

# Chapter 6: ChatGPT and linguistic theory, with a focus on morphology

Stela Manova

## 1. Introduction

With the rise of Large Language Models (LLMs), in Linguistic Theory (LT) much attention has been paid to whether LLMs can serve as theories of language (Piantadosi 2023 and replies to other scholars in it). However, the discussion has been one-sided: only linguists have participated in it (Chomsky et al. 2023, Haider 2023, Katzir, 2023, Moro et al., 2023, Rawski and Baumont 2023, Sauerland, 2023, among others) and consequently the approach adopted has been an abstract one, with an emphasis on semantics and syntax, though LLMs understand and generate language based exclusively on form. The technical organization of LLMs and the question of what this organization means for LT have remained unaddressed. The goal of my contribution is to provide the missing pieces. Since algorithms in different LLMs may differ, I tackle ChatGPT.

A GPT (an abbreviation for Generative Pre-trained Transformer) is a type of a LLM based on an artificial neural network (transformer architecture) and pre-trained on large data sets of unlabeled text, i.e. a GPT does not use grammar of the type known from LT.

ChatGPT was launched by OpenAI on November 30, 2022 and, as its name indicates, it is a LLM chatbot. It has a user-friendly interface and has been additionally trained for dialogue with humans. The most surprising feature of ChatGPT from a linguistic point of view (because ChatGPT can accomplish non-linguistic tasks as well) is its ability to quickly understand prompts and generate human-like texts in a real-time chat. The chatbot is fluent in a number of languages and it does not operate with words but with subword units (tokens) that are not associated with semantics (on meaning-form mapping in theoretical and computational approaches in morphology, see Manova et al. 2020). Roughly, in ChatGPT semantics emerges effortlessly as an effect of the use of long sequences of tokens. The NVIDIA website provides maybe the best explanation of how form and meaning relate in a LLM:

“A transformer model is a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence.”

<https://blogs.nvidia.com/blog/what-is-a-transformer-model/>.

According to the OpenAI website, one token is approximately four characters in English, which is approximately three-quarters of a word; more information on tokens and text length can be found at: <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>. The fact that ChatGPT understands and generates language based on subword units is the explanation for why this chapter is with an emphasis on morphology.

ChatGPT can accomplish complex semantic and syntactic tasks (Haider 2023) but since it is not a word-based model, it faces significant difficulties with the solution to a very basic linguistic task such as word counting, which makes the chatbot's quickness and fluency during a real-time chat a mystery from a linguistic point of view. Another perplexing fact concerns ChatGPT's vocabulary, `cl100k_base`. Although ChatGPT has been trained on

huge amounts of texts, its vocabulary is fixed-size and contains only 100k tokens (subword units). The list of all ChatGPT tokens (from now on I will refer to it as ‘the List’), is available from the following GitHub page:

[https://github.com/kaisugi/gpt4\\_vocab\\_list/blob/main/cl100k\\_base\\_vocab\\_list.txt](https://github.com/kaisugi/gpt4_vocab_list/blob/main/cl100k_base_vocab_list.txt) (last accessed on July 1, 2024). As I will demonstrate later, the List can be verified with the ChatGPT interactive tokenizer (from now on: ‘the Tokenizer’). The interactive Tokenizer has been prepared and maintained by OpenAI and can be found on their webpage: <https://platform.openai.com/tokenizer>.

I return to all the above-mentioned issues below. In this chapter, I demonstrate that ChatGPT’s internal organization cannot be explained with a single linguistic theory but with a combination of insights from different theories. Since the focus of the text is on morphology, I pay special attention to Distributed Morphology (Halle and Marantz 1993, Harley and Noyer 1999, Embick and Noyer 2007, Bobaljik 2017) and Paradigm Function Morphology (Stump 2001, 2016, Stump and Finkel 2013, Bonami and Stump 2017). I claim that ChatGPT is compatible with LT and can serve as a theory of language.

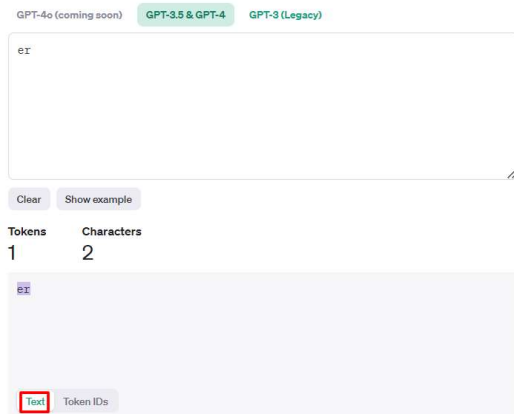
The chapter has the following structure. In section 2, I explain and illustrate tokenization, as well as how ChatGPT works with tokens, namely that when accomplishing a task the model processes language not orthographically but in terms of token IDs. Section 3 is devoted to the notion of complexity: I explain why form-based language processing (the case of ChatGPT) is less complex than semantics-based language processing (the case of LT). In LT of the two morphological components word-formation and inflection, the former has always been seen as semantically-driven, therefore in section 4 I demonstrate a form-based analysis of word-formation and provide psycholinguistic evidence that this is how native speakers process morphology. Section 5 accommodates the discussion, i.e. in this section I explain, among other things, why ChatGPT can serve as a theory of language. In the last section conclusions are drawn.

## 2. Words versus tokens

Words are the central units of analysis in LT, yet ChatGPT does not work with words but with tokens (subword units). A token can be represented as a text (letters), see the red quadrangle at the bottom of Figure 6.1, or as a token ID, which is a number, see the red quadrangle at the bottom of Figure 6.2. Thus, in the Tokenizer *-er* / ‘er’ has the token ID [261], Figure 6.2. From a technical point of view, a computer works with numbers faster than with letters, therefore in an LLM, a prompt is first tokenized (at this stage tokens are text), then encoded into token IDs, which are numbers, and only afterwards processed. When the task stated in the prompt is solved, the ChatGPT’s answer is a sequence of IDs, and it is then decoded into text. ChatGPT generates sequences of forms token-by-token and predicts the next token based on the statistics of co-occurrence of tokens in training data.

Most of the tokens coincide with linguistic units, as is the case with ‘er’ / *-er* (Figure 6.1 and Figure 6.2), which coincides with the English suffix *-er*, as in *read-er*, *open-er*, etc. As can be seen from the List, some of the tokens are letters, others are morphemes, still others are words, and there are also tokens that, according to LT, are not linguistic units, see, e.g., #257, #258, #261, and #263 in Figure 6.3 which is a screenshot of a segment extracted from the List. These four tokens are simply free spaces of different lengths.

Figure 6.1: -er (token as Text)



Source: <https://platform.openai.com/tokenizer>

Figure 6.2: -er (token ID)

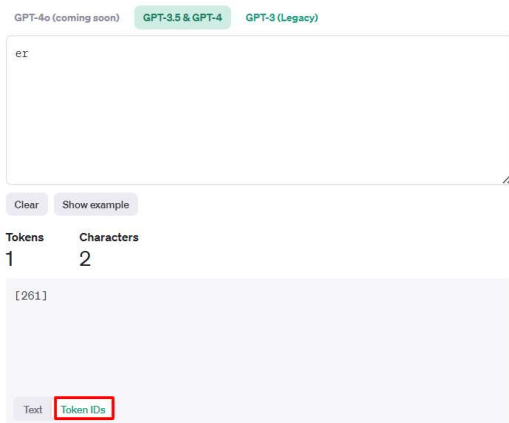


Figure 6.3. Tokens that are not linguistic units

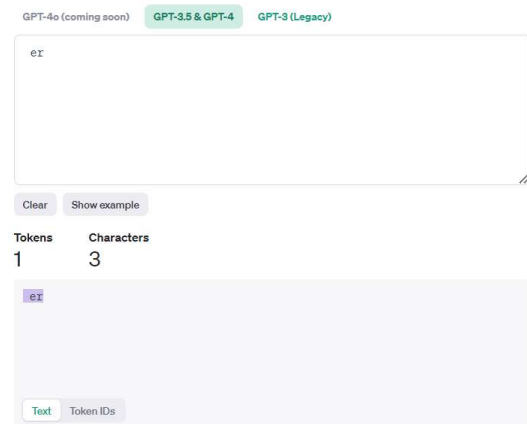
257	' '	' '
258	' '	' '
259	'in'	
260	't'	
261	' '	' '
262	'er'	
263	' '	' '

Source: [https://github.com/kaisugi/gpt4\\_vocab\\_list/blob/main/cl100k\\_base\\_vocab\\_list.txt](https://github.com/kaisugi/gpt4_vocab_list/blob/main/cl100k_base_vocab_list.txt)

For ChatGPT a free space is a meaningful character. Therefore, 'er' / -er (Figures 6.1, 6.2) differs from ' er' / er- (Figures 6.4, 6.5) and consequently these two tokens have different IDs, [261] (Figure 6.2) and [2781] (Figure 6.5), respectively. The differentiation of the two

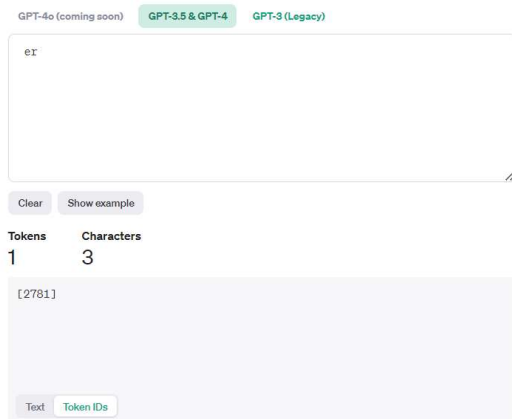
tokens is of particular importance because [2781] with the free space before it indicates a word beginning. This is the explanation why 'er' [261] in Figures 6.1, 6.2 consists of two characters, while ' er' [2781] in Figures 6.4, 6.5 has three characters (i.e., the free space counts as a character). A smaller ID means marks a token with a higher frequency, i.e. 'er' / -er [261] is more frequent than ' er' / er- [2781].

Figure 6.4: er- (token as text)



Source: <https://platform.openai.com/tokenizer>

Figure 6.5: er- (token ID)



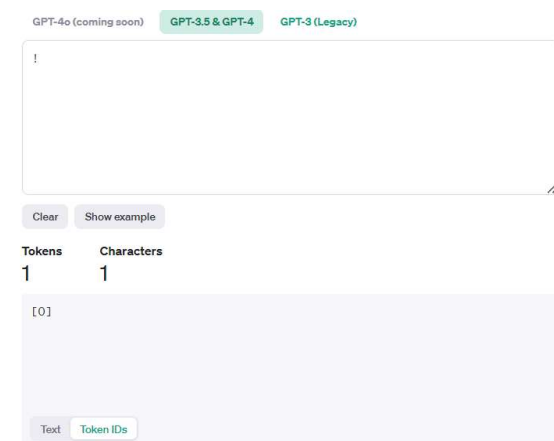
Tokenization makes possible the representation of a large amount of text with a small set of tokens, the latter are comparable with letters, morphemes and highly frequent words in LT. Now, if tokenization is so important, why is *er-* in the List #262, but in the Tokenizer its ID is [261]. This difference results from the fact that computers, unlike humans, start counting from [0], i.e. the Tokenizer starts from zero, while the List follows the human way of counting and starts from one. Therefore, #1 in the List, '!', has the token ID [0] in the Tokenizer.

Regarding vocabulary length, ChatGPT's vocabulary is declared to be of 100k tokens, yet the List contains 100256 entries. This is because we need items for vocabulary building, i.e. the first 256 items serve for initialization (building of the rest) of the vocabulary.

Figure 6.6: Token #1 in the List

1	'!'
2	'"'
3	'#'
4	'\$'
5	'%'

Figure 6.7: Token ID [0] in the Tokenizer



For tokenization, ChatGPT uses tiktoken (<https://github.com/openai/tiktoken>), a fast Byte Pair Encoding (BPE) algorithm (Sennrich et al. 2016). An easy-to-understand explanation of the BPE logic can be found at: <https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp/>. The most important steps of the algorithm, extracted from the explanation at this link, are listed in (1). For the sake of clarity, I expanded the descriptions of some of the algorithm's steps with additional information.

Now let me illustrate tokenization in practice: Imagine that we have a text corpus that should be tokenized and that this corpus consists of four words, as shown in (1):

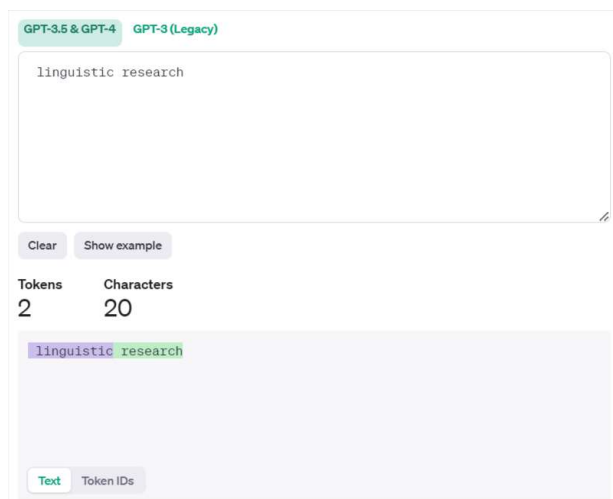
- (1) Byte Pair Encoding (BPE)  
Text corpus: "**ab**", "**bc**", "**bcd**", "**cde**" (i.e., the corpus consists of four words)
  - a) Step 1: Initialize the vocabulary (i.e., establish all characters (bytes) used in the corpus)  
Vocabulary = {"a", "b", "c", "d", "e"}
  - b) Step 2: Calculate the frequency of each character  
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}
  - c) Step 3a: Find the most frequent pair of two characters (in the same position)  
The most frequent pair in the initial position is "bc", with a frequency of 2.

- (“cd” does not count because in “bcd” it is in a final position, while in “cde”, it is in an initial position)
- d) Step 3b: Merge the pair  
Merge “b” and “c” to create a new subword unit “bc”.
  - e) Step 3c: Update frequency  
Update the frequency count of “bc”:  
Frequency = {“a”: 1, “b”: 2, “c”: 3, “d”: 2, “e”: 1, “bc”: 2}
  - f) Step 3d: Add the new subword unit to the vocabulary  
Add “bc” to the vocabulary:  
Vocabulary = {“a”, “b”, “c”, “d”, “e”, “bc”}  
Repeat steps 3a-3d until the desired vocabulary size is reached.<sup>31</sup>

As already mentioned, the ChatGPT vocabulary is limited to 100k. In other words, the frequency of the 100k-th token serves as a threshold for vocabulary inclusion. All items that are part of the vocabulary count as single tokens and are therefore considered non-derived, which makes ‘linguistic’, as in *linguistic research*, a single token, i.e. a non-derived item (*research* is also a single token), Figures 6.8, 6.9; but ‘linguistic’ in *crosslinguistic* is a derived item, Figures 6.10, 6.11. For the sake of completeness, Figures 6.13, 6.12 demonstrate that spelling (hyphenated versus together) does not influence the parsing of ‘linguistic’.

Because of the hyphen, see [12] in Figure 6.13, *crosslinguistic* and *cross-linguistic* differ in a character, 16 and 17 characters, respectively, Figures 6.10, 6.11 and Figures 6.12, 6.13.

Figure 6.8: ‘linguistic research’ (Text)



<sup>31</sup> A Python implementation of the BPE can be found at the bottom of: <https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp/>. Further useful information on tokenization at: [https://cookbook.openai.com/examples/how\\_to\\_count\\_tokens\\_with\\_tiktoken](https://cookbook.openai.com/examples/how_to_count_tokens_with_tiktoken). An alternative Tokenizer is available as a web app at: <https://tiktokenizer.vercel.app/>. With this app, the reader can experiment with the tokenization strategies employed by different models.

Figure 6.9: 'linguistic research' (IDs)

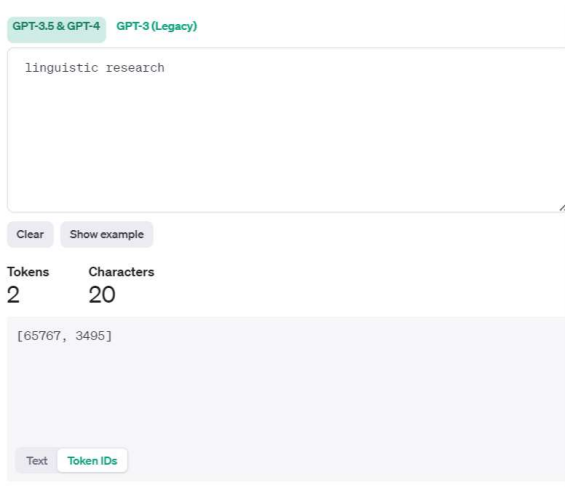


Figure 6.10: 'crosslinguistic' (Text)

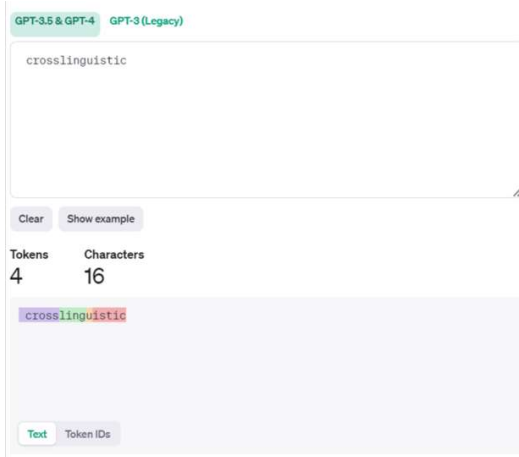


Figure 6.11: 'crosslinguistic' (IDs)

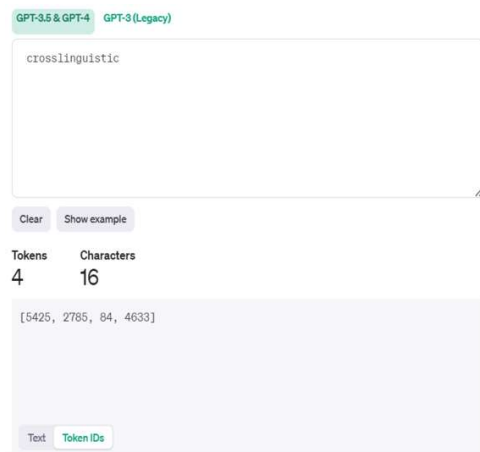


Figure 6.12: ‘ cross-linguistic” (Text)



Figure 6.13: ‘ cross-linguistic” (IDs)



In sum, ChatGPT works with tokens (subword units), the latter can be letters, morphemes or highly frequent words; there are also tokens that do not coincide with linguistic units, yet most tokens do. Tokens are not associated with semantics. Language is a long uninterrupted sequence of tokens in which spaces indicate word beginnings. Depending on position and frequency of use in this position, the same sequence of letters may have different IDs or may be treated as both non-derived and derived. A smaller ID indicates a higher frequency. ChatGPT does not process text but IDs.

I am aware that for a linguist it is somehow hard to believe that ChatGPT does not work with words but with tokens. So as an illustration of the issue, I gave the task in (2) to ChatGPT. I asked the chatbot to count repeated words in a text (pay attention to the length of the prompt, namely that it is not a single word but a sentence). The text used for word counting is the abstract of one of the versions of Piantadosi (2023).

(2) Counting repeated words with ChatGPT:



**Prompt:** List and count words used more than once in the following text:

"The rise and success of large language models undermines virtually every strong claim for the innateness of language that has been proposed by generative linguistics. Modern machine learning has subverted and bypassed the entire theoretical framework of Chomsky's approach, including its core claims to particular insights, principles, structures, and processes. I describe the sense in which modern language models implement genuine theories of language, including representations of syntactic and semantic structure. I highlight the relationship between contemporary models and prior approaches in linguistics, namely those based on gradient computations and memorized constructions. I also respond to several critiques of large language models, including claims that they can't answer "why" questions, and skepticism that they are informative about real life acquisition. Most notably, large language models have attained remarkable success at discovering grammar without using any of the methods that some in linguistics insisted were necessary for a science of language to progress."

**ChatGPT:** To list and count words used more than once in the provided text, I'll go through the text and tally the occurrences of each word, disregarding case and punctuation.

Here is the processed count of words used more than once:

**language:** 7 times  
**models:** 4 times  
**large:** 3 times  
**claims:** 3 times  
**of:** 9 times  
**and:** 5 times  
**the:** 8 times  
**linguistics:** 3 times  
**including:** 3 times

In this counting, there are a number of mistakes and I asked ChatGPT to recount the results for 'the', 'models', and 'claims', in a separate session for each item. The whole conversation is available at: <https://chatgpt.com/share/57abf75d-5a05-4686-b603-741d1da36b5c>.

For the recounting of 'the', I tried to confuse the chatbot suggesting that 'the' is used seven times. ChatGPT replied correctly that 'the' is repeated six times, see the count of 'the' in (4).

Each time ChatGPT could understand the prompt and replied to it appropriately but the word counts were not always correct. When I insisted on an explanation of why it makes mistakes in word counting, ChatGPT replied the following.

(3) Recounting of repeated words:

**ChatGPT:** I apologize for the earlier mistakes in counting words. Let's approach this systematically using a Python script to ensure we get the accurate counts for words used more than once.

Here's the Python code to count word occurrences:

```
text = """The rise and success of large language models undermines virtually every strong claim for the innateness of language that has been proposed by generative linguistics. Modern machine learning has subverted and bypassed the entire theoretical framework of Chomsky's approach, including its core claims to particular insights, principles, structures, and processes. I describe the sense in which modern language models implement genuine theories of language, including representations of syntactic and semantic structure. I highlight the relationship between contemporary models and prior approaches in linguistics, namely those based on gradient computations and memorized constructions. I also respond to several critiques of large language models, including claims that they can't answer `why` questions, and skepticism that they are informative about real life acquisition. Most notably, large language models have attained remarkable success at discovering grammar without using any of the methods that some in linguistics insisted were necessary for a science of language to progress."""
```

```

# Removing punctuation and converting to lowercase
import re
from collections import Counter
words = re.findall(r'\b\w+\b', text.lower())
word_counts = Counter(words)
# Filter words that appear more than once
repeated_words = {word: count for word, count in word_counts.items() if count > 1}
repeated_words

```

Explanation of the code:

**#** indicates a comment.

**r'\b\w+\b'** is a regular expression used to match words in a string.

**r'** denotes a raw string, which means that the backslashes (**\**) are treated as literal characters and not as escape characters.

**\b** denotes a word boundary, a position where a word character is not followed / preceded by another word character.

**\w** matches any word character.

**+** means "one or more" preceding element, i.e. **\w+** matches sequences of one or more word characters.

#### (4) Running the Python script from (3):

**ChatGPT:** Let's run this script to get the accurate counts.

Here's the accurate count of words used more than once in the provided text:

**the:** 6 times  
**and:** 7 times  
**success:** 2 times  
**of:** 8 times  
**large:** 3 times  
**language:** 7 times  
**models:** 5 times  
**for:** 2 times  
**that:** 4 times  
**has:** 2 times  
**linguistics:** 3 times  
**modern:** 2 times  
**including:** 3 times  
**claims:** 2 times  
**to:** 3 times  
**i:** 3 times  
**in:** 3 times  
**they:** 2 times

Now, the number of repetitions doubled, from 9 in (2) to 18 in (4).

In sum, linguists have evaluated ChatGPT's linguistic competence with complex semantic and syntactic tasks and, as a rule, ChatGPT performs on such tasks well, see, e.g., Haider

(2023). To the best of my knowledge, no linguist has challenged ChatGPT with a basic linguistic task such as word counting. Since ChatGPT does not process language in terms of words, it is miserable in word counting. To count words properly, ChatGPT needs additional software (the Python script in (3)). In another session on word counting, ChatGPT explained to me that if I wanted to avoid mistakes in word counting, my prompt should have required first a tokenization of the text, so that ChatGPT could compose words from the tokens and with the help of the free spaces preceding some of the tokens, it would then count words correctly; recall the discussion about the difference between 'er' (Figures 6.1, 6.2), and ' er' (Figures 6.4, 6.5), namely that ' er' indicates a word boundary (word beginning), marked by '\b' in the Python script in (3).

After the difficult solution of the word counting task, it could seem that ChatGPT handles language in a very complex way, more complex than what is usually done in a linguistic analysis. However, this is a misleading impression and the next section explains why.

### 3. Complexity

In science, a problem often allows for different solutions. The so-called *Big O* notation serves for assessment of the complexity of those solutions in mathematics and computer science (CS).<sup>32</sup>

The *Big O* notation tells us how an algorithm slows as data grow. That is, complexity is not a property of data (which is the case in linguistics, but of the algorithm (analysis)). As an illustration of this logic, let me evaluate two solutions of a task. However, it has to be mentioned here that the examples below are to help linguists understand the concept of complexity and this explanation of the *Big O* notation is an oversimplification. In computer science, the *Big O* notation evaluates the complexity of functions and data structures (e.g., *tree* search is more complex than *list* search, cf. hierarchical syntactic trees (in LT) and linear sequences of tokens (in ChatGPT)).

(5) *Problem*: Calculate the sum of the numbers from 1 to 100.

(6) *Solution 1*:  $1+2+3$ , and so on to 100, i.e. 99 summations are necessary to calculate the sum.

Let us check the behavior of this solution as data grow, e.g. let us increase the amount of the data from 100 to 1000. Following the same idea (Solution 1), to calculate the sum of the numbers from 1 to 1000, we have to perform 999 summations (algorithm steps). That is, with the growth of the data, more effort is required to come to a solution.

(7) *Solution 2*: Based on the observation made by the young Gauss that  $100+1 = 99+2 = 98+3$ , and so on to  $51+50$ .

We can calculate the sum of the numbers from 1 to 100 only in two steps: the first step involves addition, the second step consists in multiplication:  $(1+100)*50=5050$ . An increase of the amount of the data from 100 to 1000 does not reflect on the complexity of the solution and we can still calculate the sum of the numbers from 1 to 1000 in two steps:  $(1+1000)*50= 500500$ .

---

<sup>32</sup> A tutorial on the *Big O* at: <https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/>.

Both Solution 1 and Solution 2 give the same result, but the first solution is complex and therefore uninteresting, while Gauss's solution is simple and elegant and has been used as a formula for the sum of an arithmetic progression ever since.

How does all this relate to ChatGPT and research in linguistics? The ChatGPT approach to language relies on surface forms (orthography), Rule 1 in (8); while a linguistics approach usually relies on semantics, Rule 2 in (9).

- (8) Rule 1, form-based: If a word A (e.g., *nivo* 'niveau') ends in -o, attach the suffix B (e.g., -to in Bulgarian) to it.
- (9) Rule 2, semantics-based: If X is a particular type of a verb (e.g., an action verb), derive a particular type of a noun Y (e.g., an agent) by the attachment of the productive suffix Z (e.g., -er).

Now, the information on which Rule 1, (8), relies is not language-specific and is directly available: the word A either terminates in -o or not (*nivo* ends in -o), and the attachment of the suffix B gives *nivoto* 'the niveau'. The semantic information on which Rule 2, (9), relies requires additional effort in order to be discovered and Rule 2 is also language-specific, in the sense that we need some knowledge of the language from where the data come in order to apply this rule. Then, Rule 1 consists of two steps: i) we have to check whether A ends in -o and if yes, ii) to attach the suffix B (-to). Rule 2 involves the following steps: a) evaluation whether the word from the initial set is a verb; if yes, b) we have to ensure that the verb is of the type we need (an action verb), afterwards c) addition of the productive suffix -er to derive an agent noun, if d) the derivation is possible, because e.g. *to edit* is an action verb but it does not co-occur with -er (moreover, according to LT *to edit* is a backformation from *editor*, Manova, 2011a). Therefore, we conclude that Rule 2 is more complex than Rule 1.

Before moving to Section 4, in which I demonstrate a form-driven analysis of word-formation, let me illustrate a form-driven inflection (like in (8)) with more data from my mother tongue Bulgarian (Slavic), (10) and (11), see also Manova (2023a). Bulgarian has a suffixal definite article and indefinite nouns and adjectives in this language may end in -a. If semantics is considered, there should be four different -a morphemes, cf. the morphosyntactic feature values in (10) and (11) where all -a morphemes are bolded and indexed for convenience. The four different -a morphemes all select the definite article -ta (Manova and Dressler, 2001), though the article has allomorphs, see (1d) and the example in (8).

- (10) Nouns: indefinite → definite  
 sg.fem: *bluz-a<sub>1</sub>* 'blouse' → *bluz-a<sub>1</sub>-ta* 'the blouse'  
 sg.masc: *bašt-a<sub>2</sub>* 'father' → *bašt-a<sub>2</sub>-ta* 'the father'  
 pl.neut: *sel-a<sub>3</sub>* 'villages' → *sel-a<sub>3</sub>-ta* 'the villages'  
 cf. sg.neut: *sel-o* 'village' → *sel-o-to* 'the village'
- (11) Adjectives: indefinite → definite  
 sg.fem: *krasiv-a<sub>4</sub>* 'beautiful' → *krasiv-a<sub>4</sub>-ta* 'the beautiful'

#### 4. A form-based analysis of word-formation

As mentioned, (10) and (11) are inflectional morphology and it is well-known from LT that inflection produces word forms and therefore may be accounted for without reference to meaning. By contrast, word-formation, since deriving new words, has always been seen as a semantic operation. In other words, if derivation without reference to meaning is less complex than derivation with reference to meaning, it should also be possible to account for word-formation only in terms of relations of forms, i.e., to provide an analysis of word-formation similar to how ChatGPT generates language. In this section, I demonstrate that such an analysis is indeed possible and that actually this is how native speakers combine word-formation suffixes.

Undoubtedly, English is the language with the most profoundly studied word-formation. (Overviews of research on word-formation from a cross-linguistic perspective in Lieber and Štekauer, 2014; Plag and Balling, 2016; and Lieber, 2017). While more recent studies analyze English word-formation based primarily, if not exclusively, on semantics (Lieber, 2004, among many others), previous research known as the *Stratal approach* (Siegel, 1974; Selkirk, 1982; Kiparsky, 1982) is form-focused, see (12): based on morphological and phonological information (the types of morpheme junction marked by '+' and '#' respectively) forms of affixes are distributed into different strata (classes) so that class II affixes are always outside class I affixes in the word-form.

- (12) English: Stratal approach, from Spencer (1991:79)
- a. Class I suffixes: *+ion, +ity, +y, +al, +ic, +ate, +ous, +ive, +able, +ize*
  - b. Class I prefixes: *re+, con+, de+, sub+, pre+, in+, en+, be+*
  - c. Class II suffixes: *#ness, #less, #hood, #ful, #ly, #y, #like, #ist, #able, #ize*
  - d. Class II prefixes: *re#, sub#, un#, non#, de#, semi#, anti#*

Another example of a form-focused analysis is Fabb (1988). This study distributes the English suffixes into four groups as shown in (13):

- (13) English: Suffix-driven selectional restrictions (Fabb 1988)
- a. Group 1: suffixes that do not attach to already suffixed words
  - b. Group 2: suffixes that attach outside one other suffix
  - c. Group 3: suffixes that attach freely
  - d. Group 4: problematic suffixes

An alternative form-focused analysis recognizes closing suffixes: a particular suffixal form cannot be followed by other suffixes in a language (see Szymanek 2000 for English and Polish; see also Aronoff and Fuhrhop 2002). Closing suffixes have been established in a number of languages. Manova (2015b) is an overview of research on the topic.

A form-based restriction on the order of English derivational suffixes of a different kind is reported in Manova (2011b) and Manova and Knell (2021). Manova (2011b) sees derivational suffix combinations as binary structures of the type SUFF1-SUFF2, where SUFF1 has three valency positions for further suffixation: SUFF2<sub>Noun</sub>, SUFF2<sub>Adjective</sub> and SUFF2<sub>Verb</sub>. The idea of this distribution of outputs according to the lexical-category specification of SUFF2 is based on the Gauss-Jordan elimination, a mathematical method for solving systems of linear equations numerically, that is, only with the help of elementary operations such as substitution, addition or multiplication. (14) is an example of a system of linear equations.

$$\begin{aligned}
 (14) \quad & 2x + y + 2z = 10 \\
 & x + 2y + z = 8 \\
 & 3x + y - z = 2
 \end{aligned}$$

The goal of Gauss-Jordan is, based only on well-known facts and elementary operations with them, to come to a single option for a variable (the unknown);  $x$ ,  $y$  and  $z$  are the variables in (14). If there is only one option for a variable, this option is the solution to the problem.

With respect to affix order in word-formation, the well-known information is information about the lexical category specification of an affix, i.e. whether the affix derives nouns (N), adjectives (A) or verbs (V); a single option for a variable means one affix combination of a kind. As can be seen from Table 6.1, this method allows linguistic data to be distributed so that in most cases there is indeed one option of a kind: for N (*-ist<sub>N-dom<sub>N</sub></sub>*) and for V (*-ist<sub>N-ize<sub>V</sub></sub>*). I label such combinations *fixed*.

Table 6.1: Combinability of the English suffix *-ist*

SUFF1	Lexical category of SUFF1	SUFF2 classified for lexical category; in brackets, number of types (lemmas) derived with the combination SUFF1-SUFF2	
<i>-ist</i>	N	N: <i>-dom</i> (2)	[ <i>fixed combination</i> ]
		A: <i>-ic</i> (631), <i>-y</i> (5)	[ <i>predictable combination</i> ]
		V: <i>-ize</i> (3)	[ <i>fixed combination</i> ]

Source: Previous research of mine; data from Aronoff and Fuhrhop, 2002, based on OED, CD, 1994.

If more than one SUFF2 of the same lexical category is available (see for A(djective) in Table 6.1), one of the SUFF2 suffixes attaches by default, suffix *-i<sub>CA</sub>* in our case: in English, the combination *-ist<sub>N-i<sub>CA</sub></sub>* derives 631 types, while *-ist<sub>N-y<sub>A</sub></sub>* derives only 5 types, which makes *-ist<sub>N-i<sub>CA</sub></sub>* a *predictable* combination.<sup>33</sup> Regarding default suffixes, having counted suffix combinations in large dictionaries and corpora for different languages, Manova (2011b, 2015a), Manova and Talamo (2015), and Manova and Knell (2021) maintain that a default suffix derives more than ten types (such suffixes are referred to as productive), while SUFF2 suffixes that compete with the default suffix derive ten or fewer types each. Thus, in table 6.1 different forms of suffix combinations correspond to different meanings (by default), which means a one-to-one mapping of meaning and form and allows us to identify the suffix combinations at hand both in terms of form (cf. the ChatGPT approach) and in terms of meaning (the LT approach).

Table 6.2 applies the logic of Gauss-Jordan to a more complex case, the combinability of the Polish suffix *-arz*.<sup>34</sup> Polish, unlike English, is an inflecting fusional language and derivational suffixes are often followed by inflection, i.e. in Polish the inflection is obligatory

<sup>33</sup> According to ChatGPT, both 'istic' and 'isty' are single tokens, [4633] and [88623], respectively, which confirms my reasoning.

<sup>34</sup> I thank Bartosz Brzoza for his help with the Polish data.

for the well-formedness of a word. All inflectional suffixes in Table 6.2 are in brackets. Descriptions and analyses of Polish derivational morphology by Polish scholars, as a rule, give derivational suffixes together with the inflection that follows them, either in brackets, as done in Table 6.2, or unmarked, as a single suffix with the derivational one, e.g., as *-n(y)* or *-ny*, respectively; see the first adjectivizing SUFF2 in Table 6.2. (For a semantics-based analysis of the combinability of Polish word-formational suffixes, see Burkacka 2015; see also the discussion of Polish word-formation in Szymanek 2010.)

As shown in Table 6.2, the suffix *-arz* combines with more than one adjectivizing SUFF2 and with a set of nominalizing SUFF2 suffixes. While for the derivation of adjectives, there is only one default suffix, *-sk(i)* (>10), three different nominalizing suffixes that derive more than ten types can follow the suffix *-arz*: *-czyk* (>10), *-ni(a)* (>10) and *-stw(o)* (>10), all bolded in Table 6.2 for convenience. The existence of three default suffixes of the same type (nominalizers) all “competing” for *-arz* seems to challenge my analysis. However, the three suffixes differ in both form and meaning: *-czyk* (>10), default for derivation of *persons*; *-ni(a)* (>10), default for derivation of *places*; and *-stw(o)* (>10), default for *abstract/collective nouns*. That is, neither homophony nor polysemy is involved in the suffix combinations in Table 6.2. I therefore conclude that all suffix combinations in Table 6.2 are predictable, and since they are instances of one-to-one mapping of form and meaning (by default), they can also be identified only in terms of form (the ChatGPT approach).

Considering the fact that derivational suffixes in English and Polish seem to form only fixed and predictable combinations, i.e., should be identifiable in terms of form, I hypothesized that native speakers should have memorized those combinations and, consequently, should be able to produce them without reference to meaning, that is, based exclusively on form. To test this hypothesis, I designed a psycholinguistic experiment the results of which are reported below. Here I present only the results of the native speakers of English and Polish, but the experiment was also conducted with native speakers of German, Italian, Spanish and Slovene, and with advanced non-native speakers of English and German. Overall, the results of all iterations converge. (For curious readers, the scores of the non-native speakers of English are reported in Manova and Knell 2021; the scores of the native and non-native speakers of German can be found in Brosche and Manova 2022).

Table 6.2: Combinability of the Polish suffix *-arz*

SUFF 1	Lexical category of SUFF1	Lexical category of SUFF2	SUFF1-SUFF2 exemplified	Notes
<b>-arz</b>	N	i. ADJ: <i>-n(y)</i> (2)	<i>moc-ar-n(y)</i> ‘strong’	[derives only 2 adjectives]
		ii. ADJ: <i>-ow(y)</i> (1)	<i>gęśl-arz-ow(y)</i> ‘of fiddler’	[derives a single adjective]
		iii. <b>ADJ: <i>-sk(i)</i> (&gt;10)</b>	<i>pis-ar-sk(i)</i> ‘of writer’	<b>[default for derivation of adjectives]</b>
		a. <b>N: <i>-czyk</i> (&gt;10)</b>	<i>piek-ar-czyk</i> ‘baker’s apprentice’	<b>[default for derivation of persons, cf. f]</b>

b. N: <i>-k(a)</i> (2)	mur- <i>ar-k(a)</i> 'bricklaying'	[derives only 2 abstract nouns, cf. e]
c. N: <i>-ni(a)</i> (>10)	kreśl- <i>ar-ni(a)</i> 'drafting studio'	[derives nouns for <b>places</b> ]
d. N: <i>-nik</i> (1)	piek- <i>ar-nik</i> 'oven'	[derives a single <b>object</b> ]
e. N: <i>-stw(o)</i> (>10)	księg- <i>ar-stw(o)</i> 'all booksellers'	[ <b>default abstract/collective nouns</b> , cf. b]
f. N: <i>-yn(a)</i> (5)	mur- <i>arz-yn(a)</i> 'bad bricklayer'	[derives only 5 nouns for persons, cf. a]

---

#### 4.1 Method

64 native Polish speakers and 45 native English speakers were tested, they all participated on a voluntary basis. The questionnaire presented to them consisted of three parts:

- A series of general demographic questions regarding age, gender, nationality, native language(s), other languages spoken, level of education, and experience in a linguistic or other language-related field.
- A small practice to ensure that the participants understood the task properly. The training examples were not part of the test stimuli.
- The main task: 60 suffix combinations (e.g. *-istic* in English, *-army* in Polish) were presented in a randomized order, and participants were asked to identify which of the combinations exist and which do not exist as word terminations in the respective language. Of the 60 combinations, 30 existed in the respective language and 30 did not. Of the existing combinations, 15 were productive (derive more than ten types) and 15 were unproductive. Of the non-existing combinations, 15 were created from a permutation of an existing combination (reversing the order of the two suffixes such that the combination was not possible in the respective language), and 15 were created through a spelling manipulation of an existing combination (changing one letter from an existing combination such that the new form does not exist in the particular language). No non-existing combinations included any phonological and/or orthographical impossibilities in the respective language.

#### 4.2 Data analysis

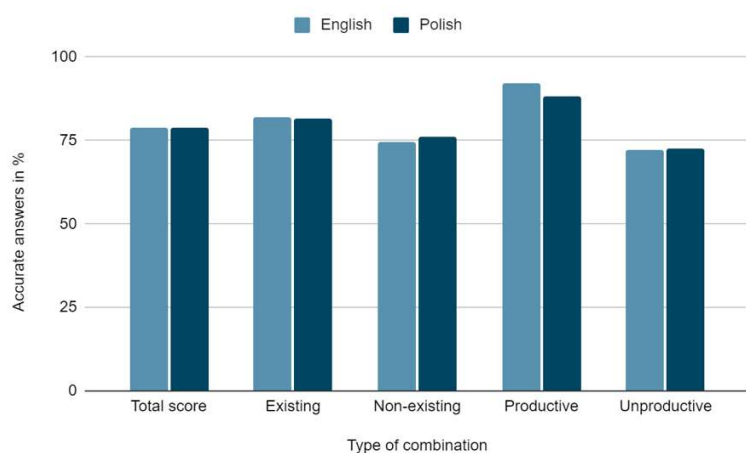
Independent t-tests were used to consider possible significance of overall scores, as well as for stimulus type: existing vs. non-existing and productive vs. unproductive combinations. Figure 6.14 presents the results of the native speakers of English and Polish.

The participants in the experiment did not need semantic cues to process suffix combinability, i.e. they could differentiate between existing and non-existing suffix combinations presented to them without lexical bases such as roots/stems/words. Statistically significant were the differences between existing and non-existing



combinations, and between productive and unproductive combinations. As already mentioned, English and Polish differ typologically, in the sense that English has very poor inflectional morphology, while Polish is characterized by a very rich inflectional system. Nevertheless, the results obtained for the two languages are virtually the same, the total score of the correct answers for English is 79% and 78.86% for Polish (Figure 1), though combinations of three suffixes (trigrams, the case of Polish where two derivational suffixes are often followed by inflection) should be easier to recognize than combinations of two suffixes (bigrams, the case of English derivational suffix combinations). In other words, inflection did not seem to have an impact on the processing on suffix combinability in word-formation. I therefore conclude that native speakers of Polish see inflection as forming a natural subword unit with the derivational material that precedes it; on this issue see also Manova (2023b).

Figure 6.14: Native speakers' accuracy of recognition of the 60 suffix combinations tested in the experiment (only statistically significant results).



Total score = correct answers for all types of suffix combinations tested: existing combinations of two types (productive and unproductive) and non-existing of two types (permutations and manipulations, see Method).

Since suffix combinability is not taught at school and all linguistic theories assume that a morphological derivation always starts with a root/stem, depending on the theory, the only plausible explanation why native speakers of English and Polish successfully accomplished a task they should not be able to solve is that they had extracted and subconsciously memorized adjacent suffixes in terms of bigrams<sup>35</sup> and trigrams, during language acquisition (cf. tokenization in ChatGPT).

## 5. Discussion

In this section I address the question of whether ChatGPT can serve as a theory of language. I claim that the answer to this question is “yes”, especially if the new theory is allowed to mix insights from different approaches to morphology. I discuss the morpheme-based Distributed Morphology (Halle and Marantz, 1993; Harley and Noyer, 1999; Embick and Noyer, 2007; Bobaljik, 2017) and the paradigm-based Paradigm Function Morphology (Stump, 2001, 2016; Stump and Finkel, 2013; Bonami and Stump, 2017). The former

<sup>35</sup> Analyses of affix order in terms of bigrams are proposed in Ryan (2010) and Mansfield et al. (2020).

denies the importance of the paradigm as an analytical tool, the latter does the same with the morpheme. Additionally, DM is syntax-centered, i.e. syntactic rules derive morphological structure, while PFM claims for independence of morphology from syntax. I therefore consider DM and PFM a good example of competing theories. Of course, the two theories also have things in common, e.g., they both are meaning-first. A DM derivation concatenates abstract morphemes that have meaning but no form. Such morphemes are placed in terminal nodes on the syntactic tree and the association with phonological form is postponed, the so-called *late insertion*. PFM, on its turn, assumes the existence of content paradigms (ordered paradigm cells identifiable in terms of sets of morphosyntactic features); the content paradigm is then linked to a corresponding form paradigm. (In some versions of the theory, the existence of a third paradigm called realizational is assumed, the form-paradigm copes with meaning-form mismatches, the realizational paradigm gives the final forms.)

Before starting with the theoretical discussion, let me mention some a-theoretical issues typical of language in general. ChatGPT, similar to human languages, has a set of letters and symbols that serve for vocabulary initialization, i.e. for production of larger elements, tokens or subword units (recall the List of 100256 tokens). A human language in a written form is organized in the same way, in the sense that it also possesses letters (an alphabet and also uses some additional symbols for numbers, etc.) and derives basic units (morphemes) as combinations of letters. Note that morphemes are, like tokens, subword units. Thus with respect to letters and basic units built from them, ChatGPT and human languages appear to be very similar, if not the same.

If we think of ChatGPT in the context of theories of language, a highly relevant issue is how the model learns vocabulary items. In ChatGPT, the process is entirely frequency-governed (cf. BPE): the frequency of occurrence of adjacent letters determines whether the letter combination they form will be added to the vocabulary or not. Can such a scenario serve for language acquisition (LA) by humans? It seems to me that the answer to this question should be “yes”, though, to the best of my knowledge, there is no research on the topic. Additionally, linguists should be aware that if LA is researched only in terms of morphemes and words, as done in LT, the conclusion could be that LA is, like ChatGPT, token-based.

Now on theoretical issues, the first significant difference between ChatGPT and LT consists in the fact that ChatGPT’s basic units, the tokens, are not associated with semantics, which thus makes ChatGPT incompatible with DM but in harmony with PFM where word-internal structure consists of markings that have only form and no meaning (recall that in PFM meaning is assigned only to paradigm cells).

Regarding the identification of the basic units, ChatGPT tokens can be morphemes and words, though letters that serve for production of tokens are also listed on the vocabulary and referred to as ‘tokens’. This fact is compatible with DM because in this theory the same syntactic rules are responsible for the combinations of morphemes and words, which thus makes morphemes and words the same type of unit. PFM, since rejecting morphemes and assigning semantics only to whole words occupying paradigmatic cells, is incompatible with ChatGPT on this criterion.

As for the demarcation of derivation and inflection, similar to DM ChatGPT, does not make a distinction between derivation and inflection and tokens may be non-derived words, derived words or inflected words, e.g. *read*, *reads*, *organization* and *organizations* are four

single tokens, Figures 6.15 and 6.16 for 'read' and 'reads', respectively, and Figures 6.17 and 6.18 for 'organization' and 'organizations'. In other words, all four words are frequent enough to be part of the vocabulary and therefore count as single tokens.

Figure 6.15 'read' is a single token

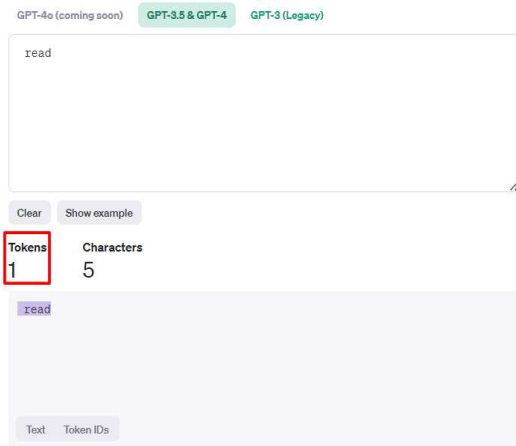


Figure 6.16 'reads' is a single token

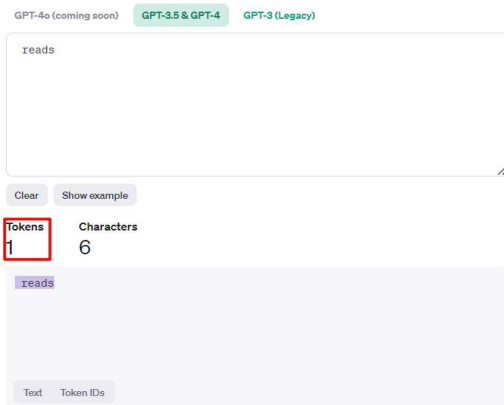


Figure 6.17: 'organization' is a single token

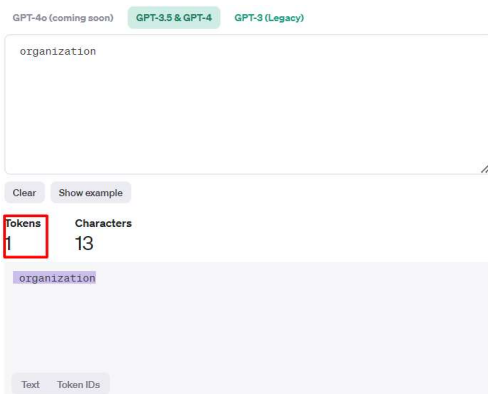
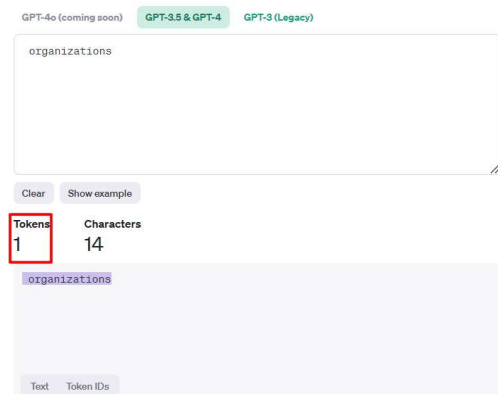


Figure 6.18: 'organizations' is a single token



There are also facts about ChatGPT that are incompatible with both theories of morphology discussed, DM and PFM. For example, ChatGPT's basic units (tokens) are a fixed-size set but emerge from data (are extracted from a large amount of text based on the frequency of adjacent letters without reference to meaning (recall the logic of the BPE algorithm), while morphemes in LT are more or less a predefined set too but one can enlarge this set with, e.g., morphemes borrowed from other languages, or with newly created morphemes, and the enrichment of the initial set of morphemes does not have anything to do with frequency of use of adjacent characters, which is the only criterion for vocabulary inclusion in ChatGPT. Moreover, based on the inventory of the ChatGPT's basic units, we can conclude that ChatGPT sees phonology, morphology and syntax as taking place simultaneously; while in LT morphology either precedes (the case of PFM) or follows syntax (the case of DM, where actually both morphology and phonology are postsyntactic). Then, as demonstrated in Section 2, tokens do not always coincide with linguistic units. To understand the issue let us scrutinize the tokenization of *illegible*, *illogical*, *illegal*, and *illiberal*, Figures 6.19 to 6.22. According to LT, all four words are derived with the prefixes *il-*. ChatGPT, however, parses the four words in four different ways and acknowledges a prefix 'il' / *il-* only in *illogical*. Nevertheless, ChatGPT's parsing of the four *il-* words is compatible with the results of a linguistic research in parsability, the so-called *Parsability Hypothesis* (Hay 2002). Hay claims the following:

“If the derived form is more frequent than the base it contains (*illegible* is more frequent than *legible*), then the whole-word route will have an advantage. If the derived form is less frequent than the base it contains (*illiberal* is less frequent than *liberal*), then the decomposed route will be advantaged.” (Hay 2002, 529)

ChatGPT's tokenization appears to have the same logic, just Hay overlooked that *illegible* does not have only a prefix but also a very frequent suffix, thus *il-* is not parsed (as Hay predicts) but *-igible* is, but it is actually in accord with the Parsability Hypothesis. In the Tokenizer it is *illeg-ible*, Figure 6.19. As regards *illiberal*, it is indeed decomposed as Hay predicts but the parsing in the Tokenizer differs from Hay's: *ill-ib-eral*, Figure 6.22. The tokenization of *ill-ib-eral* is, of course, due to the high frequency of its constituents, which is entirely compatible with Hay's research, only Hay assumes that English has a predefined set of morphemes and this does not allow her to acknowledge subword units that are highly frequent but not in the set.

Figure 6.19: 'illegible'

GPT-4o (coming soon) **GPT-3.5 & GPT-4** GPT-3 (Legacy)

illegible

Clear Show example

Tokens	Characters
2	10

illegible

Text Token IDs

Figure 6.20: 'illogical'

GPT-4o (coming soon) **GPT-3.5 & GPT-4** GPT-3 (Legacy)

illogical

Clear Show example

Tokens	Characters
2	10

illogical

Text Token IDs

Figure 6.21: 'illiberal'

GPT-4o (coming soon) **GPT-3.5 & GPT-4** GPT-3 (Legacy)

illiberal

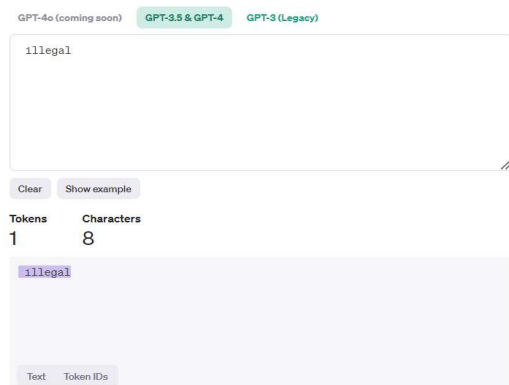
Clear Show example

Tokens	Characters
3	10

illiberal

Text Token IDs

Figure 6.22: 'illiberal'



So we come to the length of forms used by ChatGPT and in linguistic morphology. As a rule, ChatGPT’s prompts are long and consist of more than one token, i.e., prompts are (comparable with) sentences. By contrast, in morphology the longest sequences of basic units (morphemes) are words. Significantly, ChatGPT does not encounter form-meaning mismatches, while morphology in LT is about form-meaning mismatches. Thus, if ChatGPT can generate language based only on form, form and meaning in language are in a perfect relationship, i.e., form-meaning mismatches that are the central issue in morphological research seem to be fictitious problems that arise from data presentation (the small length of sequences analyzed). Moreover, in LT it is well-known that disambiguation of morphemes, (15), and words, (16) require longer sequences of units of the same kind.

- (15) Disambiguation of *-er*  
*read* → *read-er* → *read-er-ship*                    (nominal *-er*)  
 versus  
*large* → *larg-er* → \**larg-er-ship*                    (adjectival *-er*)
- (16) Disambiguation of *read*  
*I read*                    (*read* is a verb)  
 versus  
*I gave a read of*        (*read* is a noun)

In sum, the form-meaning mismatches that are the central topic of research in morphology are due to the small sequences of form analyzed.

Finally, ChatGPT does not rely on abstract morphemes and syntactic trees (the case of DM);<sup>36</sup> it does not recognize abstract lexemes and morphological paradigms, either (the case of PFM)—because working with abstract units such as lexemes and complex structures such as paradigms and trees is always more complex than working with real, existing forms organized in linear sequences (recall the discussion on complexity in Section 3). Note also that ChatGPT’s long sequences of tokens have their parallel in LT: verbs in polysynthetic languages (Fortescue et al. 2017) which are similar to sentences in English.

<sup>36</sup> On the problems with syntactic trees (they, among other things, grow in the wrong direction, from leaves to the root), see Manova (2022).

For the sake of completeness and fairness, Piantadosi (2023), with the title of his paper “Modern language models refute Chomsky’s approach to language”, put the focus on the discussion on the relationship between LLMs and LT on the Chomsky’s approach. However, LLMs reveal problems with other theoretical frameworks as well. For example, the paradigmatic approach is also challenged because from a ChatGPT perspective using paradigms for language processing is inefficient, in the sense that if language is generated step-by-step in terms of a linear sequence of forms, and we look for the next piece in that sequence, we need only a single piece of form. Then, the forms in a paradigm are, as a rule, not substitutable in a sentence slot, i.e., they do not compete with each other during sentence generation. Therefore, it does not make sense to activate the whole paradigm in order to find the single form needed. With the inclusion of inflected words in its vocabulary (Figures 6.15 to 6.18), ChatGPT provides a solution to the problem: no search for a specific piece of form in a paradigm but a direct access to this form in the vocabulary (the lexicon in LT terminology).

The goal of this section is not to provide an exhaustive list of differences and commonalities between ChatGPT and LT but to demonstrate a way of thinking that makes possible the use of ChatGPT as a theory of language. I hope to have demonstrated that ChatGPT is well compatible with LT.

## 6. Conclusion

The basic units with which ChatGPT understands and generates language are the tokens (subword units). ChatGPT’s vocabulary contains 100k tokens. I started with the introduction of the Byte Pair Encoding algorithm that serves for tokenization in ChatGPT. I showed that tokens can be letters, morphemes or words, as well as non-linguistic units. Consequently, tokens are not associated with semantics. Following the ChatGPT’s approach of form-based language processing and my previous research in morphology, I demonstrated a form-driven analysis of word-formation and supported it with the results of a psycholinguistic experiment with native speakers of English and Polish: similar to ChatGPT, native speakers process combinations of word-formation suffixes without reference to meaning. I used the technical details of the ChatGPT’s internal organization to compare the model with two major theories of morphology, the morpheme-based DM and the paradigm-based PFM. ChatGPT appears to mix insights from both theories. The model not only can serve as a theory of language but can also provide a novel perspective on existing theories and reveal shortcomings. With respect to morphology, the logic behind ChatGPT’s language processing shows that form-meaning mismatches, the central topic of research in morphology, are fictitious problems, in the sense that they are due to the way data are presented. The use of long(er) sequences of form, like in ChatGPT prompts, effortlessly resolves form-meaning mismatches.

## Bibliography

Aronoff, Mark, and Nanna Fuhrhop. 2002. “Restricting Suffix Combinations in German and English: Closing Suffixes and the Monosuffix Constraint.” *Natural Language and Linguistic Theory* 20: 451-490.

- Bobaljik, Jonathan D. 2017. Distributed Morphology. *Oxford Research Encyclopedia in Linguistics*, Oxford University Press, <https://doi.org/10.1093/acrefore/9780199384655.013.131>.
- Bonami, Olivier, and Gregory Stump. 2017. "Paradigm Function Morphology." In *The Cambridge handbook of morphology*, edited by Andrew Hippisley and Greg Stump. Cambridge University Press: 449–481.
- Brosche, Kimberly, and Stela Manova. 2022. "German Word Formation and the Organization of the Mental Lexicon." *Annual Conference of the Middle European Master Program in Cognitive Science*. Zagreb, June, [https://homepage.univie.ac.at/stela.manova/uploads/1/2/2/4/12243901/poster\\_suffix\\_combinations\\_version.pdf](https://homepage.univie.ac.at/stela.manova/uploads/1/2/2/4/12243901/poster_suffix_combinations_version.pdf)
- Burkacka, Iwona. 2015. "Suffix sets in Polish de-nominal derivatives." In *Affix ordering across languages and frameworks*, edited by Stela Manova. Oxford University Press: 233–258.
- Chomsky, Noam, Ian Roberts, and Jeffrey Watumull. 2023. Noam Chomsky: The false promise of ChatGPT. *The New York Times*, <https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html>
- Embick, David, and Rolf Noyer. 2007. Distributed morphology and the syntax/morphology interface. In *The Oxford Handbook of Linguistic Interfaces*, edited by Gillian Ramchand and Charles Reiss, editors, Oxford University Press: 289–324.
- Fabb, Nigel. 1988. "English Suffixation is Constrained only by Selectional Restrictions." *Natural Language and Linguistic Theory* 6: 527-539.
- Fortescue, Michael, Marianne Mithun, and Nicholas Evans (eds.). 2017. *The Oxford Handbook of Polysynthesis*. Oxford University Press.
- Haider, Hubert. 2023. "Is Chat-GPT a grammatically competent informant?". [lingbuzz/007285](https://lingbuzz/007285)
- Halle, Morris, and Alec Marantz. 1993. "Distributed morphology and the pieces of inflection." In *The view from building 20*, edited by Kenneth Hale and Samuel Jay Keyser. MIT Press: 111–176.
- Harley, Heidi, and Rolf Noyer. 1999. "State-of-the-article: Distributed Morphology." *GLOT International* 4: 3–9.
- Hay, Jennifer. 2002. "From speech perception to morphology: Affix-ordering revisited." *Language* 78 (3): 527–555.
- Katzir, Roni. 2023. "Why large language models are poor theories of human linguistic cognition. A reply to Piantadosi (2023)". [lingbuzz/007190](https://lingbuzz/007190)
- Kiparsky, Paul. 1982. Lexical Morphology and Phonology. In *The Linguistic Society of Korea, Linguistics in the Morning Calm*. Hanshin Publishing Co: 1-91.
- Lieber, Rochelle and Pavol Štekauer. 2014. *The Oxford handbook of derivational morphology*. Oxford University Press.
- Lieber, Rochelle. 2004. *Morphology and lexical semantics*. Cambridge University Press.



- Lieber, Rochelle. 2017. Derivational Morphology. *Oxford Research Encyclopedia of Linguistics*.  
<https://oxfordre.com/linguistics/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-248>. Retrieved 17 Sep. 2023.
- Manova, Stela. 2011a. *Understanding morphological rules*. Springer.
- Manova, Stela. 2011b. "A cognitive approach to SUFF1-SUFF2 combinations: A tribute to Carl Friedrich Gauss." *Word Structure* 4: 272-300.
- Manova, Stela. 2015a. "Affix order and the structure of the Slavic word." In *Affix Ordering Across Languages and Frameworks*, edited by Stela Manova. Oxford University Press: 205-230.
- Manova, Stela. 2015b. "Closing suffixes." In *Word-Formation in the European Languages*. Vol. 2, *Handbooks of Linguistics and Communication Science (HSK) 40/2.*, edited by P. Müller, I. Ohnheiser, S. Olsen and F. Rainer. De Gruyter Mouton: 956-971.
- Manova, Stela. 2022. "The linear order of elements in prominent linguistic sequences: Deriving Tns-Asp-Mood orders and Greenberg's Universal 20 with n-grams"  
[lingbuzz/006082](http://lingbuzz/006082).
- Manova, Stela. 2023a. "Ordering restrictions between affixes." In *The Wiley Blackwell Companion to Morphology*, edited by Peter Ackema, Sabrina Bendjaballah, Eulàlia Bonet and Antonio Fábregas. John Wiley & Sons DOI: [10.1002/9781119693604.morphcom058](https://doi.org/10.1002/9781119693604.morphcom058)
- Manova, Stela. 2023b. "ChatGPT, n-grams and the power of subword units: The future of research in morphology." In *Proceedings of DeriMo 2023*, edited by Matea Filko and Krešimir Šojat. Croatian Language Technology Society: 1-12. [lingbuzz/007598](http://lingbuzz/007598).
- Manova, Stela and Wolfgang U. Dressler. 2001. "Gender and Declension Class in Bulgarian." *Wiener Linguistische Gazette* 67-69: 45-81.
- Manova, Stela, Harald Hammarström, Itamar Kastner, and Yining Nie. 2020. "What is in a morpheme? Theoretical, experimental and computational approaches to the relation of meaning and form in morphology." *Word Structure* 13: 1-21.
- Manova, Stela and Georgia Knell. 2021. "Two-suffix combinations in native and non-native English: Novel evidence for morphomic structures." In *All things morphology: Its independence and its interfaces*, edited by Sedigheh Moradi, Marcia Haag, Janie Rees-Miller and Andrija Petrovic. Benjamins: 305-323.
- Manova, Stela and Luigi Talamo. 2015. "On the Significance of the Corpus Size in Affix-Order Research." *SKASE Journal of theoretical linguistics* 12: 369-397.
- Mansfield, John, Sabine Stoll and Balthasar Bickel. 2020. "Category clustering: A probabilistic bias in the morphology of verbal agreement marking." *Language* 96: 255-293.
- Moro, Andrea, Matteo Greco and Stefano F. Cappa. 2023. "Large languages, impossible languages and human brains." *Cortex* 167: 82-85.
- OpenAI. 2023. GPT-4 Technical Report. [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) [cs.CL], accessed July 1st, 2024.
- Piantadosi, Steven. 2023. "Modern language models refute Chomsky's approach to language." [lingbuzz/007180](http://lingbuzz/007180)

- Plag, Ingo, and Laura Balling. 2016. Derivational morphology: An integrative perspective on some fundamental questions. In *Word knowledge and word usage: A cross-disciplinary guide to the mental lexicon*, edited by Pirelli, Vito, Ingo Plag and Wolfgang U. Dressler. De Gruyter: 295-335.
- Rawski, Jon and Lucie Baumont. 2023. "Modern Language Models Refute Nothing." [lingbuzz/007203](https://lingbuzz/007203).
- Ryan, Kevin M. 2010. "Variable affix order: Grammar and learning." *Language* 86: 758–91.
- Sauerland, Uli. 2023. "The Impact of Large Language Models on Linguistic Theory and Generative Grammar: A Critical Analysis." [lingbuzz/007217](https://lingbuzz/007217).
- Selkirk, Elisabeth. 1982. *The Syntax of Words*. MIT Press.
- Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. "Neural Machine Translation of Rare Words with Subword Units." arXiv:1508.07909v5 [cs.CL]
- Siegel, Dorothy. 1974. *Topics in English Morphology*. MIT Press.
- Spencer, Andrew. 1991. *Morphological Theory*. Blackwell Publishers.
- Stump, Gregory. 2001. *Inflectional morphology: A theory of paradigm structure*. Cambridge University Press.
- Stump, Gregory. 2016. *Inflectional paradigms: Content and form at the syntax-morphology interface*. Cambridge University Press.
- Stump, Gregory, and Raphael A. Finkel. 2013. *Morphological Typology: From Word to Paradigm*. Cambridge University Press.
- Szymanek, Bogdan. 2000. "On morphotactics: Closing morphemes in English." In *PASE Papers in Language Studies*, edited by Bożena Rozwadowska. Aksel: 311-320.
- Szymanek, Bogdan. 2010. *A panorama of Polish word-formation*. Wydawnictwo KUL.