DECOM: the Large Language Model to help write those pesky little comments

Marcus Roldan Nov 28, 2023 9:50 AM EST

With a little imagination, this is what stand-up meetings could be like in the future. Adobe Stock

- **DECOM is a large language model tool that generates useful documentation for any source code it is given.**
- **DECOM yields high-quality documentation because of its unique human-like deliberation process.**
- **Integrating DECOM into software development workflows will result in more efficient resource usage and increased interoperability between developers.**

How can you, as a manager or team leader, leverage large language models to allow your team to work to its fullest potential? The newly developed DECOM model promises to become an indispensable weapon in every modern software engineers' arsenal.

In the world of software development, managers have the unenviable task of mediating between the project stakeholders and the engineers who make the magic happen.

As project timescales become ever smaller, development teams become bigger. With increased (and often remote!) collaboration between engineers, documentation of software systems plays a critical role in the success and efficiency of an engineering team. As is often the case, specs are never complete at the outset of a project and are known to undergo drastic changes during development.

How can you always be sure that documentation matches implementation while specs are fluctuating during tight development cycles? The traditional answer is you can't, and this has been the root cause of many headaches managers have experienced.

DECOM: the software engineers' pocket calculator

DECOM allows for useful, accurate, and consistent documentation to be generated for any piece of source code at the click of a button.

Engineers' time is an extremely valuable resource. By reducing the amount of time that engineers need to dedicate to creating and maintaining high-quality documentation, more time can be spent extending, refactoring, or debugging the source code itself. Not to mention that by alleviating a perennial scourge of software engineers, this will create a morale boost, further improving the efficiency of software development teams.

Why is the DECOM model the most effective at generating useful documentation?

The answer is simple; DECOM uses a human-like iterative revision process, constantly improving the quality of its produced documentation. This can be illustrated through the following example of the drafts from DECOM during the generation process:

```
public CategoricalTable copy() {
    Map<Value, Double> newTable = new HashMap<Value, Double>();
    if(variable == null){
        variable = 1;
    }
    if(table.isEmpty()){
        return new CategoricalTable(variable);
    }
    for(Value v : table.keySet()){
        newTable.put(v, table.get(v));
    }
    return new CategoricalTable(variable, newTable);
}
```

Initial Draft: constructs a new multivariate table from a univariate table

Second Draft:	creates a new copy of the given distribution
Third Draft:	returns a copy of the table from this table
Final Draft:	**returns a copy of the probability table**
Actual Documentation:	**returns a copy of the probability table**

From "Automatic comment generation via multi-pass deliberation", F. Mu, X. Chen, L. Shi, S. Wang, and Q. Wang, doi:10.1145/3551349.3556917

The science and math behind the magic

To generate the initial draft, the model looks through its catalog of documented source code, and finds the most similar code in the catalog to the input. DECOM then retrieves the associated comment from the catalog to serve as the first draft.

The subsequent drafts utilize a complex system (the exact details of which are frankly irrelevant here…) to identify the semantic similarity between the draft comment and the input source code. In other words, DECOM evaluates the information being conveyed by the comment and compares it to the source code.

If DECOM determines the comment is transmitting a sufficient level of information about the code, it finishes its deliberation and produces the documentation. If not, DECOM identifies the parts of the comment that are transmitting the least amount of information and focuses its revisions there.

In order to confirm the benefits of DECOM's iterative-revision technique, it was tested against over 28,000 Java and Python code/comment pairings. The generated comments were evaluated against actual human-written ones on different bases: the similarity of phrases and the overlap of important concepts, taking into account synonyms as well.

After this rigorous testing gauntlet, DECOM was shown to create more accurate, but most importantly, more meaningful documentation. It was not uncommon for the DECOM-generated documentation to contain more information than its human-generated counterpart!

Ultimately, DECOM represents a novel, but beneficial way to integrate AI tools into the Software Engineering workflow. Allowing developers to outsource their 'paperwork' gives more time and focus on the meaningful problem-solving tasks for their projects.

Through DECOM, new levels of efficiency, consistency, and quality can be achieved by your development team.

To those who want to peel back the curtain on DECOM, the paper can be found here, or the GitHub repo for the untrained model can be found here.

To those who are not fully convinced yet, DECOM has been embedded below, allowing for its utility and consistency to be demonstrated.