ChatGPT, n-grams and the power of subword units: The future of research in morphology

Stela Manova
manova.stela@gmail.com

4th International Workshop on Resources and Tools for Derivational Morphology
Dubrovnik, October 5, 2023

What this talk is about

- Challenging times for linguistics:
  - Generative Pre-trained Transformers (GPT), large language models (LLM), versus linguistic theory;
  - innateness of language versus AI, i.e. humans versus machines;
  - linear versus hierarchical organization of language structure;
  - theory versus application.
- I address the challenges by explaining how linguists can learn from them;
- Why (derivational) morphology is in a privileged position in comparison to other linguistic (sub)fields, cf. Byte Pair Encoding (BPE);
- I identify missing resources for the study of derivational morphology.

Preliminaries: Computer science and NLP vs. linguistic theory

- Significant advances in computer science and NLP in the past ten years or so.

- Generative Pre-trained Transformers (GPT), large language models (LLM), based on artificial neural networks (transformer architecture) and pre-trained on large data sets of unlabeled text entered the field of NLP.

- **A GPT (LLM) does not use grammar of the type known from linguistic theory.**

Preliminaries: Computer science and NLP vs. linguistic theory

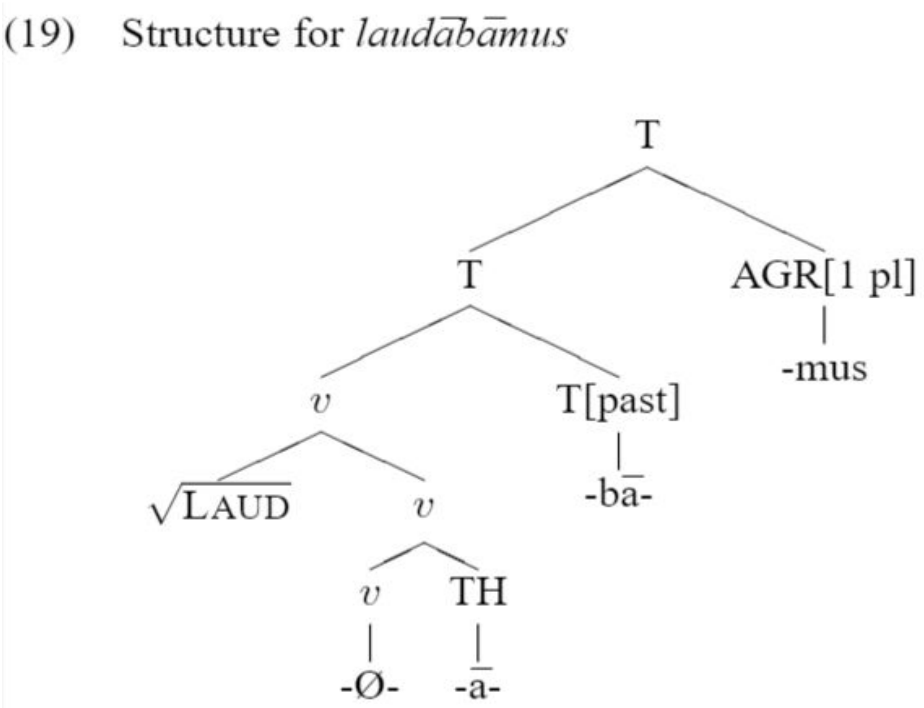
- On November 30, 2022, OpenAI launched ChatGPT, a LLM chatbot with a user-friendly interface that was additionally trained for dialogue with humans.

- ChatGPT raises questions about the future of linguistics, specifically of the correctness of the so-called Chomsky's approach that claims for innateness of language; this approach has been one of the dominant research paradigms in linguistics for years.

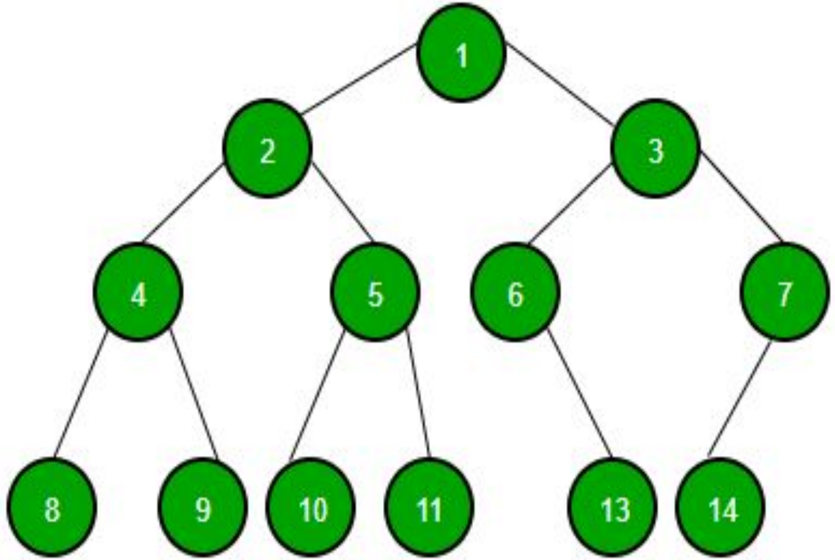
- Chomsky's approach (and most linguistic framework) assume a hierarchical organization of language evidenced in terms of syntactic trees (versus a linear analysis in LLMs).
  - What are syntactic trees: representations and/or evidence for internal organization of language?
  - Direction of growth: in linguistics, trees grow from leaves to the root, while trees in computer science follow the natural direction of growth, i.e. from the root to leaves.

Rooted binary trees in linguistics and computer science

Linguistics (Embick & Noyer 2012)



CS



Source: https://www.geeksforgeeks.org/binary-tree-data-structure/

Preliminaries: Computer science and NLP vs linguistic theory

- ChatGPT was launched in 2022 and is fluent in an impressive number of languages. Chomsky's approach celebrated 50 years of linguistics at MIT in 2011 but still cannot generate fluent language. This situation could only mean that, **most probably, Chomsky's theory (and linguistic theory in general) is unnecessarily complex**.

- ChatGPT can understand and generate language based only on form (a linear sequence of words in a prompt), which implies that **form and meaning in language are in a perfect relationship**. As ChatGPT prompts are longer than a word, often even longer than a sentence, the perfect relationship between meaning and form should be visible only if one considers long sequences of words (tokens); later, I will explain the Byte Pair Encoding (BPE) algorithm that is used for tokenization in LLMs.

Structure of the talk

- Byte Pair Encoding (BPE) and the role of subword units in NLP
- Complexity in computer science (*Big O* notation) and in linguistics
- Form-focused analysis of derivational morphology
  - A mathematical method, Gauss-Jordan elimination, will be applied to derivational data from English and Polish
  - Psycholinguistic experiment with native speakers of English and Polish
  - Discussion of results and findings
- Conclusion
  - The future of research in (derivational) morphology
  - Missing resources for research on derivational morphology

Byte Pair Encoding (BPE), Sennrich et al. (2016)

- Tokenization: dividing a string of text into a collection of tokens. [ChatGPT uses `tiktoken`, https://github.com/openai/tiktoken]
- Tokens typically serve as input to vectorization, i.e. tokens are converted into numerical representations for machine learning.
- Byte Pair Encoding (BPE) is a compression algorithm: it represents a large vocabulary with a small set of subword units.
- BPE iteratively merges the most frequent pair of consecutive bytes or characters in a text corpus until a predefined vocabulary size is reached. (ChatGPT uses cl100K_base).

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units, arXiv:1508.07909v5 [cs.CL]

Byte Pair Encoding (BPE)

Concepts related to BPE

- **Vocabulary:** A set of subword units that can be used to represent a text corpus.
- **Byte:** A unit of digital information that typically consists of eight bits.
- **Character:** A symbol that represents a written or printed letter or numeral.
- **Frequency:** The number of times a byte or character occurs in a text corpus.
- **Merge:** The process of combining two consecutive bytes or characters to create a new subword unit.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp/

BPE: Illustration

Text corpus: **“ab”, “bc”, “bcd”, and “cde”** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain “bc”:

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **“ab”, “bc”, “bcd”, and “cde”** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain “bc”:

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **“ab”, “bc”, “bcd”, and “cde”** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain “bc”:

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **“ab”, “bc”, “bcd”, and “cde”** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain “bc”:

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **“ab”, “bc”, “bcd”, and “cde”** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain “bc”:

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **"ab", "bc", "bcd", and "cde"** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain "bc":

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **“ab”, “bc”, “bcd”, and “cde”** (i.e. consists of four words)

Step 1: Initialize the vocabulary
Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)
Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters
The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair
Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts
Update the frequency counts of all the bytes or characters that contain “bc”:

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

BPE: Illustration

Text corpus: **"ab", "bc", "bcd", and "cde"** (i.e. consists of four words)

Step 1: Initialize the vocabulary

Vocabulary = {"a", "b", "c", "d", "e"}

Step 2: Calculate the frequency of each character (byte)

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1}

Step 3a: Find the most frequent pair of two characters

The most frequent pair is "bc" with a frequency of 2.

Step 3b: Merge the pair

Merge "bc" to create a new subword unit "bc".

Step 3c: Update frequency counts

Update the frequency counts of all the bytes or characters that contain "bc":

Frequency = {"a": 1, "b": 2, "c": 3, "d": 2, "e": 1, "bc": 2}

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.

Source: https://www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp

GPT tokenization

GPT-3 Codex

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: ✋🏾

Sequences of characters commonly found next to each other may be grouped together: 1234567890

Clear Show example

Tokens	Characters
64	252

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

GPT tokenization

Tokens 64

Characters 252

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: ������

Sequences of characters commonly found next to each other may be grouped together: 1234567890

TEXT TOKEN IDS

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

Token IDs

Tokens 64

Characters 252

```
[7085, 2456, 3975, 284, 530, 11241, 11, 475, 617, 836, 470, 25, 773, 452, 12843, 13, 198, 198, 3118, 291, 1098, 3435, 588, 795, 13210, 271, 743, 307, 6626, 656, 867, 16326, 7268, 262, 10238, 9881, 25, 12520, 97, 248, 8582, 237, 122, 198, 198, 44015, 3007, 286, 3435, 8811, 1043, 1306, 284, 1123, 584, 743, 307, 32824, 1978, 25, 17031, 2231, 30924, 3829]
```

TEXT TOKEN IDS

Token IDs

Tokens 64 **Characters** 252

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: ������

Sequences of characters commonly found next to each other may be grouped together: 1234567890

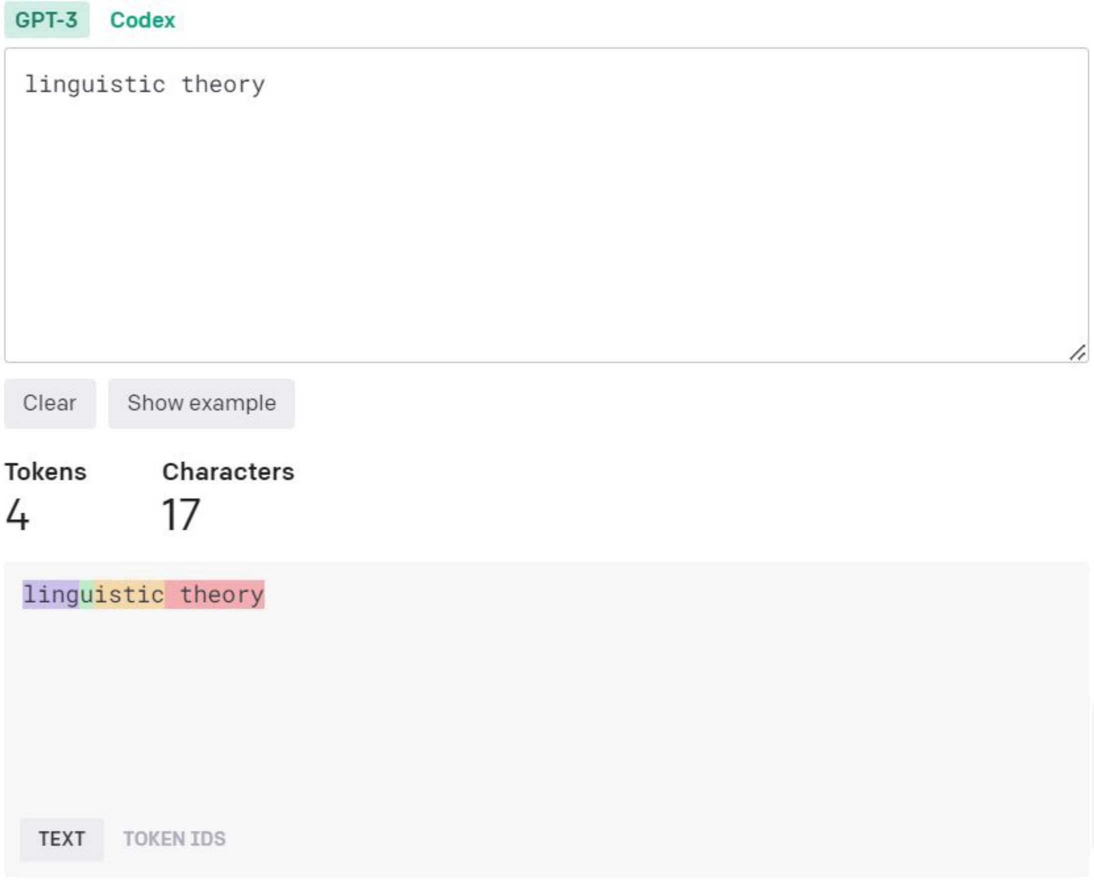
TEXT TOKEN IDS

Tokens 64 **Characters** 252

[7085, 2456, 3975, 284, 530, 11241, 11, 475, 617, 836, 470, 25, 773, 452, 12843, 13, 198, 198, 3118, 291, 1098, 3435, 588, 795, 13210, 271, 743, 307, 6626, 656, 867, 16326, 7268, 262, 10238, 9881, 25, 12520, 97, 248, 8582, 237, 122, 198, 198, 44015, 3007, 286, 3435, 8811, 1043, 1306, 284, 1123, 584, 743, 307, 32824, 1978, 25, 17031, 2231, 30924, 3829]

TEXT TOKEN IDS

Subword units vs. morphemes



Subword units vs. morphemes

GPT-3 Codex

```
linguistic theory
```

Clear Show example

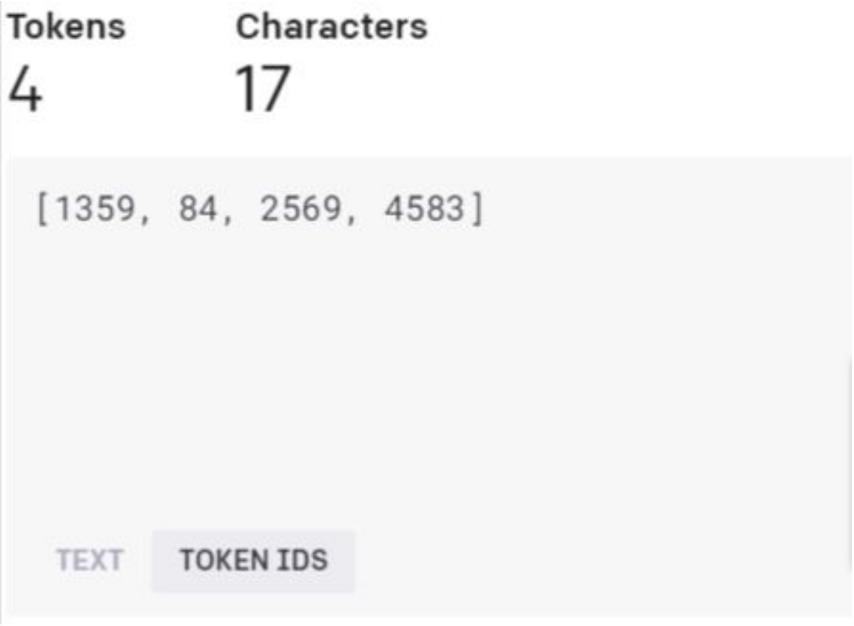
Tokens 4

Characters 17

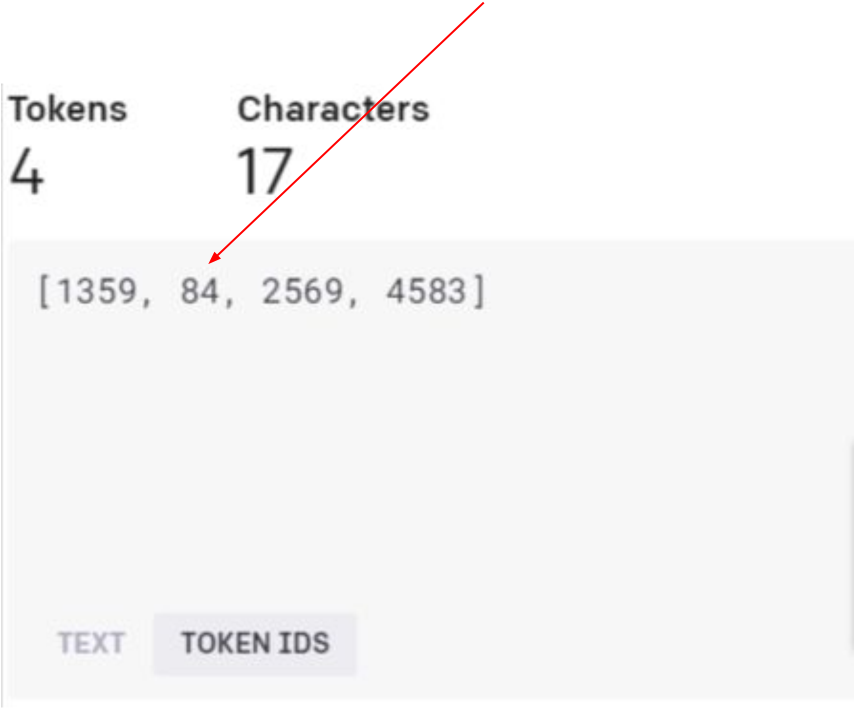
```
[1359, 84, 2569, 4583]
```

TEXT TOKEN IDS

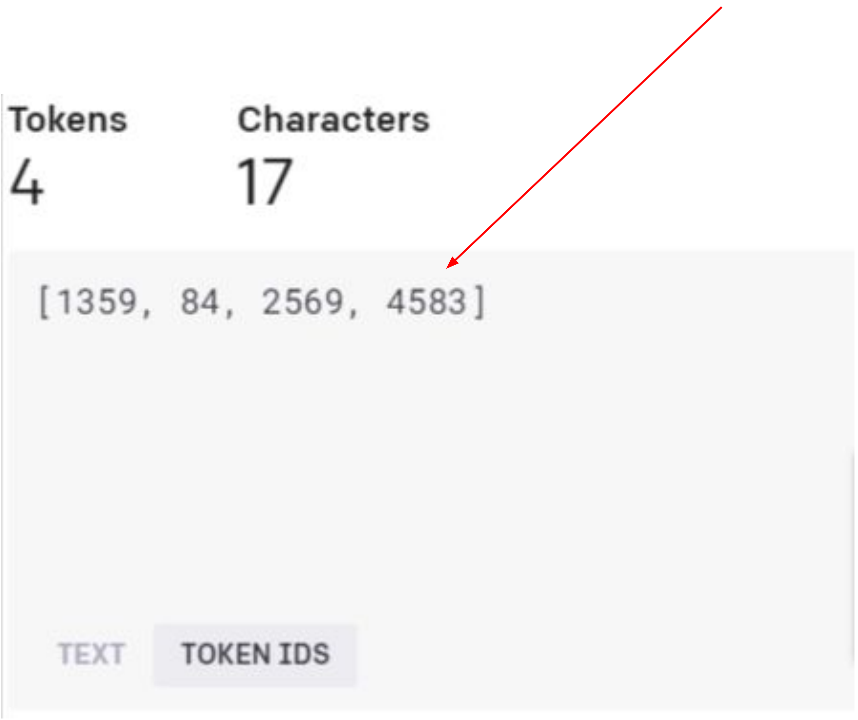
Subword units vs. morphemes



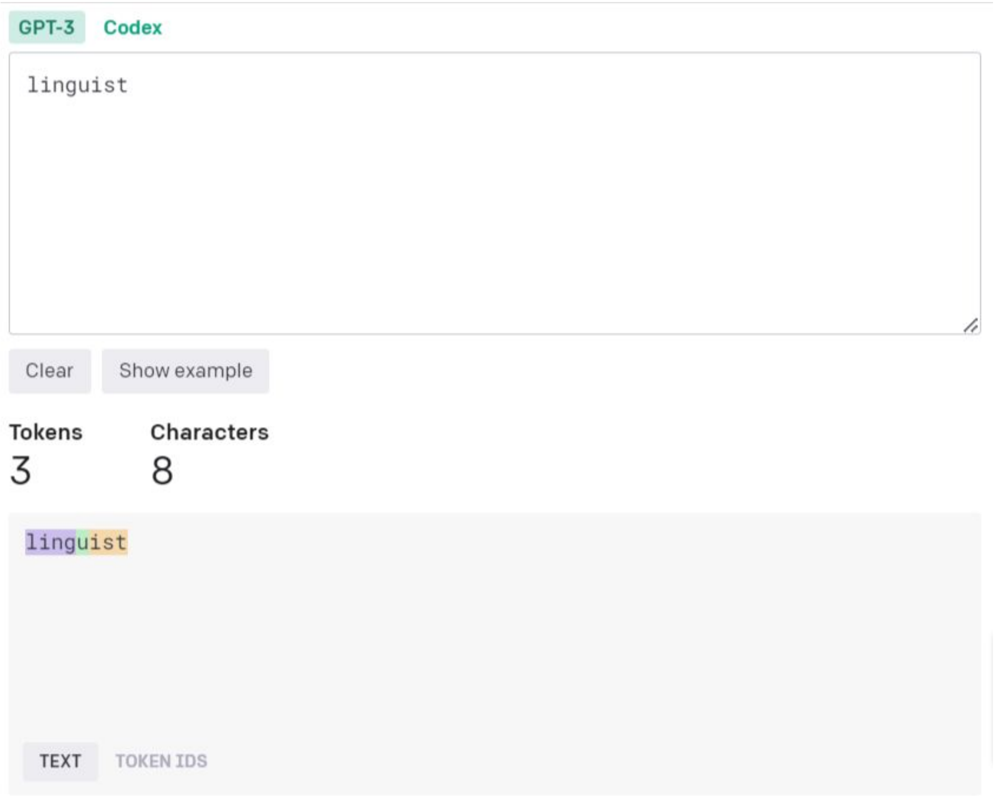
The most frequent token



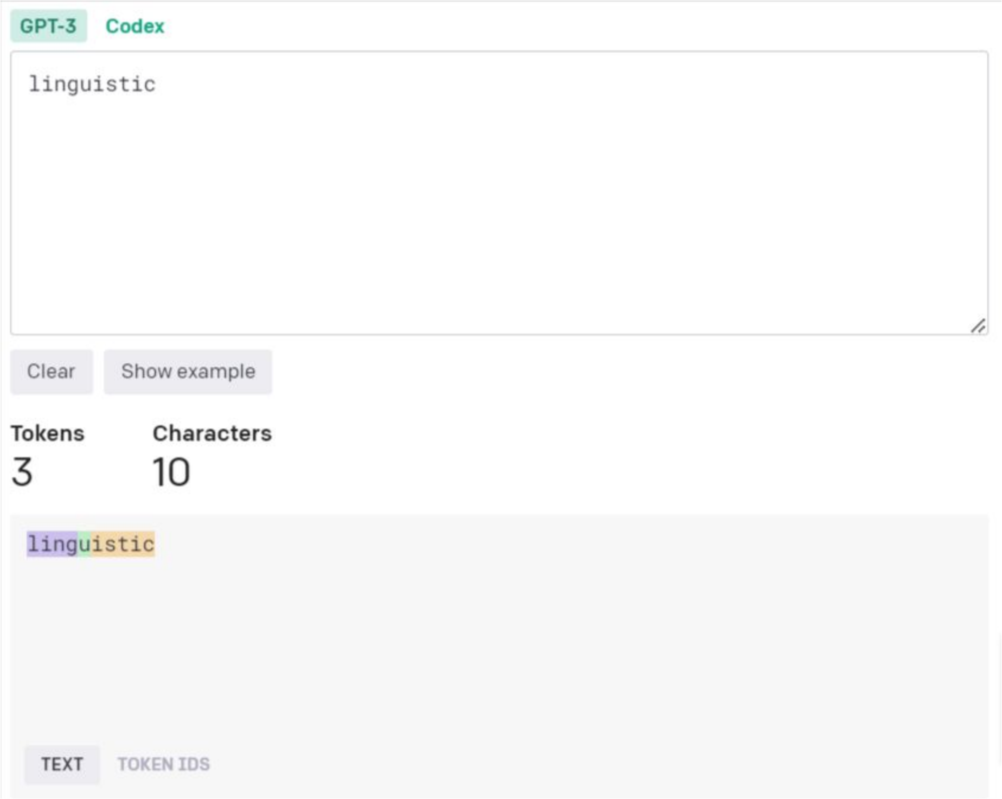
The least frequent token



Tokenization of derived words



Tokenization of derived words



Complexity

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 CS.

- The *Big O* notation tells us how an algorithm slows as data gow. That is, complexity is not a property of data (which is the case in linguistics, but of the algorithm (analysis).

- As an illustration let us evaluate two solutions of a task.
  [Note that the example is meant to help linguists understand the logic of the concept of complexity and is an oversimplification. In CS, the *Big O* notation evaluates the complexity of functions.]

The logic of the *Big O* notation

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

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 idea of Solution 1, to calculate the sum of the numbers from 1 to 1000, we have to perform 999 summations. That is, with the growth of the data, more effort is required to come to a solution.

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

The logic of the Big O notation

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 idea of Solution 1, to calculate the sum of the numbers from 1 to 1000, we have to perform 999 summations. That is, with the growth of the data, more effort is required to come to a solution.

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

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.

Complexity of a linguistic analysis

The ChatGPT approach to language relies on surface forms (for convenience, I speak of ‘phonological information’), see Rule 1; while a linguistics approach usually relies on semantics, see Rule 2.

Rule 1, form-based: If a word A ends in *-a,* attach the suffix B to it.
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*)?

- The information on which Rule 1 relies is ***not*** language-specific and is directly available: for the word A we have to evaluate whether it terminates in *-a* or not.

- The semantic information on which Rule 2 relies requires additional effort to be discovered and Rule 2 is also language-specific, in the sense that we need some knowledge of the language from which the data come in order to apply this rule.

Complexity of a linguistic analysis

***Rule 1, form-based*:** If a word A ends in *-a,* attach the suffix B to it.
***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*)?

- Rule 1 consists of two steps:
  i) we have to check whether A ends in *-a* and if yes, step ii);
  ii) attach the suffix B.

- Rule 2 involves more than two steps:
  i) evaluation whether a word is a verb; if yes, step ii);
  ii) ensure that the verb is of the type we need (an action verb); if yes, step iii);
  iii) add the productive suffix *-er* to derive an agent noun, if iv);
  iv) the derivation is possible; because e.g. *to edit* is an action verb but does not co-occur with *-er* (moreover, according to linguistic theory *to edit* is a backformation from *editor*, Manova, 2011a).

Rule 2 is more complex than Rule 1.

Rule 1: Example

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 (1) and (2), 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 *selo* 'village' in (1d).

(1)	Nouns: indefinite	→ definite
a.	sg.fem: *bluz-**a**$_1$* 'blouse'	→ *bluz-**a**$_1$-**ta*** 'the blouse'
b.	sg.masc: *bašt-**a**$_2$* 'father'	→ *bašt-**a**$_2$-**ta*** 'the father'
c.	pl.neut: *sel-**a**$_3$* 'villages'	→ *sel-**a**$_3$-**ta*** 'the villages'
d.	cf. sg.neut: *sel-o* 'village'	→ *sel-o-to* 'the village'

(2)	Adjectives: indefinite	→ definite
	sg.fem: *krasiv-**a**$_4$* 'beautiful'	→ *krasiv-**a**$_4$-**ta*** 'the beautiful'

Form-based analysis of derivational morphology

Form-based analysis of derivational morphology

- Undoubtedly, English is the language with the most profoundly studied derivational morphology. (Overviews of research on derivational morphology 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 (3): based on phonological information (see the different types of juncture 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.

(3) 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#*

Other form-focused analysis of English WF

- Fabb (1988) distributes the English suffixes into four groups:

(4) 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. Group3: suffixes that attach freely
d. Group 4: problematic suffixes

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

- Another highly relevant observation regarding the order of English derivational suffixes is reported in Manova (2011b) and Manova and Knell (2021). The observation is made with the help of the Gauss-Jordan elimination.

Gauss-Jordan elimination

Task: Solve this system of linear equations:

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

Example taken from:
https://math.libretexts.org/Bookshelves/Applied_Mathematics/Applied_Finite_Mathematics_(Sekhon_and_Bloom)/02%3A_Matrices/2.02%3A_Systems_of_Linear_Equations_and_the_Gauss-Jordan_Method

Gauss-Jordan elimination

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

Solution:

Write the augmented matrix.

$$\left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 1 & 2 & 1 & 8 \\ 3 & 1 & -1 & 2 \end{array}\right]$$

Manipulate the matrix, i.e. interchange rows or use elementary operations such as addition and multiplication until you get the matrix in a reduced row echelon form, which gives the values of all variables and is thus the solution to the problem.

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

$x = 1$

$y = 2$

$z = 3$

Gauss-Jordan elimination: The takeaway

- Pay attention to the well-known
- Manipulate well-known facts with the most simple logic
- Distribute the information so that there is only one option of a kind -- this option is the solution to the problem

Linguistic analysis: The combinability of the English suffix *-ist*

SUFF1	Lexical category of SUFF1	Followed by SUFF2 suffixes
-ist	N	*-dom, -ic, -y, -ize*

Data from Aronoff & Fuhrhop (2002), based on OED, CD 1994

Gauss-Jordan: The combinability of the English suffix *-ist*

Making suffix combinations unique pieces of word structure

SUFF1	Lexical category of SUFF1	SUFF2 suffixes according to their lexical categories
-ist	N	N: *-dom* (2) ADJ: *-ic* (631), *-y* (5) V: *-ize* (3)

Table from Manova (2011)
Data from Aronoff & Fuhrhop (2002), based on OED, CD 1994

Fixed combinations

SUFF1	Lexical category of SUFF1	SUFF2 suffixes according to their lexical categories
-ist	N	N: *-dom* (2) ADJ: *-ic* (631), *-y* (5) V: *-ize* (3)

Table from Manova (2011)
Data from Aronoff & Fuhrhop (2002), based on OED, CD 1994

Fixed combinations

SUFF1	Lexical category of SUFF1	SUFF2 suffixes according to their lexical categories
-ist	N	N: *-dom* (2) ADJ: *-ic* (631), *-y* (5) V: *-ize* (3)

Table from Manova (2011)
Data from Aronoff & Fuhrhop (2002), based on OED, CD 1994

Predictable combination

SUFF1	Lexical category of SUFF1	SUFF2 suffixes according to their lexical categories
-ist	N	N: *-dom* (2) ADJ: *-ic* (631), *-y* (5) V: *-ize* (3)

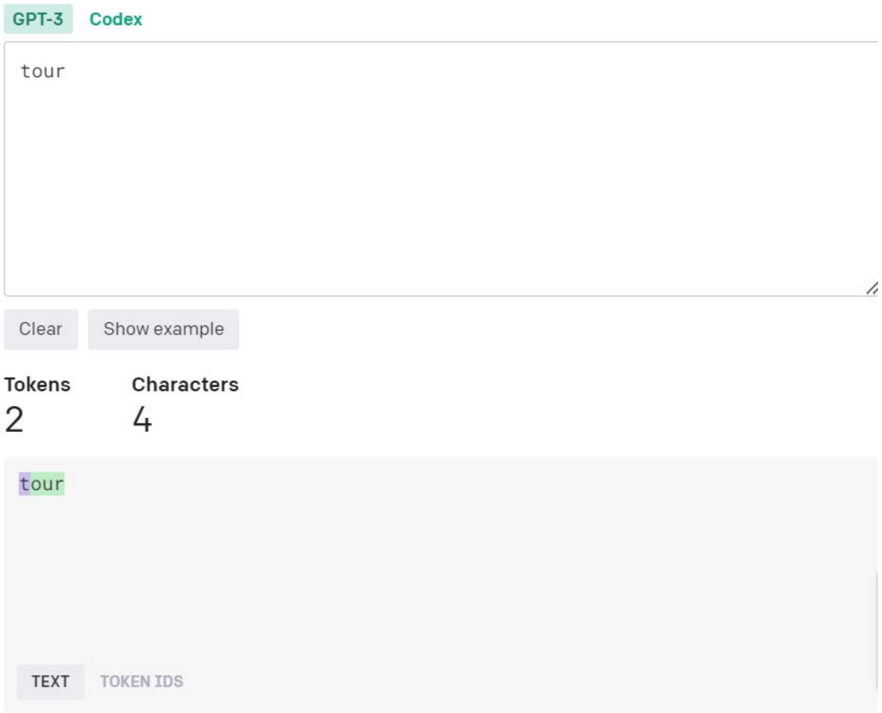
Table from Manova (2011)
Data from Aronoff & Fuhrhop (2002), based on OED, CD 1994

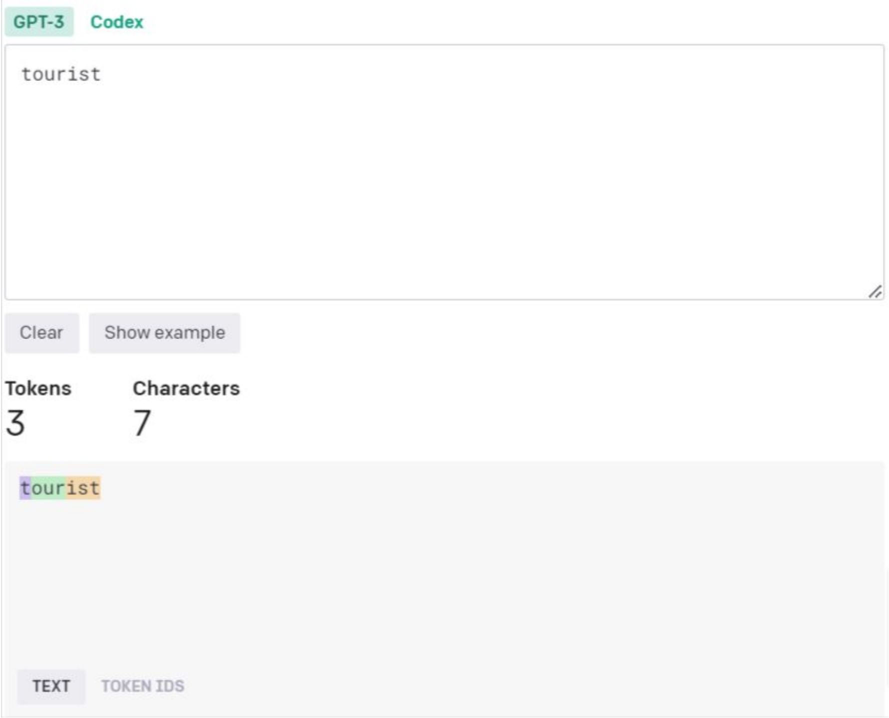
Types of suffix combinations: Summing up

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

Table 1: Combinability of the English suffix *-ist*
(data from Aronoff and Fuhrhop, 2002, based on OED, CD, 1994)

English derivational morphology: a ChatGPT perspective

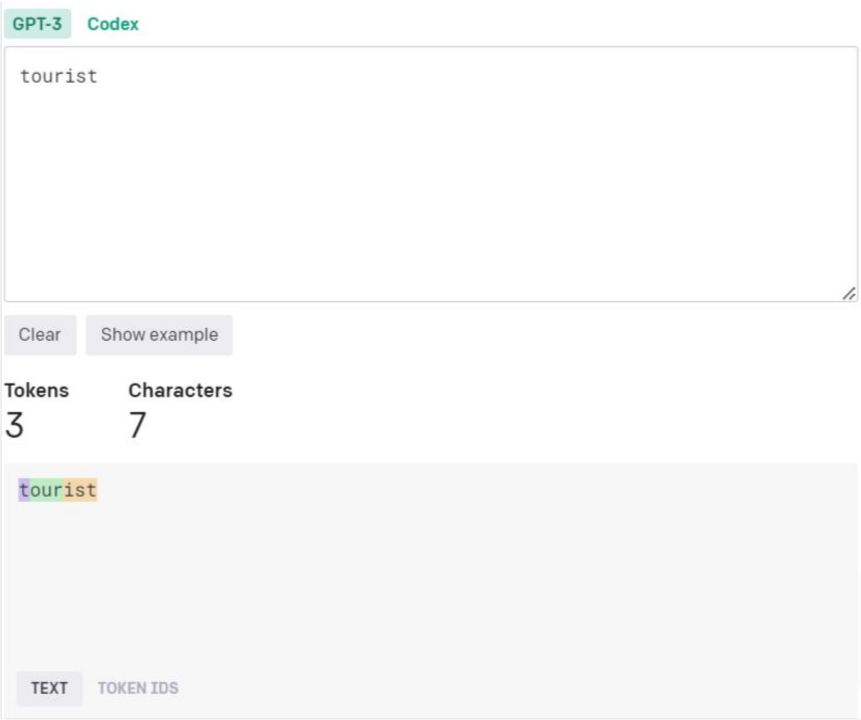


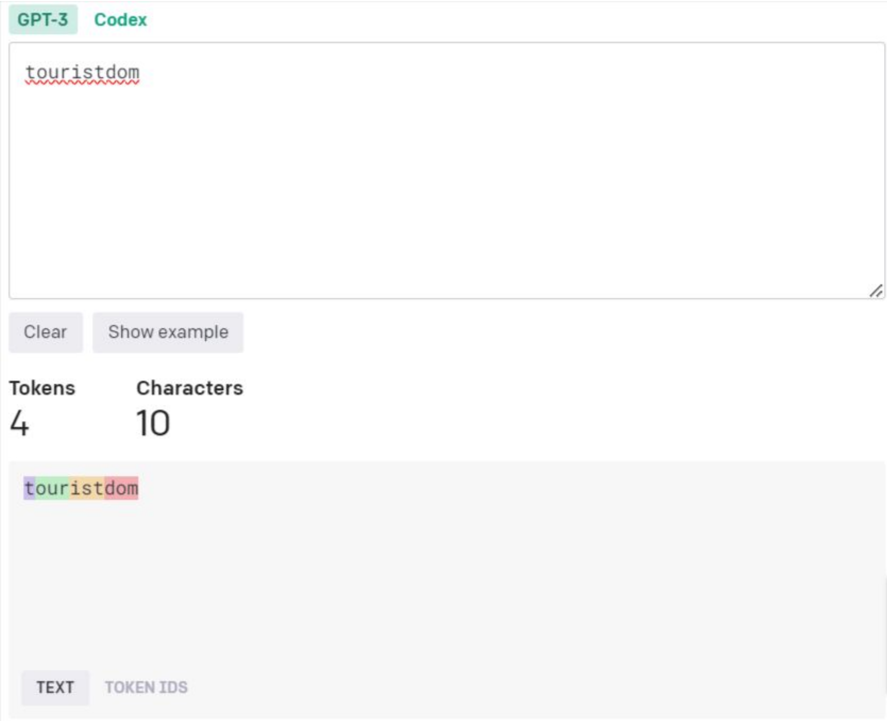


SUFF1	Lexical category of SUFF1	SUFF2 classified for lexical category; in brackets, number of types (lemmas) derived with the combination SUFF1-SUFF2	
-ist	N	N: *-dom* (2) A: ***-ic* (631)**, *-y* (5) V: *-ize* (3)	[*fixed combination*] [***predictable combination***] [*fixed combination*]

Table 1: Combinability of the English suffix *-ist*
(data from Aronoff and Fuhrhop, 2002, based on OED, CD, 1994)

English derivational morphology: a ChatGPT perspective

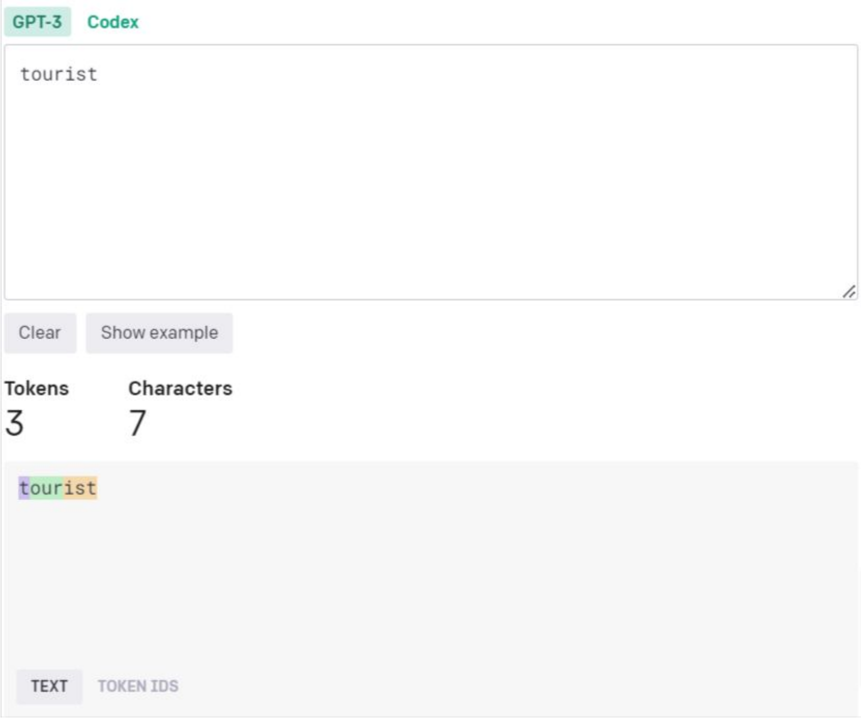


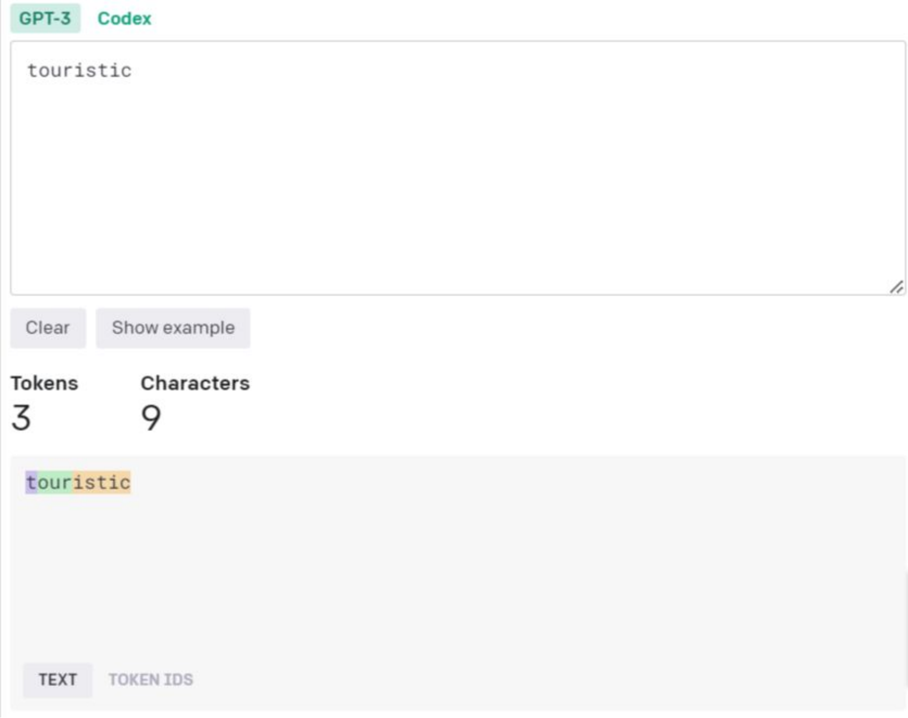


SUFF1	Lexical category of SUFF1	SUFF2 classified for lexical category; in brackets, number of types (lemmas) derived with the combination SUFF1-SUFF2	
-ist	N	N: *-dom* (2)	[*fixed combination*]
		A: ***-ic* (631)**, *-y* (5)	[***predictable combination***]
		V: *-ize* (3)	[*fixed combination*]

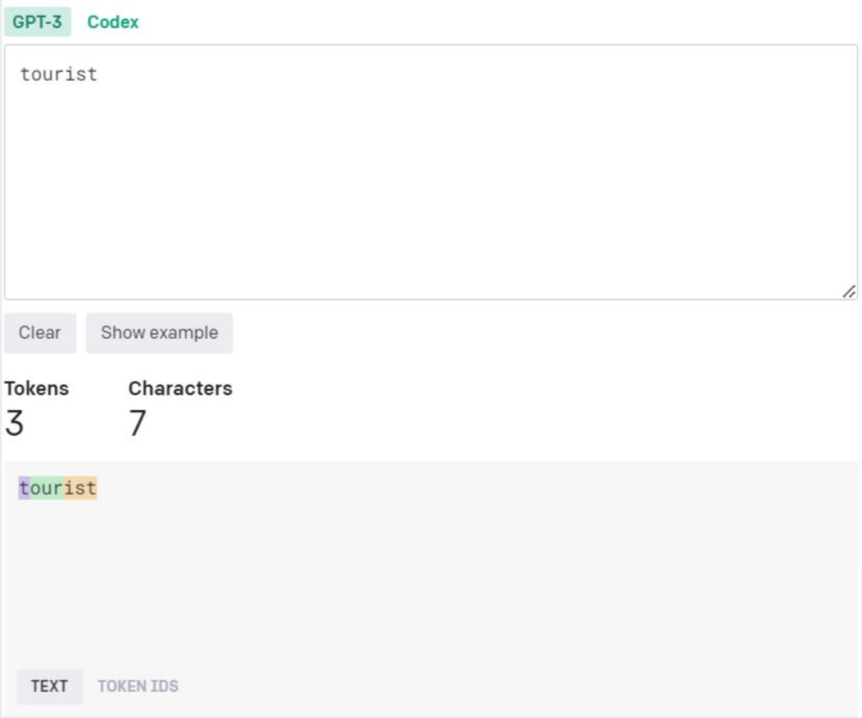
Table 1: Combinability of the English suffix *-ist*
(data from Aronoff and Fuhrhop, 2002, based on OED, CD, 1994)

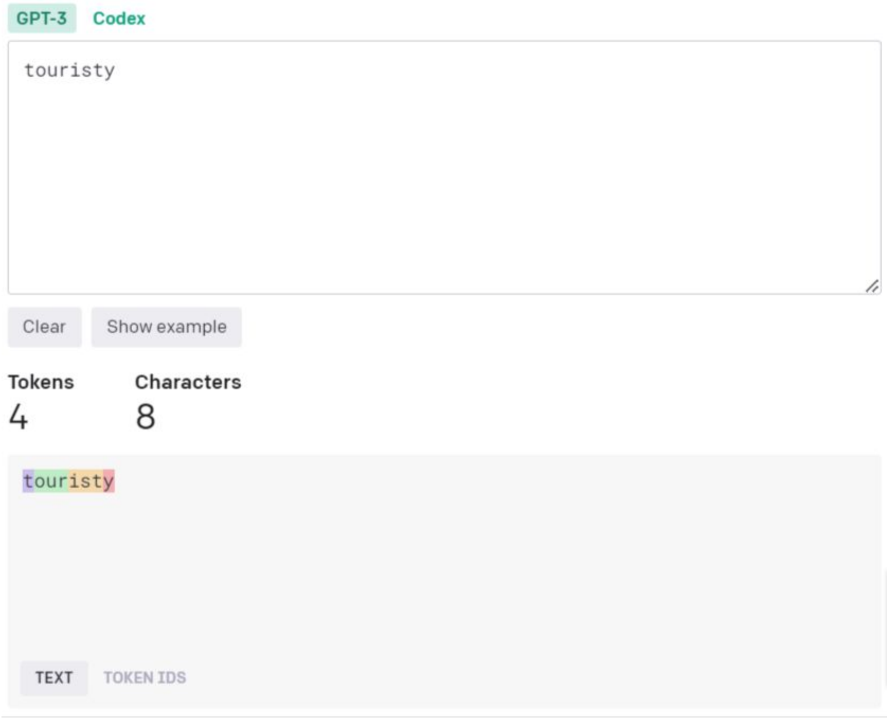
English derivational morphology: a ChatGPT perspective





English derivational morphology: a ChatGPT perspective

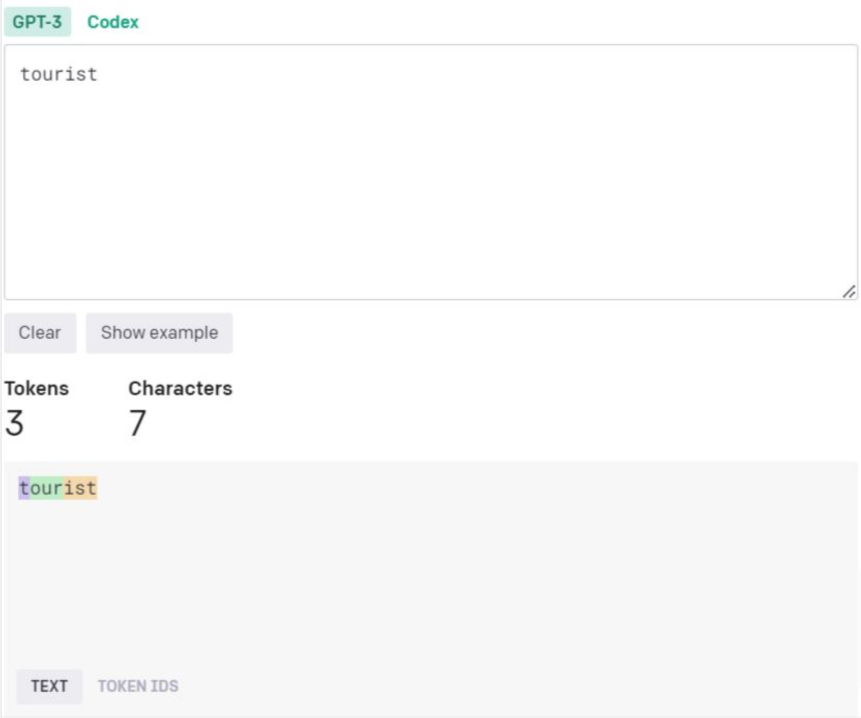


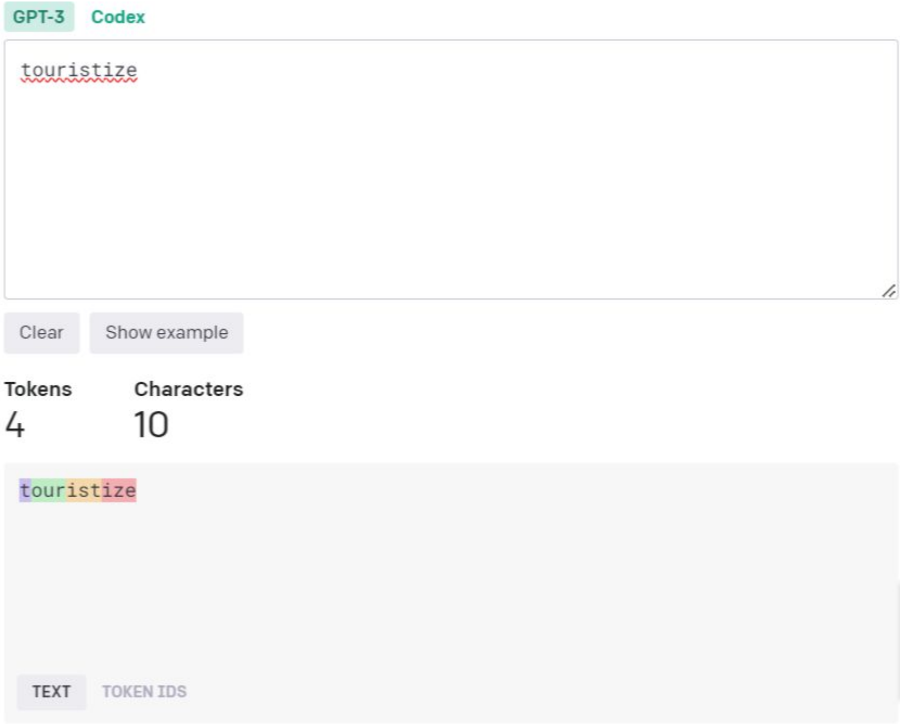


SUFF1	Lexical category of SUFF1	SUFF2 classified for lexical category; in brackets, number of types (lemmas) derived with the combination SUFF1-SUFF2	
-ist	N	N: *-dom* (2) A: ***-ic* (631)**, *-y* (5) V: *-ize* (3)	[*fixed combination*] [***predictable combination***] [*fixed combination*]

Table 1: Combinability of the English suffix *-ist*
(data from Aronoff and Fuhrhop, 2002, based on OED, CD, 1994)

English derivational morphology: a ChatGPT perspective





A more complex example from Polish

SUFF1	Lexical category of SUFF1	Lexical category of SUFF2	SUFF1-SUFF2 exemplified	Notes
-arz	N	i. ADJ: *-n(y)* (2)	moc-*ar-n(y)* 'strong'	[derives only 2 adjectives]
		ii. ADJ: *-ow(y)* (1)	gęśl-*arz-ow(y)* 'of fiddler'	[derives a single adjective]
		iii. **ADJ: *-sk(i)* (>10)**	pis-*ar-sk(i)* 'of writer	[**default** for derivation of adjectives]
		a. **N: *-czyk* (>10)**	piek-*ar-czyk* 'baker's apprentice'	[**default** for derivation of **persons**, cf. f]
		b. N: *-k(a)* (2)	mur-*ar-k(a)* 'bricklaying'	[derives only 2 abstract nouns, cf. e]
		c. **N: *-ni(a)* (>10)**	kreśl-*ar-ni(a)* 'drafting studio'	[derives nouns for **places**]
		d. N: *-nik* (1)	piek-*ar-nik* 'oven'	[derives a single **object**]
		e. **N: *-stw(o)* (>10)**	księg-*ar-stw(o)* 'all booksellers'	[**default abstract/collective nouns**, cf. b]
		f. N: *-yn(a)* (5)	mur-*arz-yn(a)* 'bad bricklayer'	[derives only 5 nouns for persons, cf. a]

Table 2: Combinability of the Polish suffix *-arz*

Processing of morphological structure by humans

- Considering the fact that derivational suffixes in English and Polish seem to form only fixed and predictable combinations, I hypothesized that native speakers should have memorized them and, consequently, should be able to process them without reference to meaning, that is, based exclusively on form.

- To test this hypothesis, I designed a psycholinguistic experiment. 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).

Psycholinguistic experiment

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, *-arny* in Polish) were presented in a randomized order, and participants were asked to decide intuitively, as quickly as possible, which of the combinations exist and which do not exist as word terminations in the respective language. Of the 60 combinations, 30 exist in the respective language and 30 do not. Of the existing combinations, 15 were productive (>10 types) and 15 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 English), 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 respective language). No non-existing combinations included any phonological and/or orthographical impossibilities in the respective language. Participants were given a 10-minute time limit to complete the main task. (On average, the subjects used approximately one third of the time.)

Data analysis

We used independent t-tests to consider possible significance of overall scores, as well as for stimulus type: existing vs. non-existing and productive vs. unproductive combinations (Figure 1).

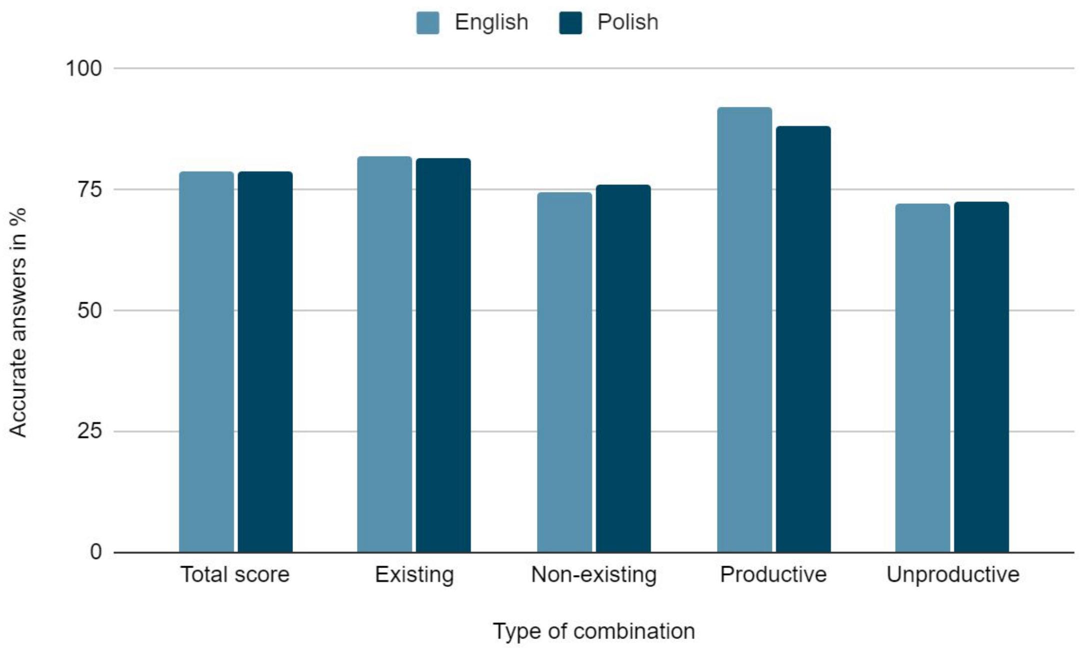


Figure 1

Discussion

- 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 (>10 types) and unproductive combinations.

- 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, 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).

- Inflection did not seem to have an impact on the processing on suffix combinability in derivation. I therefore conclude that native speakers of Polish see inflection as forming a natural subword unit with the derivational material that precedes it.

Discussion

- 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 subconsciously extracted and memorized adjacent suffixes in terms of bigrams and trigrams, during language acquisition (cf. the training of ChatGPT).

- Further support to the conclusion that adjacent derivational and inflectional suffixes should be treated together provides Polish diminutive morphology. Polish, like the other Slavic languages (Manova 2015a), derives second-grade diminutives the forms of which contain a sequence of two adjacent diminutive suffixes:

  dom 'house' → DIM1 *dom-ek* 'small house' → DIM2 *dom-**ecz-ek*** 'very small house'.

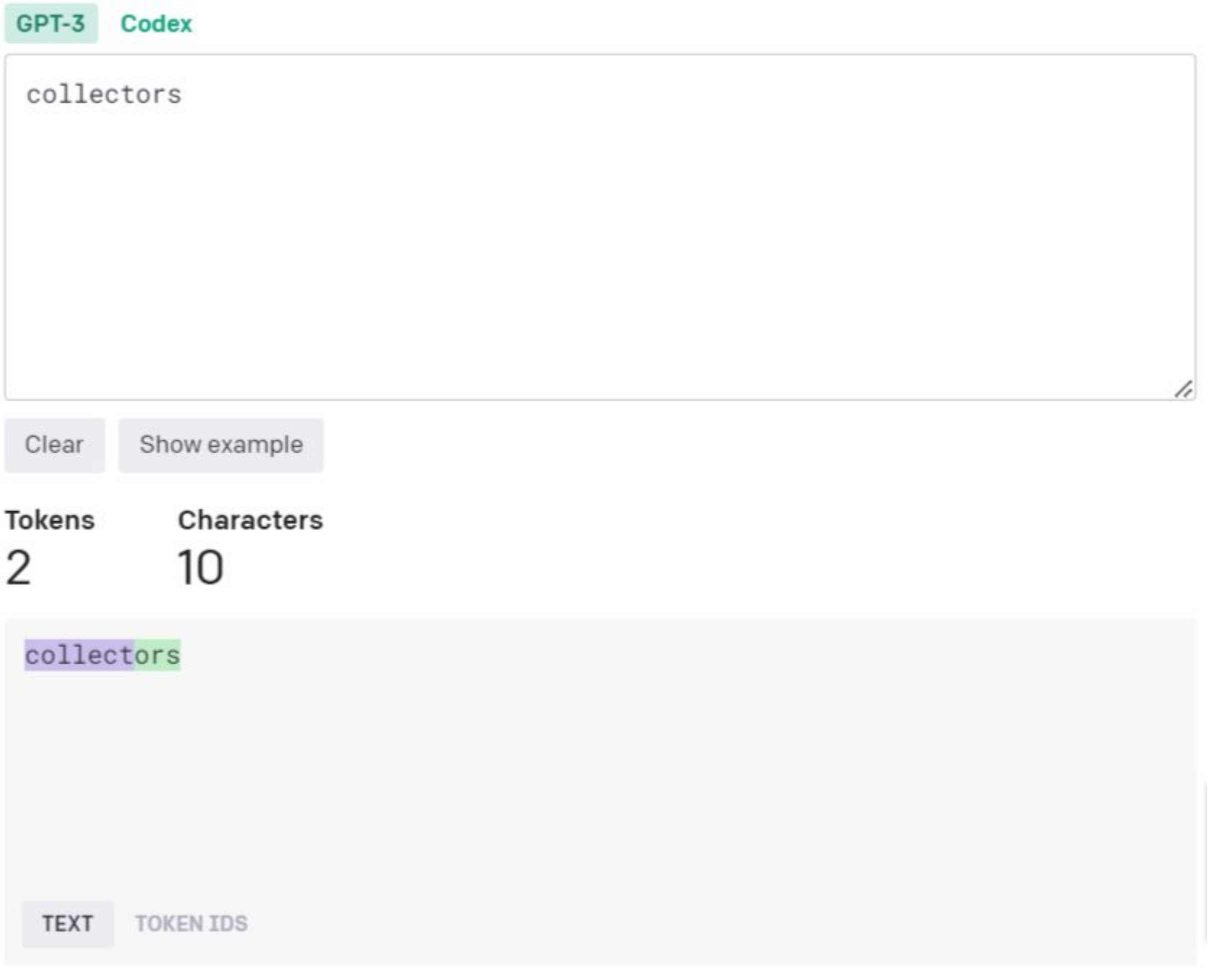
  The selection of the second diminutive suffix entirely depends on the phonological make-up of the first diminutive suffix. The selection of the DIM1 suffix is also form-driven in all but one case: the unproductive class of the feminine-gender nouns in -C selects DIM1 suffix based not on phonology but on gender.

	DIM1 suffixes	DIM2 suffixes	
Nouns in		Productive (attach by addition)	Unproductive (attach by substitution of a DIM1 suffix, i.e. do not combine with DIM1 suffixes)
-C	*-ek* *-ik / -yk* *-uszek* (unproductive)	*-ek*	*-uszek, -aszek*
	-iszek /-yszek (unproductive) *-aszek* (unproductive) *-ulek* (unproductive) *-ka* (unproductive, selects feminine nouns)		
-a	*-ka*	*-ka*	
	-uszka (unproductive) *-iczka /-yczka* (unproductive)		
-o / -e	*-ko*	*-ko*	
	-uszko (unproductive)		

Table 3: Combinability of the DIM suffixes in Polish (from Manova & Winzernitz 2011)

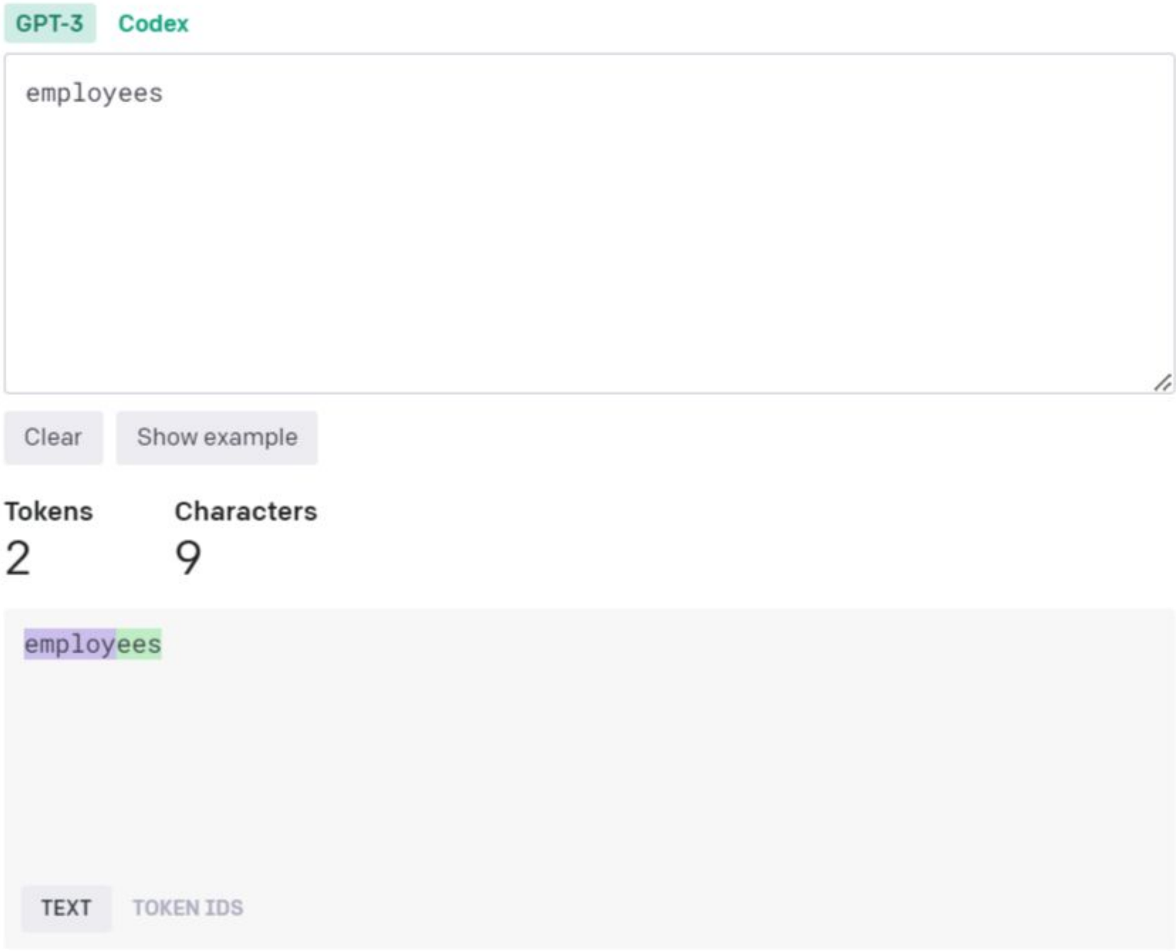
GPT

The derivation-infection distinction in English



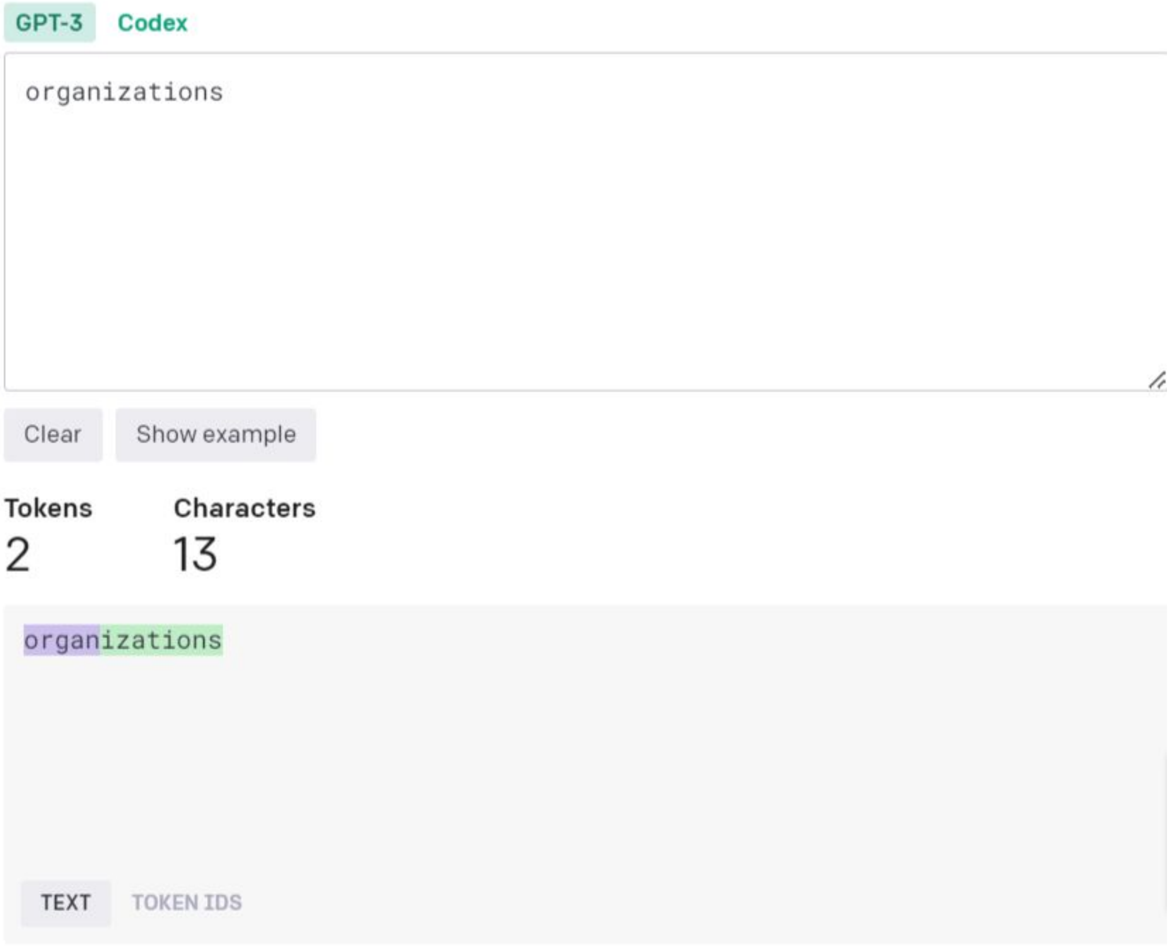
GPT

The derivation-inflection distinction in English



GPT

The derivation-inflection distinction in English



Conclusions

- Based on the BPE algorithm used for tokenization in LLMs, a mathematical method for problem solving, the so-called Gauss-Jordan elimination, and previous research on affix order (by other authors and my own), I put forward the idea of form-based analysis of derivational morphology and illustrated it with data from two typologically distinct languages, English with very poor inflectional morphology, and Polish with very rich inflection.

- A psycholinguistic experiment with native speakers of Polish and English confirmed the correctness of the proposal: Native speakers do not need semantic cues to process affix ordering in derivation. They seem to have subconsciously memorized linearly adjacent affixes, be they derivational or inflectional, as bigrams and trigrams, without reference to semantics, which is exactly what happens during the subword tokenization in a LLM.

Conclusions

- Morphology works with units of a very small length and the form-meaning correspondences in my analysis (and in (derivational) morphology in general) are not perfect, cf. the long sequences of form used in ChatGPT where form and meaning appear to be in a perfect one-to-one relationship. Nevertheless, a flexible approach, one that operates with defaults and a fixed reasonable number of exceptions (ten or fewer exceptions in my analysis) successfully derives new words from already suffixed ones in English and Polish.

- Future research is needed to see how the suggested approach works with unsuffixed bases, although cf. psycholinguistic research on derivations such as *work-er* and pseudoderivations such as *corn-er*, for the human parser they contain the same morpheme *-er*.

- **Form-focused** (preferably cross-linguistic) **resources for (derivational) morphology** providing information about word structure in terms of bigrams and trigrams (linear sequences of adjacent subword units) and their frequency will be essential for future research. Such resources do not exist currently.

- Claims that ChatGPT does not reflect human-like language processing in morphology (and not only) are, most probably, due to the lack of linguistic research that adopts a ChatGPT perspective on language.

Thank you for your attention!

Stela Manova
manova.stela@gmail.com
https://homepage.univie.ac.at/stela.manova/
https://sites.google.com/view/stelamanova

References

Aronoff, Mark. 1976. *Word Formation in Generative Grammar*. MIT Press, Cambridge, Ma.

Aronoff, Mark. 1994. *Morphology by itself: Stems and Inflectional Classes*. MIT Press, Cambridge, Ma.

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.

Bauer, Laurie, and Paul Nation. 1993. Word Families. *International Journal of Lexicography* 6: 253–279.

Bobaljik, Jonathan D. 2017. Distributed Morphology. *Oxford Research Encyclopedia in Linguistics*, Oxford: Oxford University Press, https://doi.org/10.1093/acrefore/9780199384655.013.131.

Bonami, Olivier, and Gregory Stump. 2017. Paradigm Function Morphology. In Andrew Hippisley and Greg Stump, editors, *The Cambridge handbook of morphology* Cambridge University Press, Cambridge, pages, 449–481.

Bonami, Olivier, and Jana Strnadová. 2019. Paradigm structure and predictability in derivational morphology, *Morphology* 29: 167–197.

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

Burkacka, Iwona. 2015. Suffix sets in Polish de-nominal derivatives. In Stela Manova, editor, *Affix ordering across languages and frameworks*. Oxford University Press, New York, pages 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-chatgptai.html

Fabb, Nigel. 1988. English Suffixation is Constrained only by Selectional Restrictions. *Natural Language and Linguistic Theory* 6: 527-539.

Embick, David, and Rolf Noyer. 2007. Distributed morphology and the syntax/morphology interface. In Gillian Ramchand, and Charles Reiss, editors, *The Oxford Handbook of Linguistic Interfaces*. Oxford University Press, New York, pages, 289–324.

Halle, Morris, and Alec Marantz. 1993. Distributed morphology and the pieces of inflection. In Kenneth Hale, and Samuel Jay Keyser, editors, *The view from building 20*. MIT Press, Cambridge, Ma, pages 111–176.

Hammarström, Harald, and Lars Borin. 2011. Unsupervised Learning of Morphology. Computational Linguistics 37 (2): 309–350. doi: https://doi.org/10.1162/COLI_a_00050

Harley, Heidi, and Rolf Noyer. 1999. State-of-the-article: Distributed Morphology. *GLOT International* 4: 3–9.

Hathout, Nabil, and Fiammetta Namer. 2019. Paradigms in word formation: what are we up to? *Morphology* 29, 153–165.

Herce, Borja. 2023. *The Typological Diversity of Morphomes: A Cross-Linguistic Study of Unnatural Morphology.* Oxford University Press, Oxford.

Huyghe, Richard, anf Rossella Varvara. 2023. *Affix rivalry: Theoretical and methodological challenges. Word Structure* 16: 1-23.

Katzir, Roni. 2023. Why large language models are poor theories of human linguistic cognition. A reply to Piantadosi (2023), lingbuzz/007190

Kiparsky, Paul. 1982. Lexical Morphology and Phonology. In *The Linguistic Society of Korea, Linguistics in the Morning Calm*. Hanshin Publishing Co, Seoul, pages 1-91.

Kozachkov, Leo, Ksenia V. Kastanenka, and Dmitry Krotov. 2023. Building transformers from neurons and astrocytes. Proceedings of the National Academy of Sciences (PNAS), Vol. 120 No. 34, pages = e2219150120, https://doi.org/10.1073/pnas.221915012

Körtvélyessy, Lívia, Bagasheva, Alexandra and Štekauer, Pavol. 2020. *Derivational Networks Across Languages*, De Gruyter Mouton, Berlin, Boston.

Lieber, Rochelle. 2004. *Morphology and lexical semantics*. Cambridge: 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.

Lieber, Rochelle, and Pavol Štekauer. 2014. *The Oxford handbook of derivational morphology*. Oxford University Press, Oxford.

Luís, Ana R., and Ricardo Bermúdez-Otero, editors, 2016. *The morphome debate*. Oxford University Press, Oxford.

Maiden, Martin. 2004. Morphological autonomy and diachrony. *Yearbook of Morphology* 2004: 137–175.

Manova, Stela. 2011a. *Understanding morphological rules*. Springer, Dordrecht.

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 Stela Manova, editor, *Affix Ordering Across Languages and Frameworks*. Oxford University Press, New York, pages 205-230.

Manova, Stela. 2015b. Closing suffixes, In P. Müller, I. Ohnheiser, S. Olsen, and F. Rainer, editos, *Word-Formation in the European Languages*. Vol. 2, Handbooks of Linguistics and Communication Science (HSK) 40/2. De Gruyter Mouton, Berlin, pages 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

Manova, Stela. 2023. Ordering restrictions between affixes. In Peter Ackema, Sabrina Bendjaballah, Eulàlia Bonet, and Antonio Fábregas, editors, *The Wiley Blackwell Companion to Morphology*. John Wiley & Sons, Hoboken, NJ. DOI: 10.1002/9781119693604.morphcom058

Manova, Stela, and Wolfgang U. Dressler. 2001. Gender and Declensional Class in Bulgarian. *Wiener Linguistische Gazette* 67-69: 45-81.

Manova, Stela, and Mark Aronoff. 2010. Modeling affix order. *Morphology* 20: 109-131.

Manova, Stela, and Kimberley Winternitz. 2011. Suffix Order in Double and Multiple Diminutives: With Data from Polish and Bulgarian. *Studies in Polish Linguistics* 6: 115-138.
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.
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 Sedigheh Moradi, Marcia Haag, Janie Rees-Miller, and Andrija Petrovic, editors, *All things morphology: Its independence and its interfaces.* Benjamins, Amsterdam, pages 305-323.
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 [cs.CL], accessed July 1st, 2023.
Piantadosi, Steven. 2023. Modern language models refute Chomsky's approach to language, lingbuzz/007180
Plag, Ingo, and Laura Balling. 2016. Derivational morphology: An integrative perspective on some fundamental questions. In Pirelli, Vito, Ingo Plag, and Wolfgang U. Dressler, editors, *Word knowledge and word usage: A cross-disciplinary guide to the mental lexicon.* De Gryuter, Berlin, pages 295-335.
Rainer, Franz. 2016. Blocking. *Oxford Research Encyclopedia of Linguistics.* Retrieved 19 Sep. 2023, from https://oxfordre.com/linguistics/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-33.
Rawski, Jon, and Lucie Baumont. 2023. Modern Language Models Refute Nothing, 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.
Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units, arXiv:1508.07909v5 [cs.CL]
Selkirk, Elisabeth. 1982. *The Syntax of Words*. MIT Press, Cambride, Ma.
Siegel, Dorothy. 1974. *Topics in English Morphology*. MIT Press, Cambride, Ma.
Spencer, A. 1991. *Morphological Theory*. Oxford: Blackwell Publishers.
Stump, Gregory. 2001. *Inflectional morphology: A theory of paradigm structure.* Cambridge University Press, Cambridge.
Stump, Gregory. 2016. *Inflectional paradigms: Content and form at the syntax-morphology interface*. Cambridge University Press, Cambridge.
Stump, Gregory, and Raphael A. Finkel. 2013. *Morphological Typology: From Word to Paradigm*. Cambridge University Press, Cambridge.
Szymanek, Bogdan. 2000. On morphotactics: Closing morphemes in English. In Bożena Rozwadowska, editor, *PASE Papers in Language Studies*, Aksel, Wrocław, pages 311-320.
Szymanek, Bogdan. 2010. *A panorama of Polish word-formation*. Wydawnictwo KUL, Lublin.
Ševčíková, Magda, and Zdeněk Žabokrtský. 2014. Word-formation network for Czech. In Nicoletta Calzolari et al., editors, *Proceedings of the 9th International Language Resources and Evaluation Conference (LREC 2014).* ELRA, Paris, pages 1087-1093, http://www.lrec-conf.org/proceedings/lrec2014/pdf/501_Paper.pdf.