Techniques used to extract info for summarising source code:

41% of the research summarised by Zhu et al [x]

Zhu et al.'s review of automatic code summarisation techniques indicates that the primary methods used for extracting data to generate code summaries were information retrieval, machine learning, and artificial neural networks.

1. Information retrieval: “Information Retrieval is widely used to obtain proper information from source code to automatically generate natural language descriptions.”
   1. Keyword identification approaches
      1. G. Salton, A. Wong, and C.-S. Yang, “A vector space model for automatic indexing,” Communications of the ACM, vol. 18, no. 11, pp. 613–620, 1975.
      2. D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” Journal of machine Learning research, vol. 3, no. Jan, pp. 993–1022, 2003.
      3. T. K. Landauer, P. W. Foltz, and D. Laham, “An introduction to latent semantic analysis,” Discourse processes, vol. 25, no. 2-3, pp. 259–284, 1998.
      4. S. Haiduc, J. Aponte, and A. Marcus, “Supporting program comprehension with source code summarization,” in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2. ACM, 2010, pp. 223–226.
   2. Eye tracking techniques found that tested developers spend more time reading method signatures than method invocations
      1. P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D’Mello, “Improving automated source code summarization via an eye-tracking study of programmers,” in Proceedings of the 36th international conference on Software engineering. ACM, 2014, pp. 390–401.
   3. Topic modelling techniques (hierarchical PAM)
      1. D. Mimno, W. Li, and A. McCallum, “Mixtures of hierarchical topics with pachinko allocation,” in Proceedings of the 24th international conference on Machine learning. ACM, 2007, pp. 633–640.
   4. Static analysis
      1. S. Rastkar, “Summarizing software concerns,” in Proceedings of the 32nd ACM/IEEE International Conference on Software EngineeringVolume 2. ACM, 2010, pp. 527–528.
      2. Java program to AST – with predefined natural language template. This approach is very similar to the approach we selected to develop JSONTalk K. A. DAWOOD, K. Y. SHARIF, and K. T. WEI, “Source code analysis extractive approach to generate textual summary.” Journal of Theoretical & Applied Information Technology, vol. 95, no. 21, 2017.
2. Stereotype identification: “Stereotypes are abstractions of methods’ or classes’ types and roles in software systems”
   1. JSummariser: identified stereotypes within classes and defined different text templates for summarisinhg different stereotypes
      1. L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. VijayShanker, “Automatic generation of natural language summaries for java classes,” in 2013 21st International Conference on Program Comprehension (ICPC). IEEE, 2013, pp. 23–32.
   2. Summary templates created for each method steryotype
      1. N. J. Abid, N. Dragan, M. L. Collard, and J. I. Maletic, “Using stereotypes in the automatic generation of natural language summaries for c++ methods,” in 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2015, pp. 561–565.
   3. Micropatterns: “abstractions of a method’s function, but a method can be mapped to several micropatterns”
      1. M. Malhotra and J. K. Chhabra, “Class level code summarization based on dependencies and micro patterns,” in 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE, 2018, pp. 1011–1016.
      2. Converted code to XML and then identified micropatterns from there: S. Rai, T. Gaikwad, S. Jain, and A. Gupta, “Method level text summarization for java code using nano-patterns,” in 2017 24th AsiaPacific Software Engineering Conference (APSEC). IEEE, 2017, pp. 199–208.
3. External description usage
   1. “crawled code segments together with their descriptions from … Stackoverflow”. Autocomment
      1. E. Wong, J. Yang, and L. Tan, “Autocomment: Mining question and answer sites for automatic comment generation,” in 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2013, pp. 562–567.
   2. Analysis of existing code repositories and identification of similar code within external repositories to generate comments. Some use of NLP
      1. E. Wong, T. Liu, and L. Tan, “Clocom: Mining existing source code for automatic comment generation,” in 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER). IEEE, 2015, pp. 380–389.
   3. Gamified crowdsourcing of code-description mappings
      1. S. Badihi and A. Heydarnoori, “Crowdsummarizer: Automated generation of code summaries for java programs through crowdsourcing,” IEEE Software, vol. 34, no. 2, pp. 71–80, 2017.
   4. Mining of commit-comment pairs
      1. ] Y. Huang, Q. Zheng, X. Chen, Y. Xiong, Z. Liu, and X. Luo, “Mining version control system for automatically generating commit comment,” in Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE Press, 2017, pp. 414–423.
4. Natural language processing
   1. Process code using a SoftwareWord Usage Model (SWUM), to capture relationships betweed words within code.
      1. X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, “Measuring program comprehension: A large-scale field study with professionals,” IEEE Transactions on Software Engineering, vol. 44, no. 10, pp. 951976, 2017.
   2. Use of the page rank algorithm (A. N. Langville and C. D. Meyer, Google’s PageRank and beyond: The science of search engine rankings. Princeton University Press, 2011.) to retrieve the most important methods.
      1. P. W. McBurney and C. McMillan, “Automatic documentation generation via source code summarization of method context,” in Proceedings of the 22nd International Conference on Program Comprehension. ACM, 2014, pp. 279–290.
   3. Abstract Synatx tree : “Wang et al. [S24] utilized an abstract syntax tree(AST) and operations performed on related objects to identify objectrelated action units in the method”
      1. X. Wang, L. Pollock, and K. Vijay-Shanker, “Automatically generating natural language descriptions for object-related statement sequences,” in 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2017, pp. 205–216.
5. Machine learning and artificaila neural networks
   1. Supervised learning: Support vector machines (SVM), Naïve Bayes.
      1. N. Nazar, H. Jiang, G. Gao, T. Zhang, X. Li, and Z. Ren, “Source code fragment summarization with small-scale crowdsourcing based features,” Frontiers of Computer Science, vol. 10, no. 3, pp. 504–517, 2016.
   2. Supervised learning: Ranking descriptive sentencesusing and SVM classifier
      1. S. Rastkar and G. C. Murphy, “Why did this code change?” in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013, pp. 1193–1196.
   3. Unsupervised learning: Tree-based autofolding software summarisation algorithm (TASSAL). Optimises similarity between summary and source code. Autofolding folds less informative code regions
      1. J. Fowkes, P. Chanthirasegaran, R. Ranca, M. Allamanis, M. Lapata, and C. Sutton, “Autofolding for source code summarization,” IEEE Transactions on Software Engineering, vol. 43, no. 12, pp. 1095–1109, 2017.
   4. Unsupervised learning: CODE-NN, uses long short-term memory LSTM (a kind of recurrent neural network (RNN)) “attention procedure to produce highlevel summaries that describe C# code snippets and SQL queries.” Again, this model was trained on data from Stackoverflow. (found that this model outperformed Info retrieval based methods)
      1. S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, “Summarizing source code using a neural attention model,” in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 2073–2083.
   5. DeppCom: applies NLP techniques, uses RNN and LSTM. Outperforms CODE-NN
      1. X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, “Deep code comment generation,” in Proceedings of the 26th Conference on Program Comprehension. ACM, 2018, pp. 200–210.
   6. Code Attention : “RNN-based attention module to understand the structure of code.
      1. W. Zheng, H.-Y. Zhou, M. Li, and J. Wu, “Code attention: Translating code to comments by exploiting domain features,” arXiv preprint arXiv:1709.07642, 2017.
   7. “RNN to encapsulate critical structural info od source code and produce natural language comment”
      1. Y. Liang and K. Q. Zhu, “Automatic generation of text descriptive comments for code blocks,” in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
   8. Novel convolution attentional network used to generate extremely concise summarisation of code
      1. M. Allamanis, H. Peng, and C. Sutton, “A convolutional attention network for extreme summarization of source code,” in International Conference on Machine Learning, 2016, pp. 2091–2100.

How to generate natural language descriptions and what kind of summary can be automatically generated from summarizing source code?

Extracting information from source code should be followed by generating natural language descriptions, where summaries can be categorized as abstractive or extractive, often combined for better results, with extractive summary being unprocessed units while abstractive information is high-level and summarized, and both data extraction and summary generation techniques are related, to be discussed in the following subsections.

1. Term-based Summarization:
   1. Generates summary with most relevant terms for specific software unit
   2. Uses information retrieval techniques
   3. Haiduc et al. [S1][S2], Rodeghero et al. [S3], McBurney et al. [S33] used information retrieval techniques to extract information and generate keyword list capturing source code semantics
2. Template-based Summarization:
   1. Most common natural language summary generation method
   2. Researchers predefine set of summary templates and fill in templates based on target code segment and other information
   3. Dawood et al. [S34], Hammad et al. [S5], Wang et al. [S24], McBurney et al. [S23], Sridhara et al. [S20][S21][S22], Badihi et al. [S18] use this method
   4. Stereotype identification closely related to template-based generation
3. External-description-based Summarization:
   1. Uses external data such as comment-code mappings in other repositories or website forums
   2. Wong et al. [S16] collected comment-code mappings from StackOverflow and open source projects in GitHub to find most relevant comments to target code segment
   3. Huang et al. [S19] directly applies comment of most similar commit to input commit
4. Machine-learning-based Summarization:
   1. Machine translation techniques and neural network based natural language generators are more prevalent and efficient
   2. Oda et al. [S35] used statistical machine translation (SMT) to generate pseudo code from source code
   3. Iyer et al. [S27] developed CODENN, an RNN with attention that directly distributes comments' words to code tokens
   4. Allamanis et al. [S32] proposed a convolutional neural network for extreme summarization
   5. Hu et al. [S28][S29], Zheng et al. [S30], Liang et al. [S31], Jiang et al. [S36], and Loyola et al. [S37] used sequence-to-sequence (Seq2Seq) model for translation task, with encoder, decoder, and attention components.