

Artificial intelligence, phraseology and constructions: a few missing links

Jean-Pierre Colson
University of Louvain



Contents

1. Introduction: a short recap on AI, DL and Transformer

2. What are the missing links?

3. Experiments

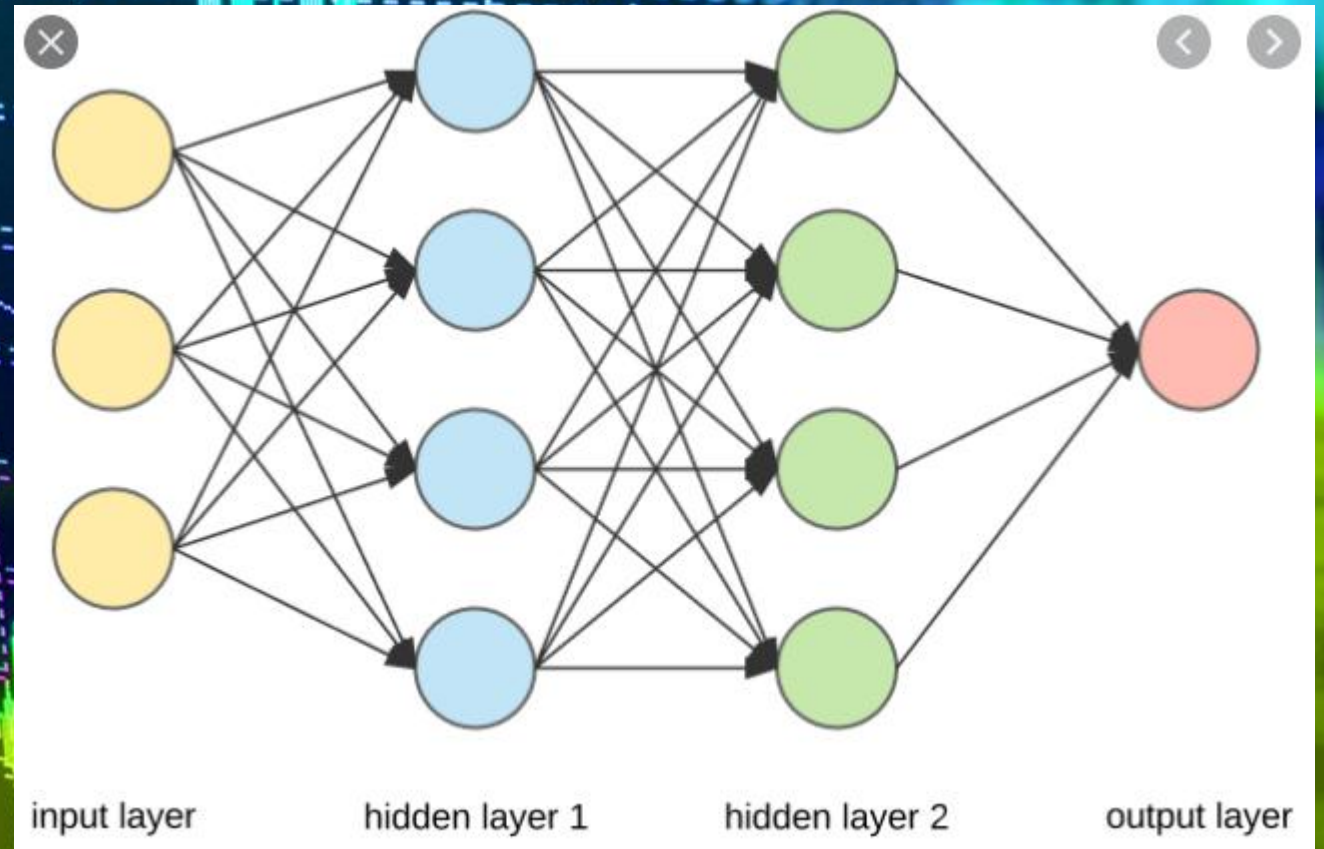
3.1. Is there any CxG in LLMs?

3.2. Fine-tuning LLMs for CxG / Phraseology



1. Intro: AI, DL and Transformer

Machine learning relies on mathematics, especially on Linear Algebra (vectors and matrices) , Calculus (derivatives and gradients), and Probability (variance, regression, distribution)

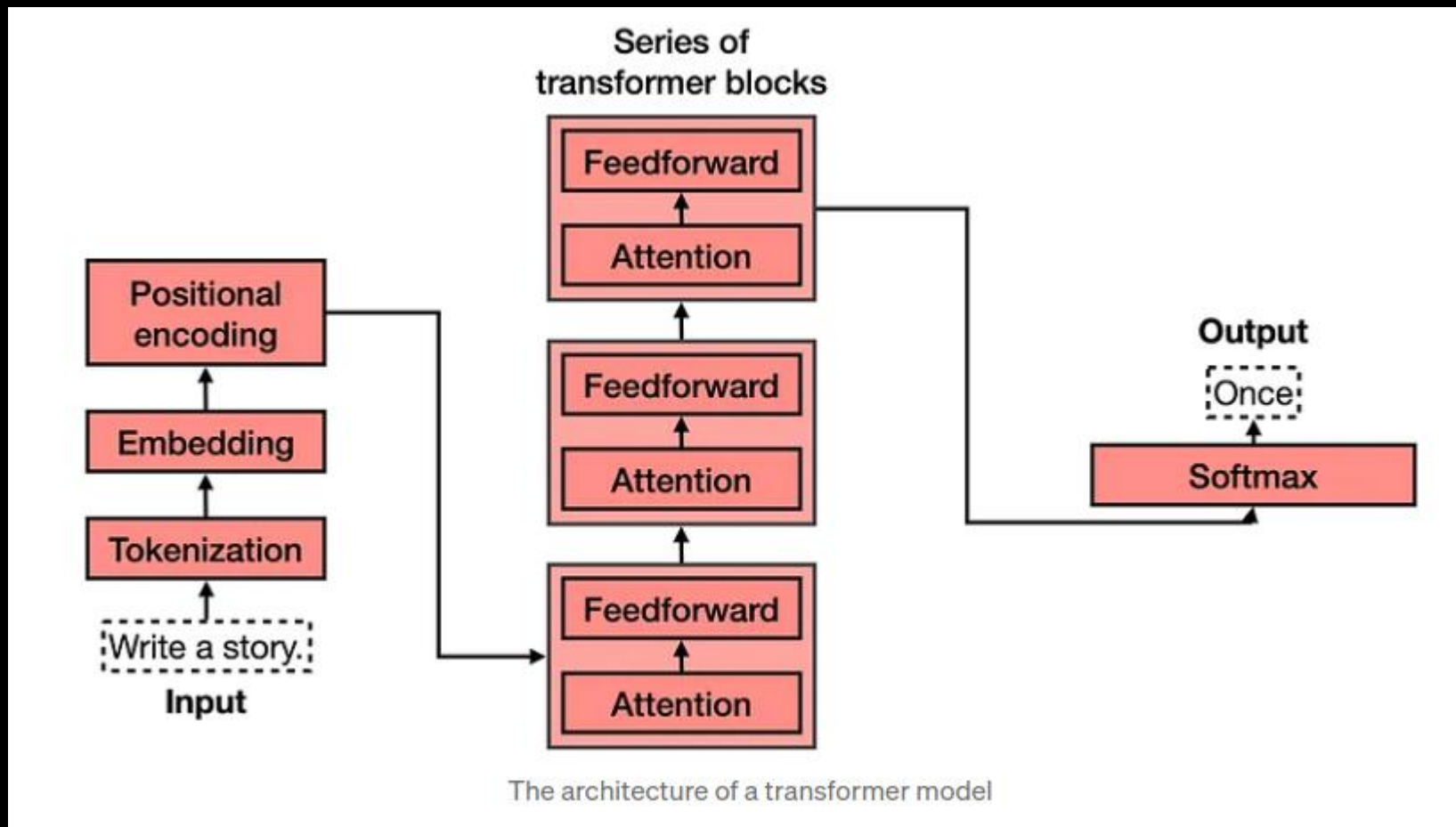


Sequential neural network (multi-layer perceptron)
(Source: Microsoft, CNTK)



- **DL models rely on a space model, mathematical functions and probabilistic choices, but they offer a much more complex and flexible methodology than simple statistical scores (such as log-likelihood, MI, etc.)**
- **DL has contributed to a drastic improvement in Machine Translation (MT) and in the functionalities of Large Language Models (LLMs), such as Bert-like models, GPT, Copilot**
- **All state-of-the-art LLMs and neural machine translation models are based on the Transformer architecture (Vaswani et al. 2017)**





The Transformer architecture (Vaswani et al. 2017). Source: medium.com

Transformer has introduced self-attention, which means from a linguistic point of view that the whole sentence in which the word is used will be considered each time by the model. In addition, the sentences will be trained in parallel, so that relatively broad contexts will be taken into account in the semantic representation, both to the left and to the right of each word.

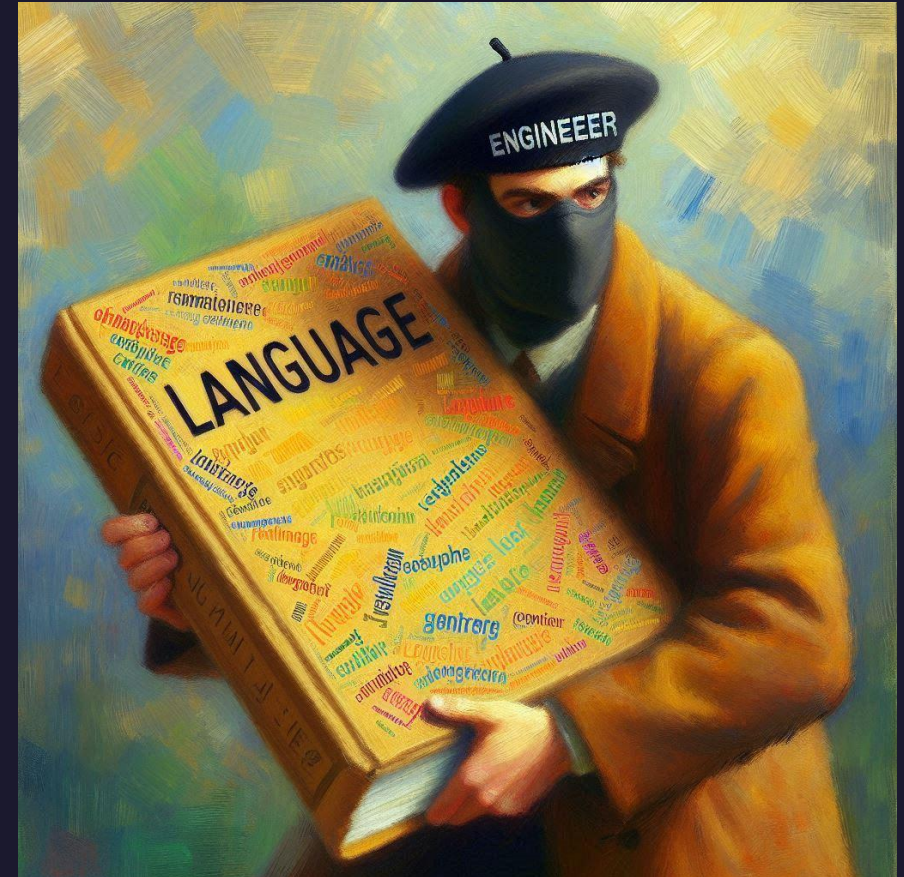


2. What are the (theoretical and practical) missing links?





- As the Transformer architecture takes the broad context into account, it should (in principle) cope very well with **PHRASEOLOGY, FORMULAE** and (maybe) **CONSTRUCTIONS**
- However, this leaves us with a number of **MISSING LINKS**





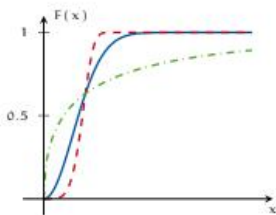
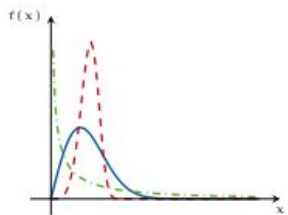
- The **THEORETICAL UNDERPINNINGS** of Transformer are, in the first place, not explicitly stated:
- Vaswani et al. (2017) make no reference to distributional semantics (although they use distributions), nor to information retrieval (although they use linear algebra and metric clusters based on probability)
- In probability, the distribution of continuous values has to take expertise into account, which in this case (language) was hardly the case

Probability | Weibull distribution

Definition: Weibull distribution

The *Weibull distribution* $\text{Weibull}(\alpha, \beta)$ with parameters $\alpha > 0$ and $\beta > 0$ is defined by the Riemann density function

$$f(x) = \begin{cases} \alpha \beta x^{\beta-1} e^{-\alpha x^\beta}, & x > 0 \\ 0, & x \leq 0 \end{cases}$$



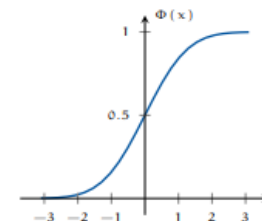
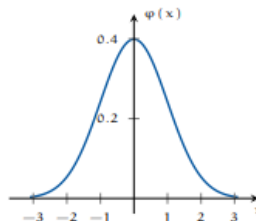
$$F(x) = \begin{cases} 1 - e^{-\alpha x^\beta}, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Probability | Normal|Gaussian distribution

Definition: Normal|Gaussian distribution

The *normal distribution*/*Gaussian distribution* $N(\mu, \sigma^2)$ with parameters $\mu \in \mathbb{R}$ and $\sigma > 0$ is defined by the Riemann density function

$$\varphi_{\mu, \sigma^2}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \quad x \in \mathbb{R}.$$



Φ has only an integral representation

$$\Phi_{\mu, \sigma^2}(x) = \Phi\left(\frac{x-\mu}{\sigma}\right), \quad x \in \mathbb{R}$$



- From a PRACTICAL POINT OF VIEW, there are also a number of missing links
- Is idiomatic information present in Large Language Models based on Transformer (LLMs)?
- Is constructional information present? Is it the same type of information?
- Is a vector approach (cosine similarity or distance) appropriate for measuring phraseology, constructions?
- Is the fine-tuning of these models necessary for phraseology / CxG?
- Can we partly answer those questions by studying the results of Neural Machine Translation (NMT), which is also based on Transformer?





- With respect to all these questions, a corner of the veil has already been lifted by some recent studies



- 
- Jean-Pierre Colson. 2018. From Chinese Word Segmentation to Extraction of Constructions: Two Sides of the Same Algorithmic Coin. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 41–50, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Jean-Pierre Colson. forthcoming. Automatic extraction of constructions and statistical associations in the construction: an exploratory methodology. In: Alexander Ziem, Alexander Willich, and Sascha Michel, *Constructing constructions*. John Benjamins Publishing Company.
- Jussi Karlgren. 2023. High-dimensional vector spaces can accommodate constructional features quite conveniently. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, pages 31–35, Washington, D.C.. Association for Computational Linguistics.
- Ludovica Pannitto and Aurélie Herbelot. 2023. CALaMo: a Constructionist Assessment of Language Models. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, pages 21–30, Washington, D.C.. Association for Computational Linguistics.
- Tim Veenboer and Jelke Bloem. 2023. Using Collostructional Analysis to evaluate BERT's representation of linguistic constructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12937–12951, Toronto, Canada. Association for Computational Linguistics.
- Takashi Wada, Timothy Baldwin, Yuji Matsumoto, and Jey Han Lau. 2022. Unsupervised Lexical Substitution with Decontextualised Embeddings. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4172–4185, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Leonie Weissweiler, Taiqi He, Naoki Otani, David R. Mortensen, Lori Levin, and Hinrich Schütze. 2023. Construction Grammar Provides Unique Insight into Neural Language Models. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, pages 85–95, Washington, D.C.. Association for Computational Linguistics.
- Alexander Ziem and Tim Feldmüller. 2023. "Dimensions of constructional meanings in the German Constructicon: Why collo-profiles matter" *Yearbook of the German Cognitive Linguistics Association*, vol. 11, no. 1, 2023, pp. 203–226. <https://doi.org/10.1515/gcla-2023-0010>.



3. Experiments



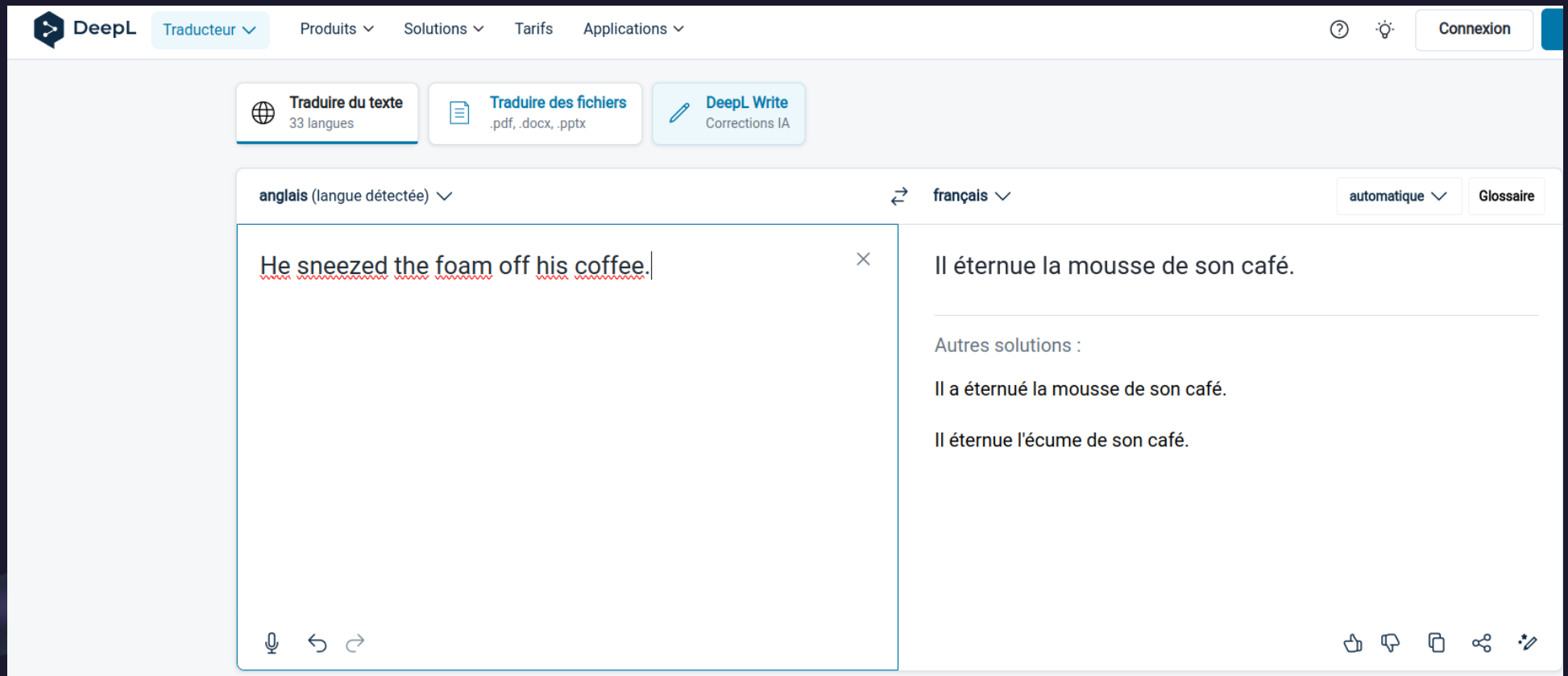


3.1. Is there any CxG in LLMs?

- Different probing techniques have been used in the literature, but there is no conclusive evidence that constructional information is present in pretrained language models : *"Current probing techniques have not yet shown that PLMs are able to adequately handle the meaning of constructions. Assuming that more comprehensive probing techniques will show conclusively that this is not the case, is it due to a lack of data? Or is there a fundamental incompatibility of current architectures and the concept of associating a pattern with a meaning?"* (Weissweiler et al. 2023, p. 92)
- In addition to the varied experiments that have been reported in the literature (see slide 11), it takes just a few minutes to find practical examples illustrating the lack / inaccuracy of constructional information in LLMs



Example 1: wrong translation of constructions by NMT



The screenshot displays the DeepL web interface. At the top, the navigation bar includes the DeepL logo, a 'Traducteur' dropdown menu, and links for 'Produits', 'Solutions', 'Tarifs', and 'Applications'. On the right side of the header are icons for help and settings, and a 'Connexion' button.

The main content area features three primary action buttons: 'Traduire du texte' (33 langues), 'Traduire des fichiers' (.pdf, .docx, .pptx), and 'DeepL Write' (Corrections IA). Below these, the language selection is set to 'anglais (langue détectée)' and 'français'. The translation mode is 'automatique', and a 'Glossaire' link is available.

The input text on the left is 'He sneezed the foam off his coffee.', with each word underlined in red. The primary translation on the right is 'Il éternue la mousse de son café.' Below this, under the heading 'Autres solutions :', two alternative translations are provided: 'Il a éternué la mousse de son café.' and 'Il éternue l'écume de son café.'

At the bottom of the interface, there are icons for voice input, undo, redo, and a set of utility icons (thumbs up/down, copy, share, and a star).

Example 2: using SBERT to measure semantic similarity between sentence embeddings

```
from sentence_transformers import SentenceTransformer, util, InputExample, losses
#Load the model(here we use minilm)
model = SentenceTransformer('all-MiniLM-L6-v2')
```

```
#We get the embeddings by calling model.encode()
emb1 = model.encode("He has disclosed the secret.")
emb2 = model.encode("He has spilled the beans.")
#Get the cosine similarity score between sentences
cos_sim = util.cos_sim(emb1, emb2)
listval = torch.from_numpy(np.array(cos_sim))
valcosinus = listval[0].item()
print(f"The cosine similarity is: {valcosinus}")
```

⇒ The cosine similarity is: 0.4278157651424408



```
#We get the embeddings by calling model.encode()
emb1 = model.encode("He has disclosed the secret.")
emb2 = model.encode("He has closed the door.")
#Get the cosine similarity score between sentences
cos_sim = util.cos_sim(emb1, emb2)
listval = torch.from_numpy(np.array(cos_sim))
valcosinus = listval[0].item()
print(f"The cosine similarity is: {valcosinus}")
```

⇒ The cosine similarity is: 0.41226205229759216

Example 2: using SBERT to measure semantic similarity between sentence embeddings

```
from sentence_transformers import SentenceTransformer, util, InputExample, losses
#Load the model(here we use minilm)
model = SentenceTransformer('all-MiniLM-L6-v2')
```

```
#We get the embeddings by calling model.encode()
emb1 = model.encode("He is eating an apple.")
emb2 = model.encode("He is smiling a little.")
#Get the cosine similarity score between sentences
cos_sim = util.cos_sim(emb1, emb2)
listval = torch.from_numpy(np.array(cos_sim))
valcosinus = listval[0].item()
print(f"The cosine similarity is: {valcosinus}")
```

⇒ The cosine similarity is: 0.31473419070243835



```
#We get the embeddings by calling model.encode()
emb1 = model.encode("He is eating an apple.")
emb2 = model.encode("He is writing a paper.")
#Get the cosine similarity score between sentences
cos_sim = util.cos_sim(emb1, emb2)
listval = torch.from_numpy(np.array(cos_sim))
valcosinus = listval[0].item()
print(f"The cosine similarity is: {valcosinus}")
```

⇒ The cosine similarity is: 0.2268688827753067



- A better understanding (probing) of LLMs for CxG / Phraseology requires a detailed examination of the specific features of Transformer: computation of the dot product between a query vector and a set of key vectors, scaling the result by the square root of the dimensionality of the key vectors (avoiding too big dot products)
- => the method is very similar to cosine distance, cosine similarity

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q . The keys and values are also packed together into matrices K and V . We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.



- BERT model (large): 16 attention heads
- GPT-3: 96 attention heads
- => many subspaces can be studied for every token (word), corresponding to various levels of representation (black box but may include spelling, morphology, syntax, collocations, idioms, CxG?)
- => HOWEVER, sufficient exposure to all those levels crucially depends on the number and nature of examples in the training dataset

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$



- Partial conclusion of 3.1.: *Is there any CxG in LLMs?* The very nature of multi-head attention in Transformer is a weak point for phraseology and for CxG: exposure to a sufficient number of relevant examples is not guaranteed
- The representation of CxG and Phraseology in LLMs is therefore partial and unpredictable
- Weissweiler et al. (2023) come to very similar conclusions:

involved in it. In a transformer-based LM, this additional meaning component cannot be stored in the static embeddings and contextualised through the attention layers, because unlike for MWEs, many constructions have very open slots, so that it is impossible to say that their meaning should somehow be stored with the meaning of the words that may instantiate them. The only place to store constructional information, therefore, remains the model weights, which are much harder to investigate or alter than the model's input, and further probing might reveal that they are unable to store it at all.

- 
- On the bright side, these elements suggest that LLMs are able to yield good results for CxG in the case of frequent tokens / constructions
 - Thus, good results have been obtained by Veenboer & Bloem 2023, Wada et al. 2022, Ziem & Feldmüller 2023

word *Jahr* ‘year’ in the N1 PREP N1 construction we find the following words: *Kalenderjahr* ‘calendar year’, *Jahrs* ‘year (inflected)’, *Jahrzehnt* ‘century’, *Jahres* ‘year (inflected)’ and *Jahrgänge* ‘years / generations’ (cf. Feldmüller et al. in prep. for details). Overall, the operation of our method can be summarized as follows:

- In a first step, we collect as many example sentences for a construction as possible and annotate instances of construction elements.
- Using decontextualized embeddings and a slightly modified implementation of unsupervised lexical substitution (Wada et al., 2022), we let BERT predict possible substitutes for the fillers of the slots, which we filter with respect to their POS tag as well as overlapping word segments.
- Finally, for every sentence, the scores of the top five predictions are summed up to generate a collo-profile.



- However, as Weissweiler et al. 2023 also suggest, a full investigation of CxG in LLMs requires other solutions:
- **Training new models with millions of examples of constructions in various languages (time consuming)**
- **FINE-TUNING** existing models





3.2. Fine-tuning LLMs for CxG / Phraseology

- Fine-tuning is now widely accepted as the recommended AI methodology for extracting very specific information from LLMs. In many cases, it is even better than prompt engineering (e.g. with GPT)

- Fine-tuning allows models to learn from new examples and potentially acquire new knowledge not part of their original training set. In contrast, prompt engineering limits models to the original knowledge and capabilities and cannot introduce new knowledge.

<https://nexla.com/ai-infrastructure/prompt-engineering-vs-fine-tuning>

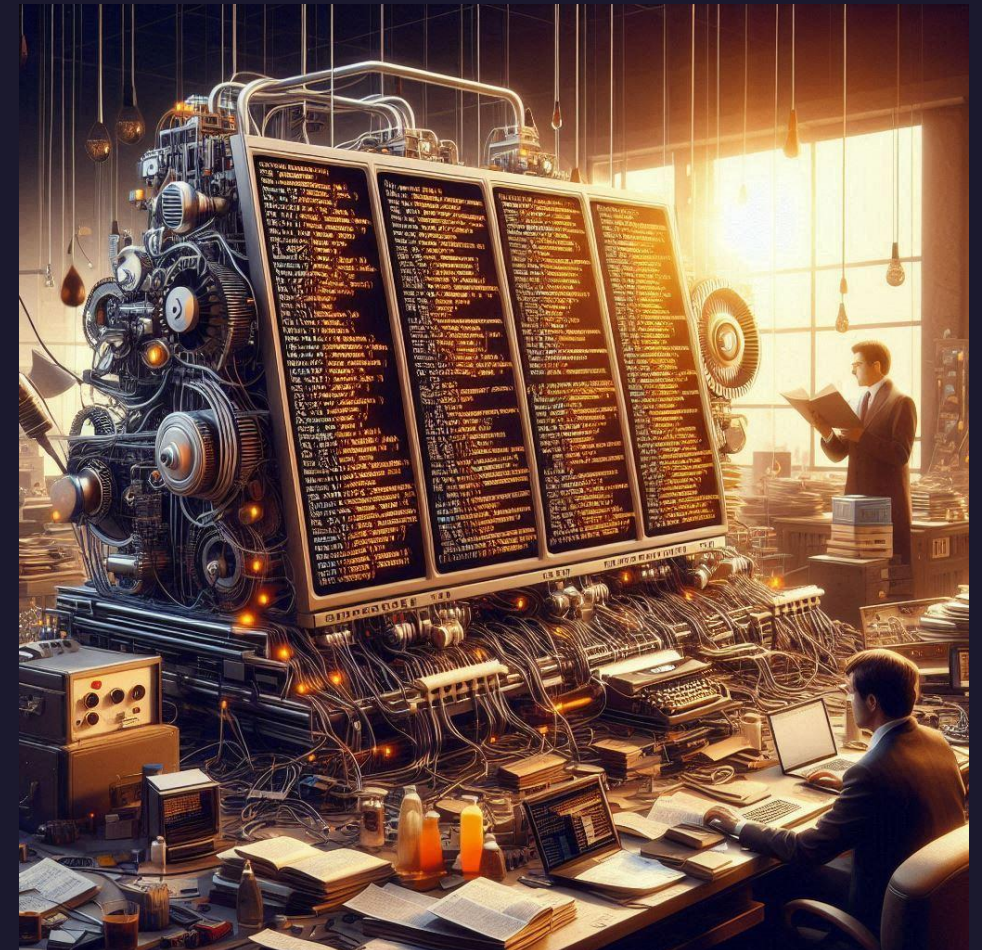


- Although there is some debate on this issue, **CxG and Phraseology** are related to each other, especially for the link between **entrenchment / conventionality / idiomaticity**
- Langacker (2008:32) distinguishes between **entrenchment** and **conventionality**: *“For ease of discussion, I am conflating two parameters that eventually have to be distinguished: entrenchment or unit status (pertaining to a particular speaker) and conventionality (pertaining to a speech community)”*
- In a sense, **ALL CONSTRUCTIONS ARE IDIOMS**: “What may license referring to some constructions as idioms and not others is merely a reflection of the fact that effects of idiomatic variation are best observable in partially schematic complex constructions – however, this does not make them fundamentally different in nature from other constructions (Wulff 2013:285)”





- If CxG and Phraseology are related to each other, AI-based experiments with Phraseology may be useful as well for CxG
- Colson (2022): experiment with automatic extraction of phraseology by fine-tuning of an LLM



METHODOLOGY

In this experiment, 4 general categories of phraseology were chosen: **IDIOM, COLLOCATION, FORMULA, FREE**

```
[ ] #Prepare the data
    categories = ['IDIOM', 'COLLOCATION', 'FORMULA', 'FREE']

    x_train = trainData['text']
    y_train = trainData['labels']
    x_test = testData['text']
    y_test = testData['labels']
```

We chose for an **ELECTRA model**, in which the linguistic masks are not randomly attributed. In BERT, a number of words are arbitrarily replaced by a blank (in order to improve the training of the model). In ELECTRA models, the masks are not blanks but plausible words, according to a statistical selection. There is also a higher number of attention heads in the model. The hypothesis was that such an approach may better capture phraseology.

```
[ ] from transformers import AutoTokenizer, AutoModelForSequenceClassification

    en_tokenizer = AutoTokenizer.from_pretrained("dbmdz/electra-large-discriminator-finetuned-conll03-english")

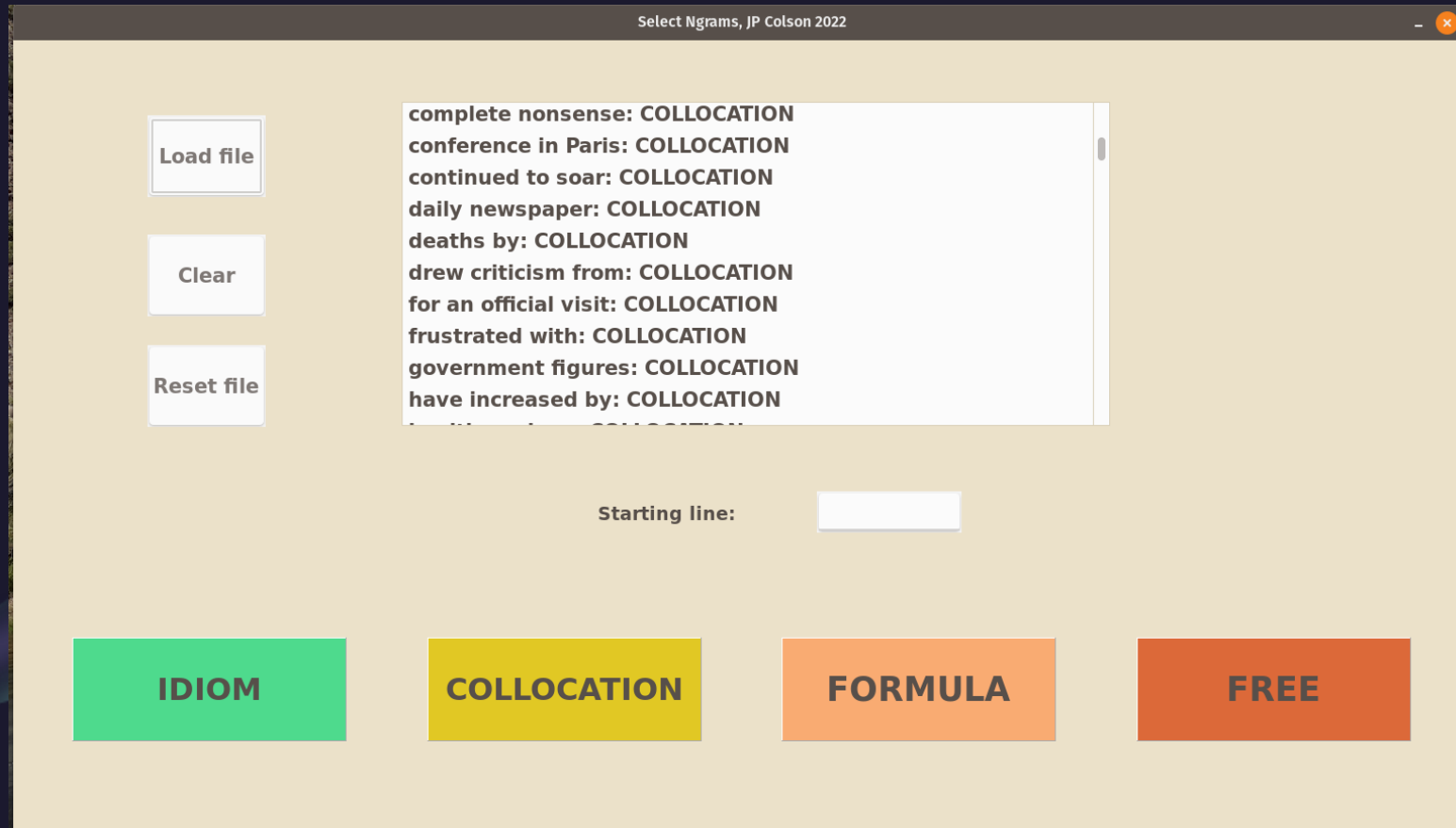
    en_model = AutoModelForSequenceClassification.from_pretrained("dbmdz/electra-large-discriminator-finetuned-conll03-english", num_labels=4)

    en_model.config
```

METHODOLOGY

In order to **train the model**, 5 000 examples of idioms, collocations and formulas were randomly selected from dictionaries and freely available lists

The model was **gradually improved** by checking the wrong results (other language, nonsensical language...); this happened manually, and by recourse to a user interface. Total size of the learning data: 30,000 ngrams



RESULTS

After just a few weeks of training, the model reached excellent results, but as always in AI for language, there is a necessary note of caution

```
[ ] learner.validate(class_names=t.get_classes())
```

	precision	recall	f1-score	support
IDIOM	0.89	0.81	0.85	247
COLLOCATION	0.80	0.86	0.83	423
FORMULA	0.90	0.69	0.78	206
FREE	0.95	0.97	0.96	1632
accuracy			0.91	2508
macro avg	0.88	0.83	0.85	2508
weighted avg	0.91	0.91	0.91	2508

EXPERIMENT WITH EXTRACTION OF Df IDIOMS BY AI



- Colson (2024, in press, Le Français moderne), *Le traitement informatique du défigement : des corpus à l'intelligence artificielle*
- *Defrosted idioms, idiom modification, creative idioms, « défigement », « Remotivation »*: the phenomenon is very common in popular newspapers (tabloids)
- What is at stake is a complex, partly schematic and partly idiomatic construction, often with cultural references



Libération

MACRON ET LES CABINETS DE CONSEIL UNE ADDICTION SALÉE

Un an après «l'affaire McKinsey», la Cour des comptes critique à son tour le recours massif aux consultants privés, notant que les dépenses ont triplé entre 2017 et 2021. Une dérive coûteuse qui flirte parfois avec l'absurde. **PAGES 2-4**

AVIGNON
Carolina
Blanchi,
la drogue
du violeur
en direct

PAGES 23-25

IDÉES
«Atlas des
épidémies»,
l'atout des
cartes

PAGES 18-20



STEFANIE JACQUES

ENQUÊTE
Sur la
Côte d'Azur,
la «saga sans
fin» d'un
hôtel qatari

PAGES 12-14

THE ART OF Wallace & Gromit



VENGEANCE MOST FOWL

RICHARD HANSOM
FOREWORD BY PETER KAY

1604 Murther most foule, as in the best it is.
W. Shakespeare, *Hamlet* i. v. 27 ...



Bob Dylan - Murder Most Foul (Official Audio)

YouTube · BobDylanVEVO

27 mars 2020

EXPERIMENT WITH EXTRACTION OF Df IDIOMS BY AI

- In spite of the complexity of such constructions (partly schematic, partly idiomatic, playing with spelling, pronunciation, cultural references), it was possible (Colson 2024) to reach promising results with a limited training dataset of 400 examples
- The crucial point was **detailed fine-tuning (few-shot learning)**

	précision	rappel	score f1
FIGEMENT	0,75	1,00	0,86
DÉFIGEMENT	1,00	0,54	0,70
AUTRE	0,83	0,71	0,77
Moyenne pondérée ⁵	0,84	0,80	0,79

EXPERIMENT WITH CxG (Colson 2024): collo-profiles with AI

- Inspired by the examples in Ziem & Feldmüller (2023), the English construction **N₁PREP_N₁** was tested for the extraction of collo-profiles by means of AI
- Methodology:
 - Tagging of a portion of 120 million tokens from corpus ukWac with Stanford PoS Tagger
 - Extraction (perl script, regex) of patterns **N₁PREP_N₁** with freq=2 as threshold
 - All patterns were submitted to the model used for the extraction of phraseology (Colson 2022), with the categories IDIOM, COLLOCATION, FORMULA, FREE
 - The results confirm the idiomatic and semi-idiomatic character of many collo-profiles (see attachment)

BY WAY OF CONCLUSION

- Provided that effective fine-tuning is used, it should be possible to compile similar models for the automatic identification of constructions
- For every construction, 400-500 examples would be necessary, but this task can be largely automated by patterns and automated scripts
- Once the model has been trained for a list of constructions, it should be able to identify constructions in text, with useful applications for language learning or translation
- In addition, pushing LLMs to their linguistic limits is always useful for AI in general

Vision Transformer

Introduced by Dosovitskiy et al. in [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#)

The **Vision Transformer**, or ViT, is a model for image classification that employs a [Transformer](#)-like architecture over patches of the image. An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard [Transformer](#) encoder. In order to perform classification, the standard approach of adding an extra learnable "classification token" to the sequence is used.

(<https://paperswithcode.com/method/vision-transformer>)