servicenow

ServiceNow Text-to-Code LLM Model Card

ServiceNow Text-to-Code LLM Model Card

Intended Use and Functionality

Purpose of the Model

The ServiceNow AI Code Assistant Model is designed to support developers by providing AI-assisted code generation and completion.

It enhances productivity by generating or completing code based on developer prompts while ensuring alignment with ServiceNow Platform-specific requirements.

Autonomy Level

The Model operates at an Assistive autonomy level, offering code suggestions that developers must review and validate before implementation.

Developers retain full control over the AI-generated output, and the Model does not support autonomous code execution or deployment.

Optimization Scope and Limitations

The Model is optimized for ServiceNow Platform-specific development tasks, particularly those involving Glide JavaScript and other supported languages.

It performs best when given well-defined prompts but may have limitations in handling complex, niche programming paradigms or non-standard code patterns. Supported languages include JavaScript (GlideScript), Java, Python, and C++.

Model Name

ServiceNow Text-to-Code LLM ("the Model")

Model Version

v1.0

Model Release Date

September 2023

Model Distribution Method

ServiceNow Platform

Model License

Please refer to your agreement with ServiceNow for all license information.

Products Using this Model

Code-generation assistance for developers on the ServiceNow Platform

User Benefits

The Model is trained and fine-tuned to provide code completion suggestions, relevant code snippets, functions, and variable names, and is intended to reduce the time spent on manual typing and potential syntax errors.

Risks

The Model may generate code that is inefficient, contains bugs or contains malware. For more information, refer to section 10 of the StarCoder paper. The Model may not create code consistent with the best practices for an organization or developer and should be reviewed and tested by developers before acceptance and usage.

Factors and limitations

- The Model needs enough context in a prompt to create an acceptable response.
- The Model can only generate code for a limited number of programming languages. It currently supports a variety of languages including Java, C++, JavaScript and Python, but it is important to note that the Model may not be able to generate code for all programming languages.
- The Model may struggle to handle complex or uncommon scenarios, such as non-standard code patterns, niche programming paradigms, or specialized use cases. In such situations, the generated code may be less reliable or fail to meet the desired requirements.
- The Model may struggle with complex or ambiguous code structures.
- The Model relies on its ability to understand the context and semantics of the code being written.

Ethical considerations

The Model has been tuned with the intention of limiting bias, toxicity, and hallucinations, although such limitations may still exist.

Some PII and malicious code was removed in training data for the StarCoder family of models, but some may still exist. Additional PII and malware detection in pre- and post-processing can help mitigate this issue. Code LLMs can produce harmful code based on how it is prompted, and the Model is not free from such limitations as the behavior of other industry LLMs.

Developers should not use the Model to generate malware or in any other way prohibited by the usage restrictions for the StarCoder family of models. To perform a membership check, along with attribution to original code and licenses, tools such as HF Code Autocomplete can be leveraged.

ServiceNow does not endorse or assume any responsibility for the accuracy, completeness, legality, or appropriateness of the results generated by such tools. When using the model customers should follow ServiceNow's guidelines on intended use available on docs.servicenow.com as well as ServiceNow's AI Acceptable Use Policy

Please report instances of hallucinations, malicious code, or unwanted data in output so that we can evaluate for remediation.

Supported Languages

Spoken and Written Languages

English: The Model excels at understanding code related text inputs in English.

Coding Languages

JavaScript (GlideScript), Java, Python, C++

Model Architecture

Base Model: StarCoder model family

Fine-Tuning: Optimized for ServiceNow-specific code-generation tasks.

Number of Parameters

15B

Maximum Input and Output Size

- **Maximum Input Size:** The Model can process up to **8K context length** in a single input sequence.
- **Maximum Output Size:** The model generates output **within the same 8K context window**, dynamically allocating tokens between input and output.
- **Shared Context Window:** The **8K context window** is shared between input and output

Input and Output Modalities

Modality Type

Single-modality: The Model processes and generates **text-to-text** outputs.

Inputs

Input Type: The Model accepts **plain text** input, typically structured as questions, commands, or prompts in natural language (e.g., “Summarize this article” or “Translate this sentence to French”).

Input Constraints:

- The Model is designed primarily for natural language inputs and may perform sub-optimally with **non-text inputs** such as raw numerical data or unstructured code.
- Some **long-form inputs** may be truncated based on token limits, affecting response completeness.
- The Model may struggle with **highly ambiguous or domain-specific jargon** if not adequately trained in those areas.

Outputs

Output Type: The Model generates **plain text responses** based on the input prompt. This can include answers to questions, summarizations, translations, code snippets, or structured text.

Output Constraints: Outputs are constrained by **token length limits**, which may truncate longer responses.

Input and Output Formats

- **Input Format:** Inputs must be formatted as **plain text** with no additional structuring required.
- **Output Format:** Outputs are generated as **plain text in the form of code.**

Training Data

ServiceNow did not conduct further pre-training.

Fine-Tuning Data

Fine-tuning was done with ServiceNow Platform-specific flow data.

Evaluation Data

- HumanEval examples include a prompt written in English, a code snippet, and a unit test. Note that the programming language is Python.
- Internally evaluated using Glide JavaScript code

Metrics

The Model was optimized for Pass@1– the likelihood that a problem is solved in a single attempt by the Model. These metric measures functional correctness, i.e., if the unit tests pass.

Technical Means for Integration

All interactions and processing occur within the platform's secure architecture.