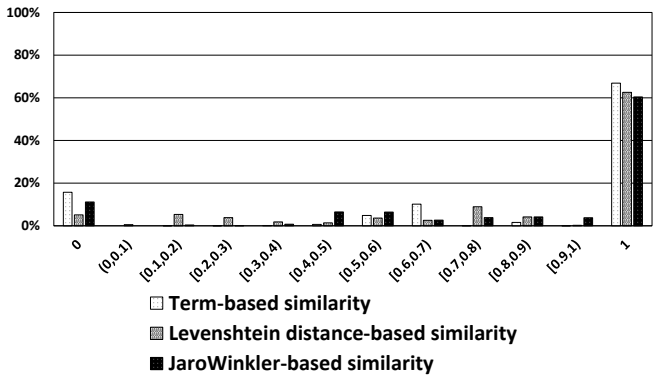

Fig. 14: Distribution of similarity between parameters and arguments associated with low-similarity parameters

3.8.2.3 Reasons for Dissimilarity: Short parameter names and generic parameter names keep to be the major reasons for the dissimilarity even if different metrics are employed. Details are presented in Fig. 13. The vertical axis represents the percentage of dissimilar arguments that are associated with specific kind of parameter names. For example, the first bar on the left means that 24% of the arguments whose term-based similarity with their corresponding parameters is zero are associated with single character parameter names. From this figure we observe that the percentage of dissimilarity caused by short parameter names, single character parameter names, or generic parameter names changes a little when switching between different similarity metrics.

From these results, we conclude that switching between different similarity metrics has little influence on the reasons for dissimilarity.

3.8.2.4 Filtering Low Similarity Parameters: In Section 3.6, we find that we can filter out low-similarity parameters with a set of low-similarity parameters from sample applications. To investigate whether such low-similarity parameters can be filtered out as well when different similarity metrics are employed, we build a set of low-similarity parameters for each kind of similarity metrics using the same filtering approach as described in Section 3.6. Evaluation results are presented in Fig. 14. The results suggest that the conclusions drawn in Section 3.6 with term-based similarity metrics hold when different metrics are employed. The filtration by Levenshtein distance-based similarity metrics removes 81% of the arguments that are completely dissimilar with their corresponding parameters. Meanwhile, the filtration by JaroWinkler-based similarity metrics removes 75% of the arguments that are completely dissimilar with their corresponding parameters.

We conclude from the results that building a set of low-similarity parameters based on sample applications is a good way to filter out parameters that are likely to be assigned to dissimilar arguments, no matter which lexical similarity metric is used.

3.8.2.5 Picking Among Alternative Arguments: In Section 3.7, we find that most arguments that are not associated with low-similarity parameters are more similar to their corresponding parameters than any other alternative arguments. To study whether the conclusions hold when switching between different similarity met-

rics, we repeat the analysis conducted in Section 3.7 with Levenshtein distance-based and JaroWinkler-based metrics, respectively.

The results show that most arguments that are not associated with low-similarity parameters are more similar to their corresponding parameters than any other alternative arguments as well when Levenshtein distance-based or JaroWinkler-based similarity metrics are employed. Among the arguments who have at least one alternative, 69% and 61% of them are more similar to their corresponding parameters than any of their alternatives when Levenshtein distance-based similarity metrics and JaroWinkler-based similarity metrics are employed, respectively. For the arguments that are not associated with low-similarity parameters, 78% and 75% of them are more similar to their corresponding parameters than any of their alternatives when Levenshtein distance-based similarity metrics and JaroWinkler-based similarity metrics are employed, respectively. We conclude from the results that switching between different similarity metrics has little influence on the similarity of alternative arguments.

3.8.2.6 Conclusions on Different Similarity Metrics: From the results in Sections 3.8.2.1 - 3.8.2.5, we conclude that the major conclusions drawn in Sections 3.1 - 3.7 hold as well when switching from term-based similarity metric to Levenshtein distance-based or JaroWinkler-based similarity metric are employed:

- Most arguments are either extremely similar or extremely dissimilar with corresponding parameters. For convenience, we call such arguments *extreme arguments*. They account for 87%, 59%, and 70% of the analyzed arguments when term-based similarity, Levenshtein distance-based similarity, and JaroWinkler-based similarity are employed, respectively.
- The similarity increases when the length of parameter names increases, whereas it decreases when the length of argument names increases. The trends keep unchanged while different similarity metrics are employed.
- Short parameter names and collection-related parameter names are the major reasons for the dissimilarity. They lead to 56%, 76%, and 69% of the dissimilar argument-parameter pairs when term-based similarity, Levenshtein distance-based similarity, and JaroWinkler-based similarity are employed, respectively.
- Dissimilar argument-parameter pairs can be filtered out by building a filter list of low similarity parameter names. 81%, 81%, and 75% of the dissimilarity argument-parameter pairs are filtered out when term-based similarity, Levenshtein distance-based similarity, and JaroWinkler-based similarity are employed, respectively.
- Most arguments that are not associated with low-similarity parameters are more similar to their corresponding parameters than any other alternatives. The ratio is up to 66%, 78%, and 75% when term-based similarity, Levenshtein distance-based similarity, and JaroWinkler-based similarity are employed, respectively.