Special Tokens: The Quotation Marks of LLMs

What Happens When a Delimiter Shows Up in Plain Text

Guillaume Lethuillier
glthr.com

February 22, 2025 · updated November 14, 2026

Series: Delimiters in Languages — Part 2 of 5
Next: Formulaic Delimiters in the Iliad and the Odyssey
(glthr.com/formulaic-delimiters-in-the-iliad-and-the-odyssey)

Abstract

I apply the distinction between evaluated and represented expressions to the tokens that mark roles and turns in LLM conversations. A reserved token ID differs from its written spelling: typed text becomes a control token only if the application permits that interpretation. A separate example shows a chat application mistaking quoted `<think>` tags for reasoning markup. The application displayed "thoughts"; the model had not started reasoning.

Delimiters Between First-Order and Higher-Order Expressions

In a previous article, I introduced the general idea that *delimiters marking transitions between first-order and second-order expressions are fundamental to all languages that convey meaning.* These ideas were applied to human languages, programming languages, and even genetic code.

In a nutshell,

- **First-order expressions** are evaluated
- **Second-order expressions** are only represented (*e.g.*, quoted, stored, passed along)
- They use the same symbols, and therefore need **delimiters** that mark the transition between these categories of expressions

As an example, in the following sentence

> She told him: "It is raining outside."

- *She told him* is a first-order expression

Cite as: Guillaume Lethuillier, "Special Tokens: The Quotation Marks of LLMs", glthr.com, February 22, 2025. https://glthr.com/special-tokens-the-quotation-marks-of-llms

- *It is raining outside* is a second-order expression
- The quotation marks are the delimiters

In this article, I suggest extending this principle to LLMs. More specifically, how are delimiters being used so that LLMs can differentiate between first-order and second-order (or, more generally, higher-order) expressions? Also: why do they "break" models when they are misused?

LLM Delimiters

Why Model Creators Need Delimiters

The role of special tokens in LLMs illustrates this idea neatly. By using these special tokens, LLMs can differentiate between the frame of a conversation (who speaks, and where each message begins and ends), which is evaluated as first-order expressions, and the text of each message, which is represented as a second-order expression: the words of a speaker, as in reported speech. For instance, the tokenizer injects these tokens into the input sequence to mark where context-dependent information begins or ends. They can practically be likened to reserved words.

A special token is not a string of characters but a reserved number in the model's vocabulary. The characters `<|eot_id|>` typed by a user become that number only if the application tokenizes the user's text with special-token parsing switched on; otherwise they stay ordinary text. Mistral says it of its control tokens: the user "will never inject" them "simply because they do not exist in the string spectrum, existing only as a token ID." It is the one case in this series where a delimiter can be taken out of the alphabet. The remedy for the injections described below is therefore less to "sanitize" a prompt than to tokenize data as data.

It is interesting to see, across LLMs, the general need for pairs of delimiters to differentiate the function of the text. Mistral AI uses *control tokens* (`[INST]`, `[/INST]`, *etc.*). DeepSeek also uses special tokens, and encloses its reasoning process between `<think>` and `</think>` (pdf, p. 6). Llama uses this approach too—for instance: `<|begin_of_text|>` and `<|eot_id|>`—and community fine-tunes of version 3.2 add "reasoning tags" of their own. Claude is steered the same way with tags typed in the prompt itself: see Why XML Tags Are So Fundamental to Claude.

LLM Delimiters in Practice

Now, let's focus on a specific pair of delimiters used by Llama 3:

> `<|start_header_id|>{role}<|end_header_id|>`: These tokens enclose the role for a particular message. The possible roles can be: system, user, assistant.

These delimiters make the LLM read the text enclosed between them as a *meaningful* role for the model and not a mere textual representation of a role.

Example:

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

Translate: Bonjour !<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

The first two delimiters (`<|begin_of_text|>`, which opens the sequence, and `<|eot_id|>`, which ends the message) indicate to the LLM where the prompt begins and where the message ends. The pair of delimiters between them (`<|start_header_id|>`, `<|end_header_id|>`) indicates to the LLM that the prompt comes from the *user*. This brings a special meaning to the model. *Here, it reads the token "user" as a user-as-a-role.* The open assistant header at the end is what asks the model to reply.

This is radically different from this fictitious example:

```
<|begin_of_text|>(user) Translate: Bonjour !<|eot_id|>
```

In this statement, "user" has the same "cognitive" value as "Translate: Bonjour !".

This example also highlights the fact that, in the context of artificial intelligence, there can be several layers of purposes, each opened and closed by its own delimiters:

- `<|begin_of_text|>` marks the beginning of the sequence
- `<|start_header_id|>` marks the transition to tokens that are evaluated as a role
- `user` is evaluated as a role (first-order)
- `<|end_header_id|>` marks the end of the role and the transition to the message
- `Translate: Bonjour !` is a second-order expression: the words of the user, represented
- `<|eot_id|>` marks the end of this second-order expression

The `<think></think>` DeepSeek delimiters were, at the time of writing, the most recent illustration of the very same mechanism. The model writes its "reasoning" between them.

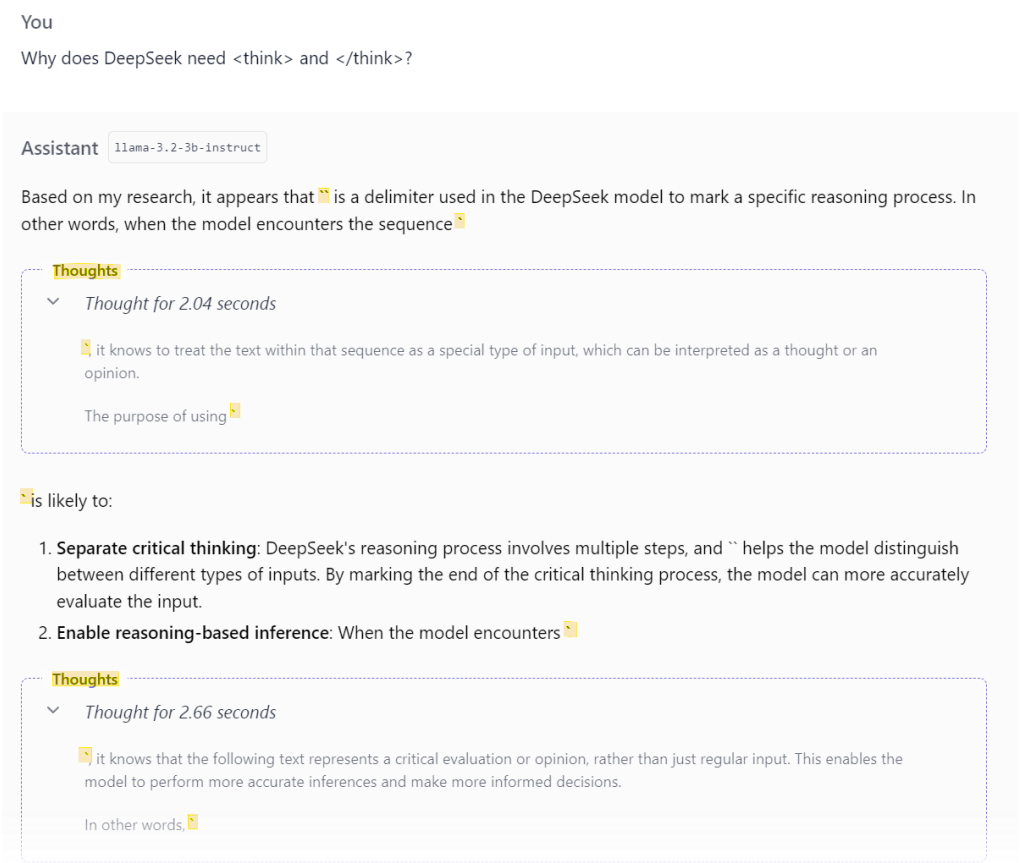
Breaking Models With Delimiters (Or: Using Delimiters in Second-Order Expressions)

Use And Abuse of LLM Delimiters

Interestingly, if delimiters are not sanitized from the prompt, they can make the model produce text that appears to be meaningless. A web article of January 2025 on "Anomalous Tokens in DeepSeek-V3 and r1" has a section on special tokens. It shows how, when a delimiter appears inside a second-order expression (so: in the text of a prompt), the model erroneously evaluates it as a real delimiter: a transition from or to another expression. In other words, when the prompt is not sufficiently sanitized, the flow of a model can be broken *by injecting delimiters.* In that sense, it is similar,

to some extent, to the class of SQL injections that use single quotation marks. While these SQL injections manipulate the transition between what is represented and what is executed, these delimiter injections aim at making a model transition to a different semantic dimension.

A related behavior is easy to reproduce. I asked a simple question to llama-3.2-3b-instruct, a model that does not reason, and quoted delimiters in my prompt: "*Why does DeepSeek need* `<think>` *and* `</think>`*?*" The model correctly identified the purpose of the question and reproduced the delimiters in its response, as text. **The chat application that displayed the response then read these quoted delimiters as real ones: it hid them and turned the text between them into "thoughts."** The model did not think; the application thought it had. A delimiter that is quoted but not escaped is read as a delimiter again, here by the next reader of the text. This is all the easier since DeepSeek-R1's tokenizer does not flag `<think>` and `</think>` as special: they travel as ordinary text, which every application has to recognize by itself.



I highlight what the application left behind: the backticks around the delimiters it hid, and the "Thoughts" boxes it drew.

LLM Delimiters Not Taken Seriously, Yet

The implications of the relationship between delimiters, first-order expressions, and higher-order expressions were, at the time of writing, seldom spelled out when a model was released, although research had begun (StruQ, February 2024, separates instructions from data with reserved tokens). Delimiters could still be handled in

unsophisticated ways. For instance, to make sure the model opens its reasoning, the DeepSeek R1 repository recommends forcing the delimiter into the output:

> Additionally, we have observed that the DeepSeek-R1 series models tend to bypass thinking pattern (i.e., outputting `<think>\n\n</think>`) when responding to certain queries, which can adversely affect the model's performance. **To ensure that the model engages in thorough reasoning, we recommend enforcing the model to initiate its response with `<think>\n` at the beginning of every output.**

Conclusion

I have argued that delimiters are crucial for LLMs to differentiate between first-order and higher-order expressions, explaining why there is a general convergence towards using delimiters regarding the creation of models. Consequently, misuse of delimiters can lead to unexpected behavior or errors in the model, making it "derail" from a given order to another. Because LLMs are evaluators and generators of languages, I believe that LLMs could be greatly improved by studying how models process delimiters and switch between first-order and higher-order expressions. Also, I recommend documenting the delimiters more systematically when a model is released.

Series and Companion Articles

Delimiters in Languages (glthr.com/page/delimiters-in-languages)

#	Article	Address
1	What Quotation Marks, String Literals and DNA Have in Common	glthr.com/quotation-marks-string-literals-and-dna
2	**Special Tokens: The Quotation Marks of LLMs** (this article)	glthr.com/special-tokens-the-quotation-marks-of-llms
3	Formulaic Delimiters in the Iliad and the Odyssey	glthr.com/formulaic-delimiters-in-the-iliad-and-the-odyssey
4	Why XML Tags Are So Fundamental to Claude	glthr.com/xml-fundamental-to-claude
5	Transitive Policies for Agents	glthr.com/transitive-policies-for-ai-agents

How to cite this article

> Guillaume Lethuillier, "Special Tokens: The Quotation Marks of LLMs", glthr.com, February 22, 2025.
> https://glthr.com/special-tokens-the-quotation-marks-of-llms

For BibTeX:

```
@misc{lethuillier2025special,
  author = {Lethuillier, Guillaume},
  title  = {Special Tokens: The Quotation Marks of LLMs},
  year   = {2025},
  month  = feb,
  url    = {https://glthr.com/special-tokens-the-quotation-marks-of-llms},
  note   = {glthr.com}
}
```

This document was typeset with LaTeX from the article published at https://glthr.com/special-tokens-the-quotation-marks-of-llms, which remains the reference version (as updated on November 14, 2026).