# Modeling Language Without Language: A ChatGPT Lesson for Language Research*

Stela Manova
https://www.stelamanova.com/

**Abstract:** In recent years, the term *token* has gained widespread use among linguists and non-linguists alike, largely due to the rise of large language models (LLMs) such as ChatGPT. However, this term does not mean what most linguists assume it does. In computational contexts, *token* refers to a subword unit—often a sequence of characters—that is neither a word nor a morpheme in the linguistic sense. ChatGPT processes only token ID numbers and operates without representations of linguistic units, which is why some computer scientists describe it as *modeling language without language*. This paper examines the terminological confusion surrounding *token* and its consequences for linguistic research on LLMs, drawing parallels to the earlier misinterpretation of the term *tree* in formal syntax. I argue that these misunderstandings stem from long-standing disciplinary divides and a reluctance among linguists to engage directly with computer science (CS) literature. Using mathematical reasoning, I show why language data—being neither ordered nor regular—require vast amounts of input for effective modeling, unlike systems based on predictable (ordered and regular) data. Finally, I reflect on the unresolved theoretical tensions between generative and usage-based linguistics in light of a functioning CS solution to language generation that aligns with neither. The paper calls for terminological clarity and increased cross-disciplinary literacy as necessary steps for future research on language.

*Proofread and edited using ChatGPT.

How fine it is if you may say:
"While the others ate,
I counted the gold scales on the fish."
*Consolation* by Boris Xristov[1]

## 1. What Is a 'Token'?

To begin with, this text is not a conventional scientific article—I deliberately do things differently. I advise readers not to take my word for granted, urge them to pause and watch a tutorial by a former OpenAI employee (a non-linguist), and refer primarily not to written sources but to freely available online videos, including materials for CS and cognitive science courses at institutions like Stanford and MIT. Despite these unconventional choices, I hope that this mathematically oriented text—written by a linguist—will prove useful and insightful for linguists.

With the rise of large language models (LLMs), *token* has become a widely used term among both linguists and non-linguists. An LLM like ChatGPT operates with a list of tokens;

[1]From *Words on Words*, translated from Bulgarian to English by Betty Grinberg, Roland Flint, and Lyuboir Nicolov.

OpenAI even sells data in terms of four-character tokens.[2] The token lists used by recent GPT models are publicly available—for instance, at https://gptforwork.com/tools/tokenizer, which provides access to both the 100K and the newer 200K token lists. From these lists, we see that some tokens coincide in form with full words, while many others do not. In fact, most tokens appear to be shorter than a typical word, which is why they are often referred to as *subword units*. This label, however, does not imply that tokens are actual linguistic morphemes. For example, in the 200K list, `less`[3] is a token (token ID[4] `[2695]`), i.e., the tokenizer (https://platform.openai.com/tokenizer) parses `catless` into two tokens, yet `catlessness`[5] is two tokens as well, and `kindness` and `blindness` are one token each, token IDs `[57842]` and `[159668]`, respectively. Why is that?

The 200K tokenizer contains the 200,000 most frequent sequences of neighboring characters in a given language. These are identified using the Byte Pair Encoding (BPE) algorithm, which I will return to later. Based on their token ID numbers (both < 200K), `kindness` `[57842]` and `blindness` `[159668]`, are among the 200k most frequent forms in English and they are therefore included in the list of tokens, while *catless* and *catlessness* are not.

Some tokens have no clear connection to linguistic units. For instance, some represent whitespace characters of various lengths (see Figure 1, a screenshot from https://gptforwork.com/tools/tokenizer). These space-based tokens are essential for coding, where precise alignment depends on specific spacing.

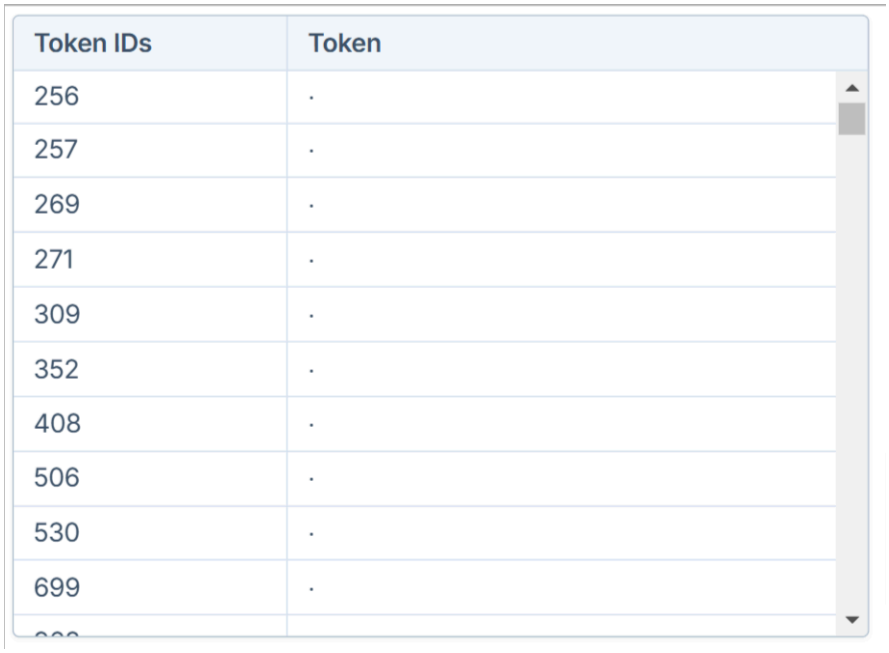


Figure 1: Tokens in the 200k list that represent whitespace characters

[2] The four-character length of a token is not a scientific fact, it is of importance only when ChatGPT charges based on data usage.
[3] Colors are only for illustrative purposes and are assigned by the tokenizer; there are only five colors.
[4] Every token has an identification number, a token ID.
[5] This is also how the human brain parses two-suffix combinations: https://benjamins.com/catalog/cilt.353.17man.

Since ChatGPT is a positional system, tokens that coincide with words behave differently depending on context. When used in longer token sequences, such tokens may be segmented differently than when used alone. For example, e.g., `meta` `[14732]` and `language` `[6439]` are valid tokens, but `metalanguage` `[11002,4088]` may also represent their combination. Similarly, `psycho` `[86207]` and `linguistic` `[113771]` are tokens, yet `psycholinguistic` is parsed into four tokens: `[10899,340,12432,6207]`.

To understand this logic, let me walk through the last two examples.

(1) Query: *metalanguage* `metalanguage` `[11002,4088]`

- The tokenizer first finds `metal` `[11002]` in the list.
- Since `metal` `[11002]` is more frequent than `meta` `[14732]` → selection of `metal` `[11002]`.
- The remaining string `anguage` is then found as a token, `anguage` `[4088]` →
  → RESULT: `metalanguage` `[11002,4088]`.

This illustrates that the model always searches for the most frequent match starting from the beginning of the list.

(2) Query: *psycholinguistic* `psycholinguistic` `[10899,340,12432,6207]`

- Found in the list: `psych` `[10899]`.
- Since `psych` `[10899]` is more frequent than `psycho` `[86207]` → selection of `psych` `[10899]`.
- The remainder *olinguistic* yields `ol` `[340]` → `psychol` `[10899,340]`
- Then, *inguistic* becomes `ingu` `[12432]` → `psycholingu` `[10899,340,12432]`
- Finally, *istic* is found in the list: `istic` `[6207]` →
  → RESULT: `psycholinguistic` `[10899,340,12432,6207]`.

In linguistics, studies on LLMs usually fall under corpus, quantitative, or computational linguistics. In these subfields, *token* typically refers to a word-form (a single instance of a word in text). This leads many linguists to assume that LLMs treat *tokens* as equivalent to words. That assumption—that tokens are words—is widespread in current linguistic writing on LLMs, but it is incorrect.

I am aware that linguists often prefer texts written by fellow linguists—even when discussing CS topics. This preference is a major reason why CS terminology is frequently misinterpreted in linguistic research. Therefore, to fully grasp what a *token* is and to evaluate the implications of the mistaken belief that GPT tokens are words, I recommend you don't simply trust my explanation (after all, I'm a linguist too). Instead, I suggest watching the popular YouTube tutorial *Deep Dive into LLMs like ChatGPT* (Feb. 5, 2025, over 2.3M views) by former OpenAI employee Andrej Karpathy : https://www.youtube.com/watch?v=7xTGNNLPyMI&t=933s; more about the author at https://karpathy.ai. Karpathy no longer publishes scientific papers, so the video is the most accessible source of his insights. It's long, but timestamped—no need to watch it all at once.

If that video doesn't suit your taste, universities like Stanford, MIT, and Harvard offer similar lectures. For example:

- Stanford CS229 | *Machine Learning – Building Large Language Models (LLMs)*: https://www.youtube.com/watch?v=9vM4p9NN0Ts&t=1005s
- MIT 6.S191 | *Introduction to Deep Learning* (video series): https://www.youtube.com/watch?v=alfdI7S6wCY&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI

At this point, I suggest you pause and watch the first 31 (ideally 43) minutes of Karpathy's video, where he explains how ChatGPT's tokenization works—something I haven't detailed above. Tokenization is the algorithm, the above-mentioned Byte Pair Encoding (BPE), that defines the list of tokens.

After watching the video, you can continue with Section 2, where I discuss how the developers of ChatGPT came up with the idea of generating language *without* language. Section 3 addresses the misinterpretation of CS terminology in linguistics and revisits the debate between usage-based and generative linguistics. Section 4 briefly explores the role of data volume in LLMs and why this issue supports neither of the two linguistic theories discussed. Section 5 concludes the text.

## 2. ChatGPT or Modeling Language Without Language

I assume you've watched the first part of Andrej Karpathy's video. You might now be asking yourself: *How did the creators of ChatGPT come up with the idea to model language in this way?*

The answer begins with a simple observation: the people behind ChatGPT are not linguists. That is, they are not constrained by traditional linguistic dogmas or theoretical commitments. Most of them are not only highly skilled in mathematics—they are mathematically gifted. By *math-gifted*, I mean people who solve complex problems using simple logic. One of the best examples of such thinking is my favorite mathematician, Carl Friedrich Gauss.

In mathematics, unlike in many other disciplines, seniority and authority carry little weight—what matters is who has the sharpest insight. The mindset of a math-gifted person might be summarized by the following principles:

1) Every problem has more than one solution.
2) All valid solutions to a problem must yield the same result.
3) Every problem has already been solved somewhere in the real world—you just need to find the right analogy.
4) If a problem remains unsolved for too long, you must change your approach.
5) The best solution is the simplest one.

Let's briefly apply each of these principles to the problem of language modeling.

***1) Every problem has more than one solution.***
In our case, the problem is: *How can we generate fluent language?*

This problem can have both linguistic and non-linguistic solutions. So far, only a non-linguistic solution exists—LLMs like ChatGPT. Linguistics, for all its formal complexity, has not produced a comparable system for fluent language generation.

***2) All valid solutions to a problem yield the same result.***
If two different approaches address the same problem, their outputs must be functionally equivalent. For language generation, that would mean both linguistic and non-linguistic models should be able to produce coherent text. At present, this cannot be tested, because—as mentioned above—a successful linguistic model of language generation does not yet exist. But the principle still holds: if one were to exist, it should produce similarly fluent output.

***3) Every problem has already been solved somewhere in the real world—you just need to find the right analogy.***
For language generation, the observable "solution" in the real world is *text*: a linear, uninterrupted sequence of characters. In CS, linear sequences are familiar territory. Binary code is a linear sequence, and modern AI is built on the idea that anything can be encoded in binary. Binary is also a positional system: the values of '0' and '1' depend on their position within the sequence.

Similarly, ChatGPT uses *tokens* in a positional manner. For instance, the character "!" is token ID #0 in the 200K token list, but " !" (with a preceding space) is a different token with a different ID—#1073. The same character in a different position results in a distinct token. As shown in Section 1, the parsing of *lessness*, *psycholinguistic*, and *metalanguage* also follows this positional logic. This applies not only to tokenization but also to token *embeddings*, which encode positional information.

*Side note:* In computing, lists start counting from zero, which is why token ID #0 exists.

***4) If a problem remains unsolved for too long, you must change your approach.***
For decades, linguists have attempted to model language generation based on words, meanings, and grammatical rules. Despite this, no system has come close to the fluency achieved by models like ChatGPT. From a problem-solving perspective, this suggests it's time to abandon the old path and try something entirely different. ChatGPT is that different approach—it works without words in the linguistic sense and without traditional semantics.

***5) The best solution is the simplest one.***
In CS, complexity has a precise technical definition (e.g., in terms of Big O notation). In linguistics, however, *complexity* is more intuitive and subjective—often defined inconsistently or not at all. From the CS perspective, a guiding principle might be: *An analysis based solely on form is less complex than one based on meaning, or on both form and meaning.*

This explains why LLMs like ChatGPT do not process form and meaning in the way linguists expect—and why they succeed. They solve the problem of fluency by operating on form alone. Their success is not accidental but grounded in the deliberate choice of a simpler, more scalable solution.

Of course, I cannot read the minds of the developers who built ChatGPT. But after listening to their explanations and observing the design decisions they've made, this logic becomes clear. And why do I think I can empathize with such thinking? Because I was trained as a math-gifted person myself—for about ten years.

### 3. The Sad Fate of CS Terminology in Linguistics

Broadly speaking, linguists fall into three types when it comes to mathematics and CS:

1) Those who lack training in math/CS and readily admit it.
2) Those who lack training in math/CS but pretend otherwise (typically without any formal education in the field).
3) Those who are actually trained in math/CS—often through degrees.

Of these, Type 2 is the most problematic. Type 2 linguists often work in computational, corpus, quantitative, or so-called mathematical linguistics, or they teach statistics to other linguists. Type 1 linguists usually steer clear of these subfields altogether, while Type 3 linguists—despite their technical competence—tend to gravitate toward theoretical linguistics and often avoid engaging with the computational mainstream of LLM-related research.

Recently, I was invited to review several manuscripts for a special issue on Learning Models of the journal *Linguistics Vanguard*. Against my better judgment, I accepted. The authors were Type 2 linguists. Their texts relied heavily on terminology borrowed from CS, yet the usage of that terminology bore little resemblance to its original meaning.

For instance:

- They had heard of *tokenization* but misunderstood it, assuming that since "tokens are words," there was no need to understand how tokens are established and used.
- Some even claimed that ChatGPT does *not* process text as a linear sequence of tokens—apparently because *recurrent neural networks (RNNs)* exist.
- That is, the authors equated RNNs with *transformers*, which is simply false (see MIT 6.S191: Recurrent Neural Networks, Transformers, and Attention, 2025: https://www.youtube.com/watch?v=GvezxUdLrEk).
- All authors assumed that LLMs operate exclusively on words—because this was in agreement with their research.

In other words, the authors were unaware that ChatGPT operates entirely on *token IDs*—abstract numerical representations—and that there's no internal representation of linguistic "words" as such. Ironically, this lack of understanding was presented as an advantage, enabling them to claim novel linguistic insights beyond the reach of computer scientists.

Despite relying on limited datasets of isolated words (not even running text), these authors argued that their findings were central to understanding LLMs and could even help improve their architecture. Disappointingly, neither the area editor nor the editors-in-chief seemed concerned that their journal was about to publish a series of papers built on

misrepresentations of how LLMs function. For the record, the (senior) authors came from Tübingen, Zurich, and Miami. I omit their names and paper titles because, to my knowledge, the papers have not yet been published.

***Why Is This Issue Urgent?***

Because some of the editors and authors involved are also part of the 2025 Linguistic Institute *Language in Use* (https://center.uoregon.edu/LSA/2025/). If these papers are published and reinforced through LSA Institute courses, the confusion over LLM-related terminology—and the underlying conceptual errors—will likely remain in the field for years to come, if not permanently. This is especially likely since linguists and computer scientists rarely attend the same conferences or review each other's work.

***A Familiar Case: The 'Tree'***

This situation is reminiscent of an earlier misuse of CS terminology in linguistics—the term *tree*. I was the first to point out (see https://ling.auf.net/lingbuzz/006082) that syntactic trees in Chomskyan grammar grow *unnaturally* from leaves to root, unlike CS trees, which grow naturally from root to leaves. This was not merely a diagrammatic reversal but a fundamental conceptual misalignment incompatible with rooted binary trees, the corresponding tree type in CS (see Figure 2 which is a comparison of syntactic and CS trees).

Generative linguists eventually acknowledged the problem and proposed a "solution": to carry on as if nothing had happened. Though the innateness hypothesis caused philosophical tensions, the tree issue was more severe—arguably even a theoretical disaster. A formal response is now expected in an upcoming book co-authored by the leading figure in generative grammar and two computer scientists: *Mathematical Structure of Syntactic Merge: An Algebraic Model for Generative Linguistics* (MIT Press, August 2025): https://mitpress.mit.edu/9780262552523/mathematical-structure-of-syntactic-merge. The book draws on earlier papers (e.g., https://ling.auf.net/lingbuzz/_search?q=Marcolli) but does not cite my work—consistent with a long tradition in the field whereby scholars from certain geographies (especially female scholars from Eastern Europe) are systematically ignored. In other words, the authors appear to have recognized the tree problem independently, but since *Syntactic Structures* (1957), it has been conveniently overlooked.

In contrast to older generative trees (see Syntactic tree in Figure 2), the new algebraic model adopts CS-compliant trees that grow from the root (node #1) to the leaves, see CS tree in Figure 2. Yet, despite its rigor, the book will likely be inaccessible to most linguists—especially Types 1 and 2—because of its mathematical sophistication. Unsurprisingly, there is already resistance among generative linguists to embracing the revised trees.

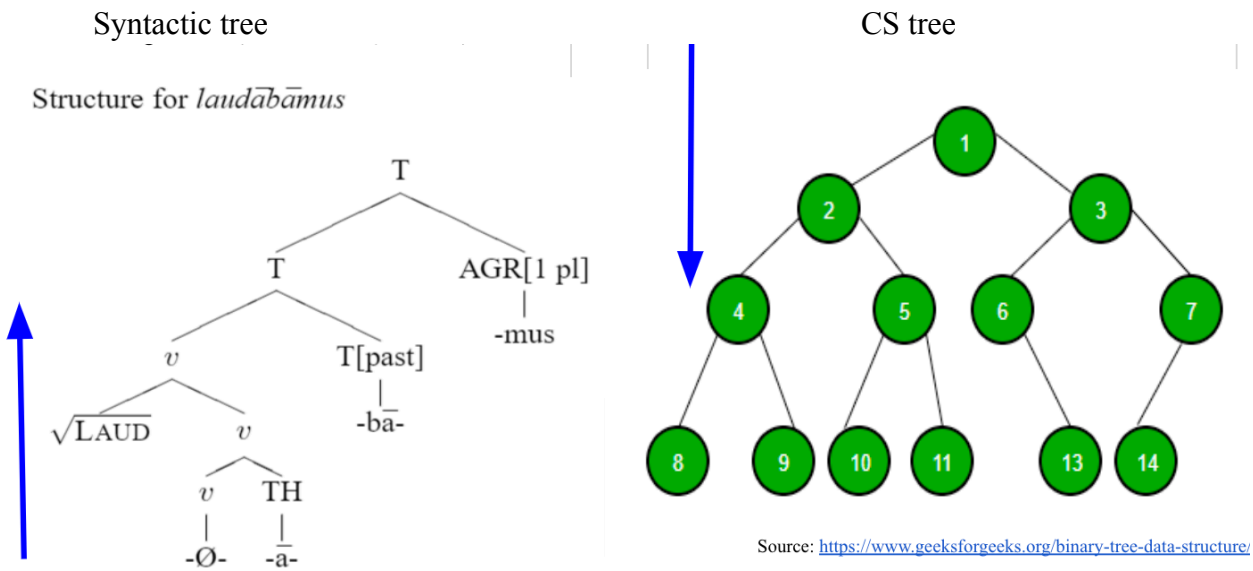

Figure 2: Rooted binary trees in linguistics and in CS, with a direction of growth, the blue arrows

Some generative linguists have attempted to relate LLMs to syntax and semantics using the old, flawed tree model. Their reasoning goes something like this: If LLMs can perform grammatical and semantic tasks—and even draw syntactic trees—then such trees and structures must be somehow embedded in the model. Linguists conclude: *Yes*. Mathematicians and computer scientists respond: *No*.

The reason for the different conclusions is straightforward: if an LLM can draw a tree that grows from leaves to root, it's not because this structure is part of the model's internal grammar—but because such representations appeared in its training data. (The newer root-to-leaf structures are rare in existing research, so ChatGPT likely cannot reproduce them yet.)

In short, LLMs can *imitate* linguistic analysis because they *saw* linguistic analyses—trees and all—during training. This explains both their surprising linguistic behavior and their occasional errors.

### *From Generative Grammar to Usage-Based Linguistics (and Back?)*

There has been premature celebration in the usage-based linguistics camp over having finally overtaken generative grammar. But now it seems that usage-based linguistics is about to face the same kind of theoretical upheaval that generative grammar has been undergoing in recent years.

Interestingly, generative grammar—despite its many flaws—shares with ChatGPT a key structural feature: the separation of form and meaning. Usage-based approaches, by contrast, typically treat form and meaning as inseparable. This distinction becomes highly relevant when trying to reconcile usage-based models with LLM output—and it is unclear to me why such a central issue has remained unaddressed by usage-based linguists. Perhaps this, too, stems from the widespread assumption in linguistics that *tokens are words* and *words are tokens*.

As for how the relation between form and meaning is treated in *Mathematical Structure of Syntactic Merge: An Algebraic Model for Generative Linguistics*—and how that relates to ChatGPT—I refer here to an earlier text by the same authors: *Syntax–Semantics Interface: An Algebraic Model*:

> We propose the following fundamental distinction between the roles of syntax and semantics in language
>
> • Syntax is a computational process.
>
> • Semantics is not a computational process and is in essence grounded on a notion of topological proximity.
>
> The first statement is clear in the context of generative linguistics, and in particular in the setting of Minimalism, where the computational process is run by the fundamental operation Merge. The second assertion requires some contextual clarification. Saying that semantics is only endowed with a notion of proximity of a topological nature does *not* mean that it is not possible, or desirable, to consider models of semantics where additional structure is present, but rather that these additional properties (metric, linear, semiring structures, for instance) only play a role to instantiate or quantify proximity relations. The compositionality of semantics does not require positing an additional computational structure on semantics itself: the computational structure of syntax suffices to induce it. In this view, **semantics is not really a part of language itself**, but rather an autonomous structure that deals with proximity classifications.
>
> https://ling.auf.net/lingbuzz/007696, p. 4, emphasis mine

This formulation aligns surprisingly well with how LLMs process language—especially when viewed through the lens of *attention mechanisms* (see: https://arxiv.org/abs/1706.03762 or the MIT 6.S191 video above: Recurrent Neural Networks, Transformers, and Attention (2025), https://www.youtube.com/watch?v=GvezxUdLrEk). If semantics is fundamentally spatial (topological) and syntax computational, then LLMs may well mimic the latter while approximating the former—without representing either in the traditional linguistic sense.

***Final Notes: The Case of 'Token'***

To be fair, *token* is also a problematic term in CS, where even seasoned engineers sometimes use *token* and *word* interchangeably in public explanations. They also tend to give examples consisting solely of common words, further fueling misunderstanding. But while this slippage is tolerable for computer scientists, it can seriously mislead linguists.

So what *should* we call a GPT token in linguistics?

I previously proposed the term *subword unit*. Another option is *GPT token*, which I use throughout this text to maintain clarity.

## 4. Why Do LLMs Need Such a Huge Amount of Data?

The mathematical explanation for why LLMs require massive amounts of data is fundamentally different from the explanations typically offered in linguistics. Importantly, this explanation does not support either of the two dominant theoretical camps evaluated in this text:

- **Usage-based linguistics**, which generally relies on corpora, or
- **Generative linguistics**, which tends to work with isolated examples.

To keep this text within reasonable bounds, I will develop the full mathematical account in a separate paper. However, I do want to use this section to connect with earlier discussions and to further illustrate how terminological confusion arises in linguistics.

Here is a simple test:

Can you, as a linguist, guess what *vLLM* stands for?

> ***In CS:***
>
> "vLLM, which stands for *virtual large language model*, is a library of open-source code maintained by the vLLM community. It helps large language models (LLMs) perform calculations more efficiently and at scale.
>
> Specifically, vLLM is an inference server that speeds up the output of generative AI applications by making better use of GPU memory." (italic emphasis mine)

Source:
https://www.redhat.com/en/topics/ai/what-is-vllm#:~:text=vLLM%2C%20which%20stands%20for%20virtual,maintained%20by%20the%20vLLM%20 community

> ***In Linguistics:***
>
> *vLLMs* stands for "very large language models."

Source:
https://www.plus.ac.at/news/antrittsvorlesung-univ-prof-mag-susanne-wurmbrand-phd/

This discrepancy is no accident—it mirrors the misinterpretations of *tree* and *token* in linguistics. Whatever "very large language models" is supposed to mean, this reinterpretation of the abbreviation *vLLMs* betrays a fundamental lack of understanding of how data are handled in CS. In CS, it is the **structure of the data** that determines how much data is needed to model it. Let me illustrate this with a simple classification of data along two dimensions: **order** and **regularity**.

(3) **Ordered Irregular Data**
a. {1, 2, 3, 7, 13}
b. {aa, bbb, c}

Evaluation:

- **Order**
  - {1, 2, 3, 7, 13}: Numerals are in ascending order.
  - {aa, bbb, c}: Strings are alphabetically ordered.

- **Regularity**
  - {1, 2, 3, 7, 13}: No consistent pattern; the gaps vary.
  - {aa, bbb, c}: No consistent format; letter groupings change.

(4) **Ordered Regular Data**
a. {2, 4, 6, 8, ...}
b. {a, aaa, aaaaa, ...}

Evaluation:

- **Order**
  - {2, 4, 6, 8, ...}: Ascending order.
  - {a, aaa, aaaaa, ...}: Ordered by length, starting with the shortest.

- **Regularity**
  - {2, 4, 6, 8, ...}: Constant difference of 2 between elements.
  - {a, aaa, aaaaa, ...}: Pattern increases by consistent 'aa' segment.

Only **ordered and regular** data can be **modeled with finite resources** to represent **infinite growth** (as signaled by the ellipsis '…'). That is, such data can be extrapolated without needing to observe every possible case.

By contrast, **language data are:**

- **Not ordered**
  - Letters in a word are not alphabetically ordered.
  - Word order in sentences varies across and within languages.
  - Sentences in a text are not arranged by any formal sequence.

- **Not regular**
  - Irregularities abound, e.g.:
    - *cat* → *cats,* but *mouse* → *mice*
    - *walk* → *walked*, but *go* → *went*
  - Syncretisms and exceptions disrupt any uniform patterns.

In short, **language data are unordered and irregular**, making them fundamentally resistant to efficient compression or generalization. This is the real reason why LLMs:

- Require massive amounts of training data
- Need deep, multi-layered architectures
- Must be updated regularly to keep up with change
- And still *hallucinate*—because no partial sample can capture the full irregularity of a changing linguistic system

Unlike mathematical sequences or structured datasets, human language cannot be modeled accurately without exposure to vast and ever-growing corpora. A "complete" dataset is impossible because language evolves continuously.

## 5. Conclusions

Linguists have a long-standing tendency to misinterpret CS terminology, a tradition that has persisted for decades and finds its most striking expression in the terms *tree* and *token*.

At present, there is no *linguistic* solution to the problem of fluent language generation. The only functioning solution is a *computational* one—large language models (LLMs) such as ChatGPT. These models operate not with words or morphemes, but with *tokens*: statistically defined, context-sensitive character sequences. These tokens are not linguistic units in any traditional sense—they do not align with any known category such as word, morpheme, or phoneme.

Each GPT token is assigned a unique ID number. ChatGPT processes only these IDs—it "sees" no words, no meanings, and no syntactic trees. This is why some computer scientists aptly describe LLMs as systems that model *language without language*.

For linguistic clarity, I suggest referring to these units as *subword units* or more specifically as *GPT tokens*—a term I have used throughout this paper.

Finally, regarding the ongoing debate between usage-based and generative linguistics: based on the available evidence, I find no grounds to declare one framework superior to the other. Both face significant challenges in explaining or integrating the architecture and performance of LLMs. The core issue, it seems, lies not in choosing sides between linguistic theories, but in recognizing the epistemological gap between linguistic assumptions and computational realities.

---

## Addendum

I have worked on all the issues addressed in the text above. You do not need to read my papers in order to understand LLMs, but you can, if you need additional explanations from a linguistic point of view:

- ➔ On binary code and what is wrong with syntactic trees: Section 6 in https://ling.auf.net/lingbuzz/006082
- ➔ Positional logic in mathematics / CS and in linguistics: https://www.researchgate.net/publication/339732483_What_is_in_a_morpheme_Theoretical_experimental_and_computational_approaches_to_the_relation_of_meaning_and_form_in_morphology
- ➔ On the complexity of form-based versus meaning-based morphological analyses, and on how the human brain treats two suffix combinations such as *-lessness* (namely, as a single unit), with data from the 100 K tokenizer: https://ling.auf.net/lingbuzz/008600
- ➔ On basic units and form-meaning mapping in human and machine learning, with data from the 200 K tokenizer: https://ling.auf.net/lingbuzz/008548

➔ Linguistic theory, psycholinguistics and large language models: https://ling.auf.net/lingbuzz/008123

**Currently in preparation:**

- The mathematical truth about why ChatGPT needs a huge amount of data

- How can generative grammar overcome its current crisis? (I am looking for generative linguists willing to collaborate with me on this. My solution completely differs from that in https://mitpress.mit.edu/9780262552523/mathematical-structure-of-syntactic-merge.)