A USER EVALUATION OF UNIXCODER USING STATEMENT COMPLETION IN A REAL-WORLD SETTING

Jorit de Weerdt
j.c.h.p.deweerdt@student.tudelft.nl

Supervisor: Maliheh Izadi
Advisor: Georgios Gousios
Professor: Arie van Deursen

0. BACKGROUND

Natural language processing models grant computers the ability to read, speak, and understand human languages. But, these models are not limited to human languages, and could be expanded with abstract syntax trees to understand code and, in turn, predict code.

State-of-the-art models have a theoretical accuracy of 70% up to 4 token lengths [1] when predicting source code in a tailored environment.

This promised accuracy makes code auto-completion a highly sought-after code extension as it can make development more efficient.

1. PROBLEM

However, the only metrics available to gauge the effectiveness of an auto-completion model are the accuracy on test source code and the potential latency of the model. Many state-of-the-art models have high accuracy scores once trained and evaluated on their source code datasets. But, this does not indicate how well such a model performs for a developer. A model could have an impeding inference time for each auto-completion or it might only suggest straightforward predictions.

Therefore, to properly evaluate a model, the interactions between the model and the developer have to be analysed in a real-world setting.

2. RESEARCH QUESTIONS

RQ1: What is the acceptance rate of suggestions from the model when used by developers?
RQ2: How useful is the model from the perspective of its users?
RQ3: How can the acceptance rate of the model be improved for everyday coding use?

3. METHODOLOGY

We created a plugin (VSC & Jetbrains) called "Code4Me". Code4Me provides the developer with a suggestion whenever a trigger point is typed or the keybind is used. The plugin sends a request with 3992 characters of context to the remote API and the server returns the a prediction generated by the model. In this study the model UniXcoder was evaluated. Then, Code4Me shows the suggestion and a verification process starts. This line was sent back to the server after 30 seconds and then compared to the original suggestion.

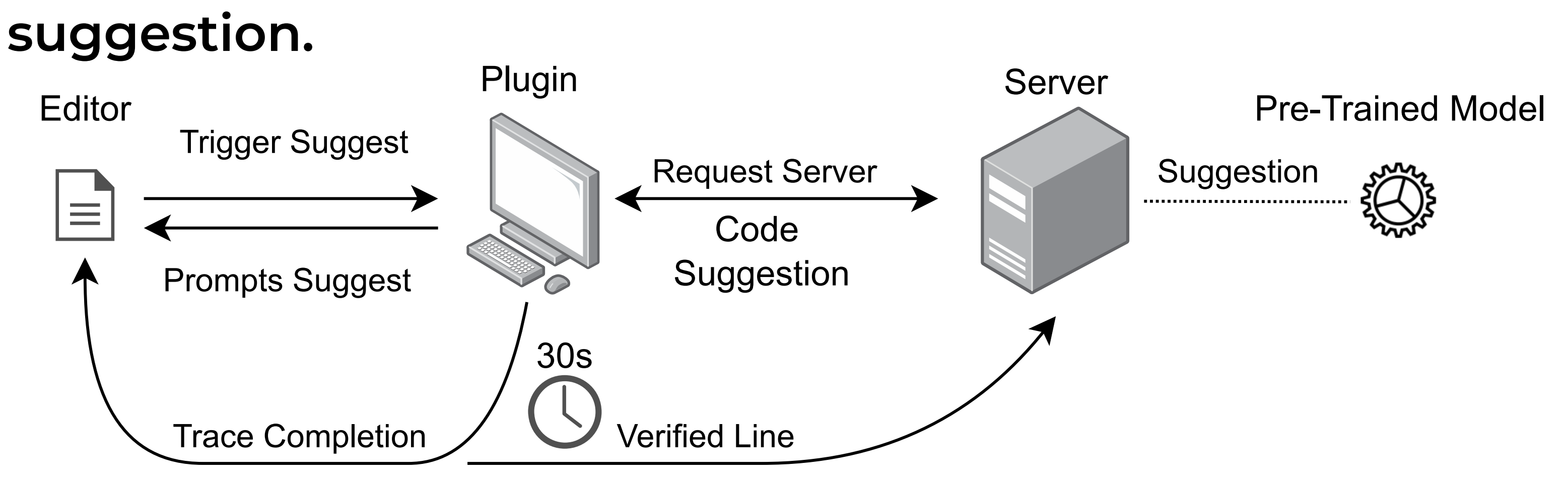

Figure 1: Workflow

4. RESULTS

Out of all 450+ downloads, 32 users programmed with Python. The results of the study can be found in Fig. 2, Fig. 3, Table 1, and Table 2.

The Exact Match metric shows that 62.5% of selected suggestions were unchanged by the users and therefore, accurate. Furthermore, BLEU~4 [2], METEOR [3], and ROUGE-L [4] are metrics used for evaluating natural language. Lastly, the Edit Similarity indicates how similar the suggestions were to the verified suggestions.

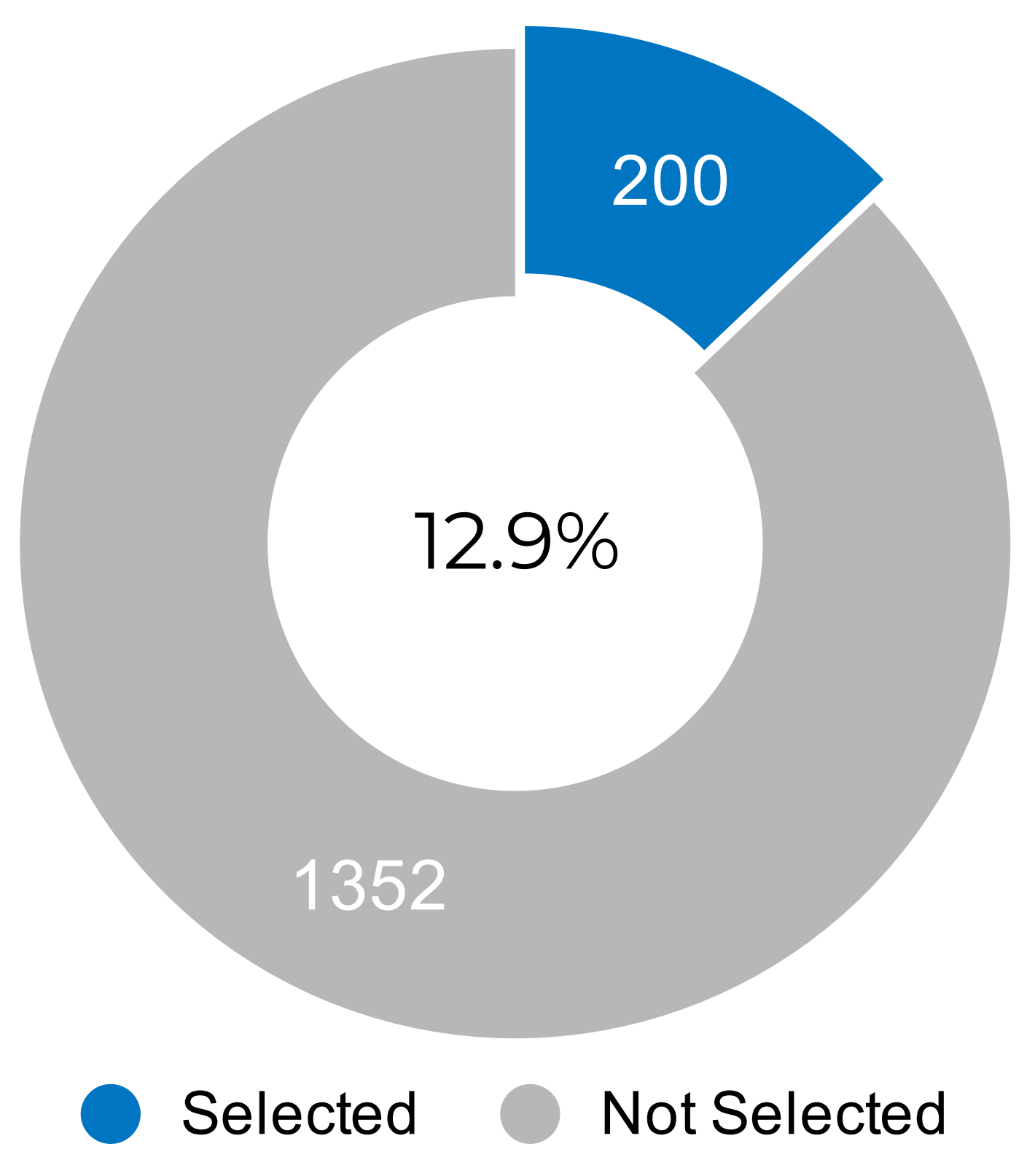

Figure 2: Number of Selected Suggestions

	Metric	Total
	Occurrence	1552
	Exact Match	22.49
	BLUE~4	34.89
ROUGE-L	Precision	40.56
	Recall	42.09
	F1 Score	39.65
	Edit Similarity	49.81
	METEOR	39.10

Table 1: Overall metrics.

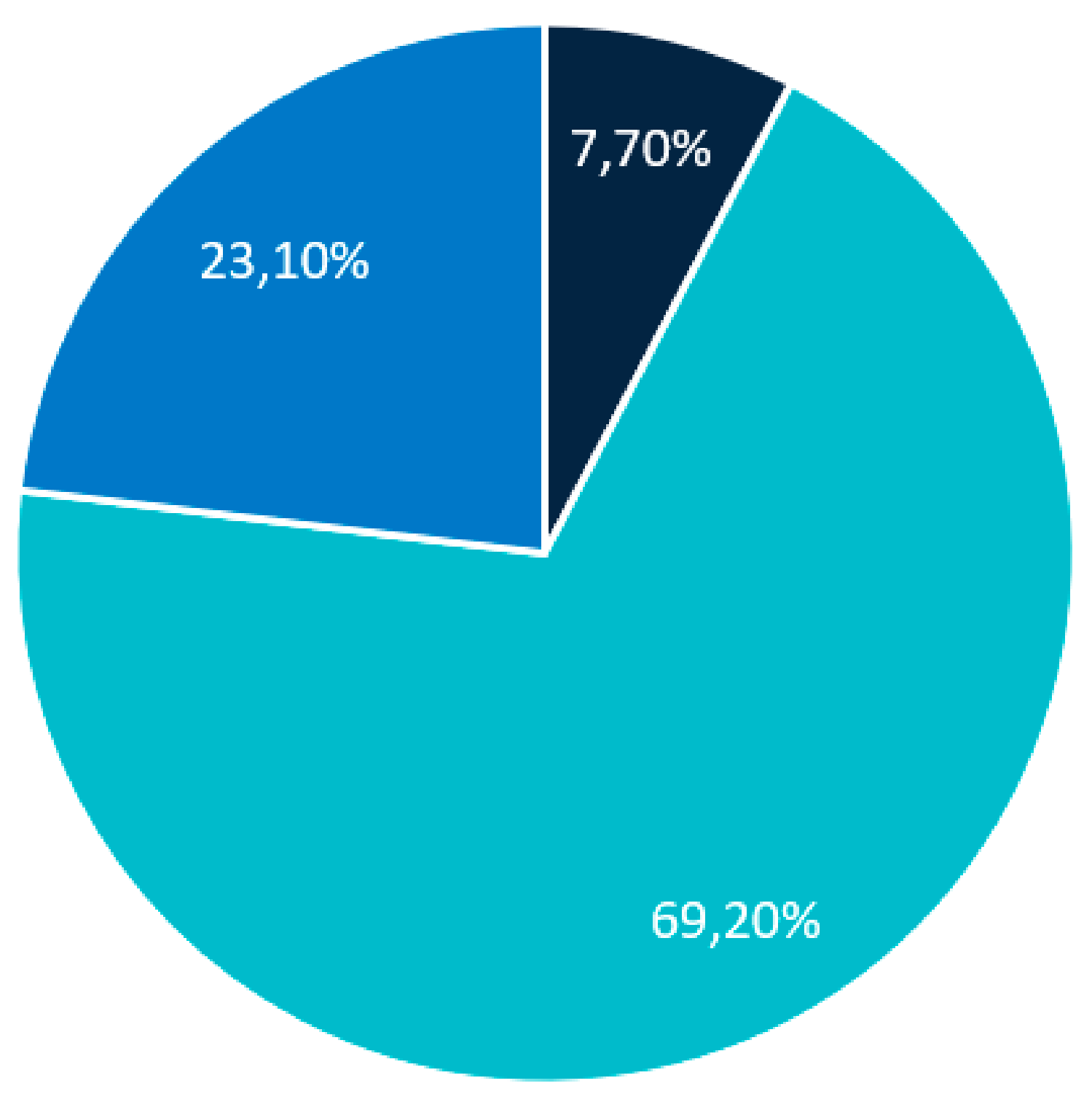

Figure 3: Perceived accuracy by users.

	Metric	Explicit Select	Not Selected
	Occurrence	200	1352
	Exact Match	62.50	16.57
	BLUE~4	67.09	30.13
ROUGE-L	Precision	78.70	34.92
	Recall	78.01	36.77
	F1 Score	76.92	34.14
	Edit Similarity	84.88	44.62
	METEOR	75.42	33.73

Table 2: Metrics per selection type.

5. CONCLUSION

The performance of the model when users select the suggestion is good with an exact match of 62.5% and high edit similarity. This is reflected by the opinion of the users as shown in Fig. 2.

The overall scores are lower than what is listed in the findings by Guo et al. [5] for UniXcoder, but when a suggestion is explicitly selected by the user then scores surpass their findings. Lastly, the acceptance rate of the model could be improved by providing multiple suggestions to the user.

[1] M. Izadi, R. Gismondi, and G. Gousios, “Codefill: Multi-token code completion by jointly learning from structure and naming sequences,” Feb. 2022.
[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu “Bleu: A method for automatic evaluation of machine translation,” in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115 / 1073083 . 107313
[3] S. Banerjee and A. Lavie, “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments,” in Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Mea- sures for Machine Translation and/or Summarization, Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72
C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in Text Summarization Branches Out, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.
[5] D. Guo, S. Lu, N. Duan, Y. Wang, M. Zhou, and J. Yin, Unixcoder: Unified cross-modal pre-training for code representation, 2022. DOI: 10.48550/ARXIV.2203.03850