One dataset, many corpora

Problems of scientific validity in corpora and corpus-derived results

Matteo Di Cristofaro

UniMoRe

29/02/2024 @EHU CRG #11

Preface

Statistics

> "Some people think of statistics as being only the very last step of the empirical process. You design your study, you collect your data, then you perform a statistical analysis of the data. This is a narrow view of statistics. [...] In particular, I view the process of getting the data in shape for an analysis as part of your actual analysis. Thus, what people talk of as ‘preprocessing’ or ‘data wrangling’ is an integral part of statistics. In fact, most of your time during an actual analysis will be spent on wrangling with the data." (pag. 1; emphasis in the original)

Winter, Bodo. Statistics for Linguists: An Introduction Using R. 1st ed. New York, NY: Routledge, 2019.

~~Statistics~~ Corpus Linguistics

> Some people think of ~~statistics~~ **corpus linguistics** as being only the very last step of the empirical process. You design your study, you collect your data, *then* you perform a ~~statistical~~ **corpus** analysis of the data. This is a narrow view of ~~statistics~~ **corpus linguistics**. […] In particular, I view the process of getting the data in shape for an analysis as part of your actual analysis. Thus, what people talk of as ‘preprocessing’ or ‘data wrangling’ is an integral part of ~~statistics~~ **corpus linguistics**. In fact, most of your time during an actual analysis will be spent on wrangling with the data.

Index

Index

Prologue

Some background details

The "role" of a corpus

We can consider a **corpus as the ground-truth of our analysis**: a digital object that *represents* (or: a *digital representation*) of the language data we wish to investigate. As such, it must be as faithful as possible to the source data.

The "role" of a corpus

We can consider a **corpus as the ground-truth of our analysis**: a digital object that *represents* (or: a *digital representation*) of the language data we wish to investigate. As such, it must be as faithful as possible to the source data.

But what does *faithful* mean?

Digital technicalities

Digital technicalities as "those notions and mechanisms that – while not classically associated with natural language – are

1. foundational of the digital environments in which language production and exchanges occur and
2. at the core of the techniques that are used to produce, collect, and process the focus of investigation, that is, digital textual data"

(Di Cristofaro 2023:4)

Two notes

- **The icon ⚗ means that the results/considerations are preliminary!**
- **All the scripts and procedures (along with the data) discussed are/will be available online at https://catlism.github.io**

From theory to practice: two collaborative tasks

A practical example: three corpora

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

A practical example: three corpora

	A	**B**	**C**
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

- **Type**: Unit of measure for the single form of each element in a corpus (usually, a *wordform*)

A practical example: three corpora

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

- **Type**: Unit of measure for the single form of each element in a corpus (usually, a *wordform*)
- **Token**: Unit of measure for each occurrence of each element in a corpus (usually, a *word*)

A practical example: three corpora

	A	**B**	**C**
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

- **Type**: Unit of measure for the single form of each element in a corpus (usually, a *wordform*)
- **Token**: Unit of measure for each occurrence of each element in a corpus (usually, a *word*)
- **File ≠ Document**

A practical example: three corpora

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

- **Type**: Unit of measure for the single form of each element in a corpus (usually, a *wordform*)
- **Token**: Unit of measure for each occurrence of each element in a corpus (usually, a *word*)
- **File ≠ Document**
    - **File**: Unit of measure for a single object of the corpus *from the perspective of the computer*
    - **Document**: Unit of measure for a single object of the corpus *from the perspective of humans*

A practical example: three corpora

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

- **Type**: Unit of measure for the single form of each element in a corpus (usually, a *wordform*)
- **Token**: Unit of measure for each occurrence of each element in a corpus (usually, a *word*)
- **File ≠ Document**
    - **File**: Unit of measure for a single object of the corpus *from the perspective of the computer*
    - **Document**: Unit of measure for a single object of the corpus *from the perspective of humans*

Judging from the details above, how would you evaluate the following statements?

- the three corpora were compiled from the same files
- the three corpora were compiled from the same documents

A practical example: three corpora

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

- **Type**: Unit of measure for the single form of each element in a corpus (usually, a *wordform*)
- **Token**: Unit of measure for each occurrence of each element in a corpus (usually, a *word*)
- **File ≠ Document**
    - **File**: Unit of measure for a single object of the corpus *from the perspective of the computer*
    - **Document**: Unit of measure for a single object of the corpus *from the perspective of humans*

Judging from the details above, how would you evaluate the following statements?

- the three corpora were compiled from the same files **False OR True** (it depends)
- the three corpora were compiled from the same documents **True**

Counting

The basics of corpus linguistics

Counting

n.	sentence	tokens	types
1.	It's Friday! And on Friday I'm in love		
2.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love		
3.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love 💗👯		
4.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love💗		

Counting

n.	sentence	tokens	types
1.	It's Friday! And on Friday I'm in love	10	9
2.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love		
3.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love 💗👯‍♀️		
4.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love💗		

Counting

n.	sentence	tokens	types
1.	It's Friday! And on Friday I'm in love	10	9
2.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love	10	10
3.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love 💗👯‍♀️		
4.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love💗		

Counting

n.	sentence	tokens	types
1.	It's Friday! And on Friday I'm in love	10	9
2.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love	10	10
3.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love 💗👯	13	13*
4.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love💗		

*: AntConc does not process emojis, while other tools do (e.g. SketchEngine)

Counting

n.	sentence	tokens	types
1.	It's Friday! And on Friday I'm in love	10	9
2.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love	10	10
3.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love 💗👯	13	13*
4.	It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love💗	10	10*

*: AntConc does not process emojis, while other tools do (e.g. SketchEngine)

An "astronomical" detour

An "astronomical" detour: *perytons*

> The peryton is a mythological hybrid animal combining the physical features of a stag and a bird. The peryton was first named by Jorge Luis Borges in his 1957 Book of Imaginary Beings, using the fictional device of a supposedly long-lost medieval manuscript. (*Wikipedia*)

Wikipedia

From *astronomical* perytons...

- **2007**: At Parkes Observatory (Australia), researchers find signals of "unknown extragalactic origin" (Burkes-Spolaor et al. 2010:1) among the data collected in 2001 for the project meant to identify signals of extragalactic provenance; further analysis of historical data shows the signal had always been present ever since the observatory was "switched on" in 1998.

From *astronomical* perytons...

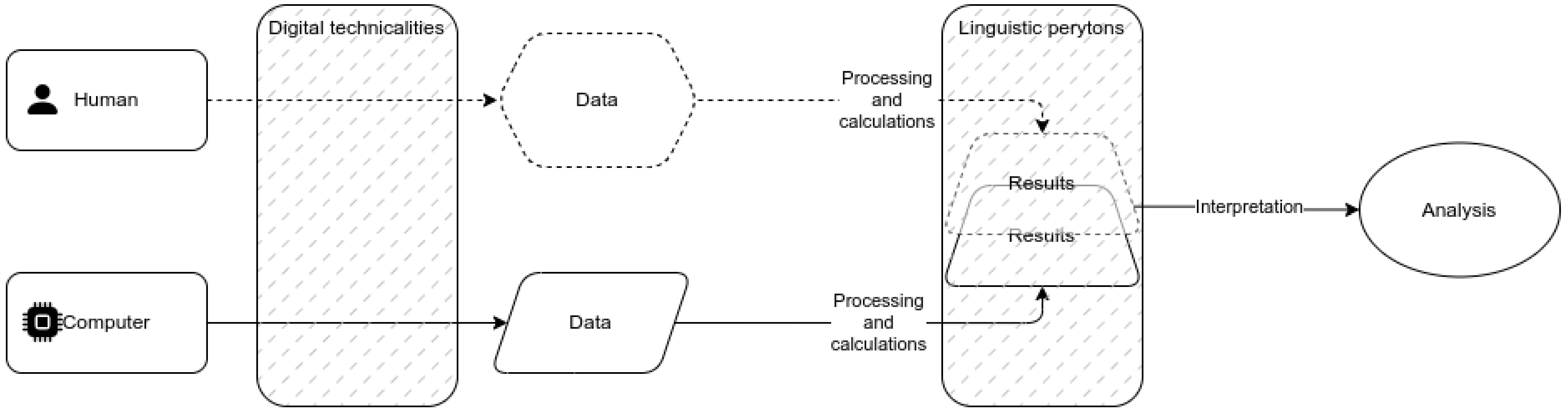
- **2007**: At Parkes Observatory (Australia), researchers find signals of "unknown extragalactic origin" (Burkes-Spolaor et al. 2010:1) among the data collected in 2001 for the project meant to identify signals of extragalactic provenance; further analysis of historical data shows the signal had always been present ever since the observatory was "switched on" in 1998.
- **2015**: Researchers find that the source of the mysterious perytons is...

From *astronomical* perytons...

- **2007**: At Parkes Observatory (Australia), researchers find signals of "unknown extragalactic origin" (Burkes-Spolaor et al. 2010:1) among the data collected in 2001 for the project meant to identify signals of extragalactic provenance; further analysis of historical data shows the signal had always been present ever since the observatory was "switched on" in 1998.
- **2015**: Researchers find that the source of the mysterious perytons is an old microwave still in use in the nearby pantry. "Subsequent tests revealed that a peryton can be generated at 1.4 GHz when a microwave oven door is opened prematurely and the telescope is at an appropriate relative angle. Radio emission escaping from microwave ovens during the magnetron shutdown phase neatly explains all of the observed properties of the peryton signals." (Petroff et al. 2015:3933)

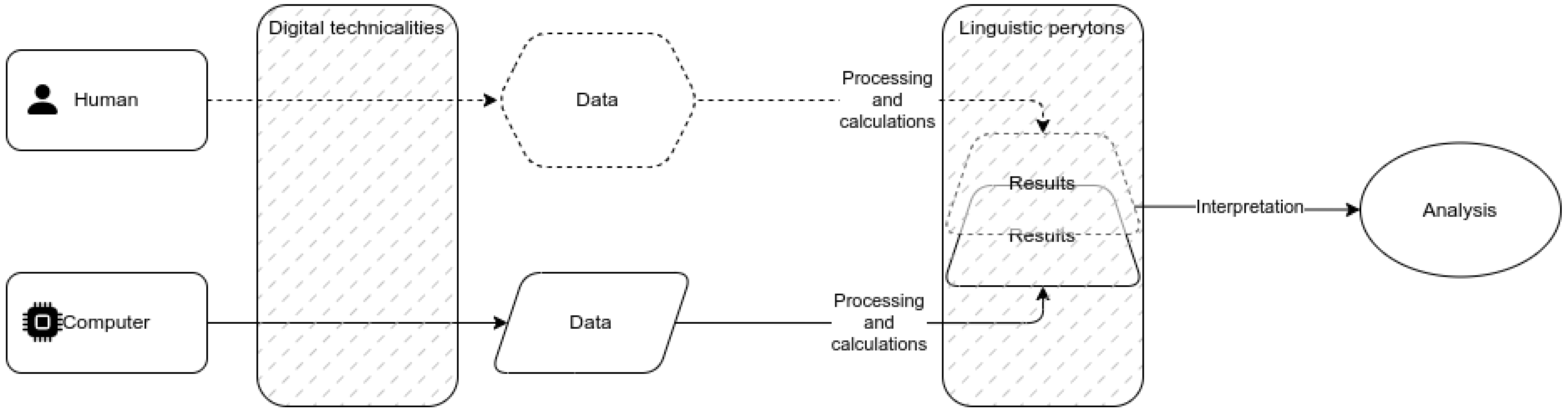
...to *linguistic* perytons

Linguistic perytons: phenomena *present* in the object of our analysis (corpus) due to the combination of our actions and the instruments we employ, but *absent* in the object we consider to be the source of our corpus.



...to *linguistic* perytons

Linguistic perytons: phenomena *present* in the object of our analysis (corpus) due to the combination of our actions and the instruments we employ, but *absent* in the object we consider to be the source of our corpus.



⚠ *Linguistic perytons* may appear - often unnoticed - in a corpus as a result of us (humans) **not** doing something: currently available corpus tools - whether they are meant to collect or analyse a corpus - do not in fact operate any *pre-processing* procedure to avoid (a number of) them!

Hold on: is it really *that* serious?

Is it serious?

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

	A-B	B-C	A-C
Tokens	0.122686%	5.37413%	5.49673%
Types	0.0629921%	0.984371%	1.04736%
Files	0%	0%	0%

Is it serious?

	A	B	C
Tokens	177,581	177,799	187,618
Types	17,468	17,457	17,286
Files	13	13	13

	A-B	B-C	A-C
Tokens	0.122686%	5.37413%	5.49673%
Types	0.0629921%	0.984371%	1.04736%
Files	0%	0%	0%

Yes, it is

Despite any potential *statistical significance* we are faced with an *epistemological* problem!
We (humans) may inadvertently produce *linguistic perytons* whenever we do corpus linguistics, effectively making the corpus a non-faithful representation of the source data. As such:

- different researchers may be working on different representations of a source dataset, while thinking they are all working on a single one;
- any result/finding is tied to *our* specific representation of the source dataset, effectively impinging on its scientific validity;
- we may be impeding the reproducibility of our research

Æá ÆÖóÞ¢áñ, can you read me?

In search for the (not so) mythical *linguistic perytons*

A text is a text?

While textual contents in digital format may appear to be "always the same", textual data can be saved in many different formats (.txt, .docx, .pdf, . rtf, etc…).
Some of these formats are commonly referred to as "plain-text", meaning that their contents can be read with any text editor, no matter what version, operating system, tool, language, etc.

However…

There Ain’t No Such Thing As Plain Text

Spolsky 2003

There Ain’t No Such Thing As Plain Text

Similar to any other type of data, text is stored by computers as sequences of bits (the `0`s and `1`s); when a file containing text is processed, the computer reads these sequences of numbers and interprets them according to a **lookup table** (aka **code space**) that maps each sequence to a **character** (aka **code point**).
A distinction must here be made between a **character** and a **glyph**:

- **character** the “abstract[…] representation[…] of the smallest components of written language” (Aliprand 2018:4663)
- **glyph**: the visual representation of a character, dependent on the software being used to display the text

The relationship between a code point and its relative character is defined by an **encoding**, i.e. the set of rules that links a sequence of bits to its corresponding code point and character. This is similar to a *dictionary*.

Encodings, characters, glyphs

glyph	character	character name
A	U+0041	LATIN CAPITAL LETTER A
B	U+0042	LATIN CAPITAL LETTER B
C	U+0043	LATIN CAPITAL LETTER C
D	U+0044	LATIN CAPITAL LETTER D
🐘	U+1F418	Elephant
🧘	U+1F9D8	Person in lotus position

glyphs	BAD
bits	01000010 01000001 01000100
characters	U+0042U+0041U+0044

Converting characters into glyphs and viceversa

Encodings

There are more than 200 encodings: some designed for Latin script, others for different alphabets. Fortunately, the most widespread encoding on the web (UTF-8) supports all known alphabets, plus emoji, and a host of other symbols - and is the *de-facto* standard encoding for the web (and for digital texts in general).

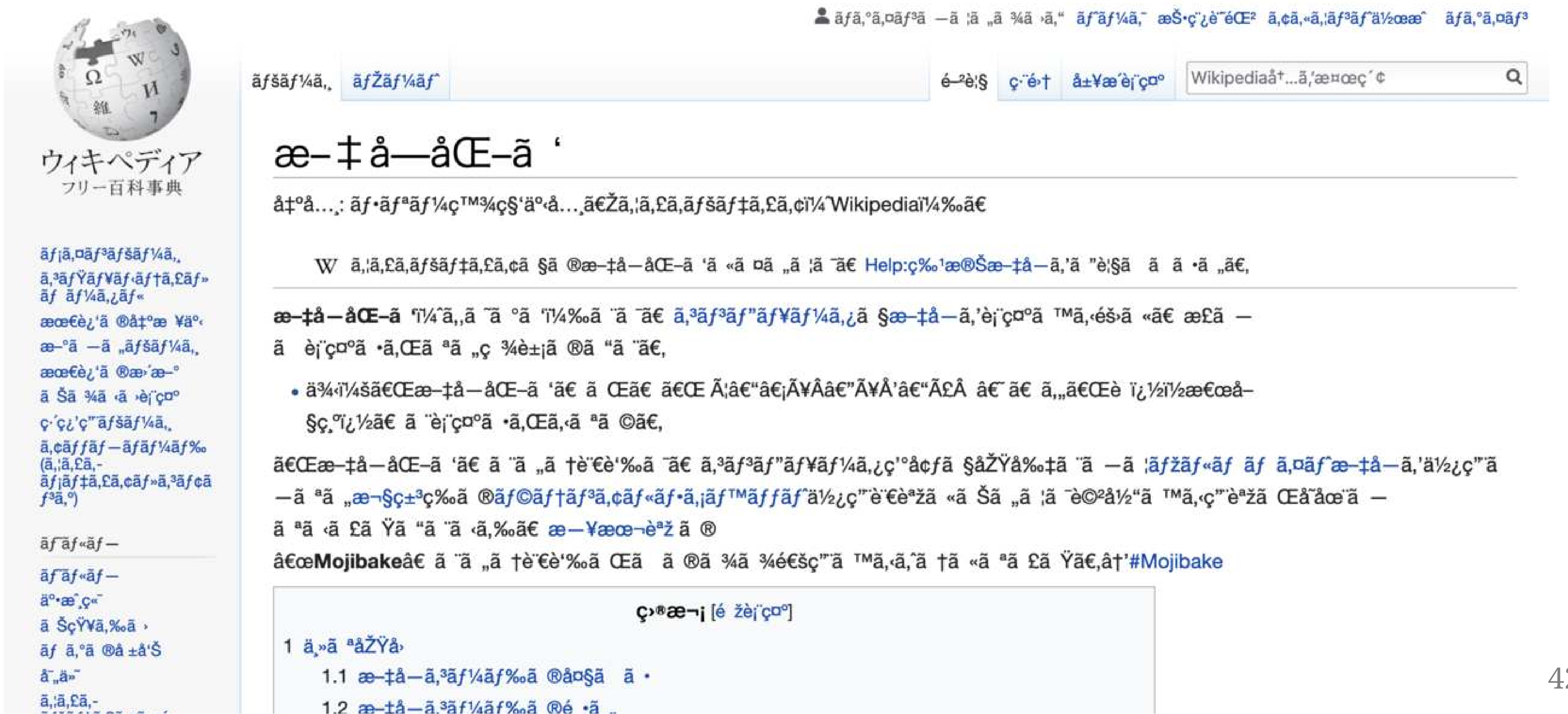
Beware, however: an incorrect encoding declaration is tantamount to rendering the text incomprehensible to the human being who has to read it - for the computer, on the other hand, we are still talking about bits.

An example?

⚠ Most important of all however is that automatic encoding detection "is, at best, an imprecise operation using statistics and heuristics" (Unicode Inc. 2016).

Mojibake!

"moji bah-keh" = *character transformation* in Japanese



ãƒã‚°ã‚¤ãƒ³ã —ã ¦ã „ã ¾ã ›ã‚“ ãƒˆãƒ¼ã‚¯ æŠ•ç¨¿è¨˜éŒ² ã‚¢ã‚«ã‚¦ãƒ³ãƒˆä½œæˆ ãƒã‚°ã‚¤ãƒ³

ãƒšãƒ¼ã‚¸ ãƒŽãƒ¼ãƒˆ é–²è¦§ ç·¨é›† å±¥æ´è¡¨ç¤º Wikipediaå†…ã‚’æ¤œç´¢

ãƒ¡ã‚¤ãƒ³ãƒšãƒ¼ã‚¸
ã‚³ãƒŸãƒ¥ãƒ‹ãƒ†ã‚£ãƒ»ãƒ ãƒ¼ã‚¿ãƒ«
æœ€è¿'ã ®å‡ºæ ¥äº‹
æ–°ã —ã „ãƒšãƒ¼ã‚¸
æœ€è¿'ã ®æ›´æ–°
ã Šã ¾ã ‹ã ›è¡¨ç¤º
ç·´ç¿'ç"¨ãƒšãƒ¼ã‚¸
ã‚¢ãƒƒãƒ—ãƒãƒ¼ãƒ‰ (ã‚¦ã‚£ã‚-ãƒ¡ãƒ‡ã‚£ã‚¢ãƒ»ã‚³ãƒ¢ãƒ³ã‚º)

ãƒ˜ãƒ«ãƒ—

ãƒ˜ãƒ«ãƒ—
äº•æ¸¸ç«¯
ã ŠçŸ¥ã‚‰ã ›
ãƒ ã‚°ã ®å ±å'Š
å¯„ä»˜
ã‚¦ã‚£ã‚-

æ–‡å—åŒ–ã ‘

å‡ºå…¸: ãƒ•ãƒªãƒ¼ç™¾ç§'äº‹å…¸ã€Žã‚¦ã‚£ã‚ãƒšãƒ‡ã‚£ã‚¢ï¼ˆWikipediaï¼‰ã€

W ã‚¦ã‚£ã‚ãƒšãƒ‡ã‚£ã‚¢ã §ã ®æ–‡å—åŒ–ã ‘ã «ã ¤ã „ã ¦ã ¯ã€ Help:ç‰¹æ®Šæ–‡å—ã‚'ã "è¦§ã ã ã •ã „ã€‚

æ–‡å—åŒ–ã ‘ï¼ˆã‚‚ã ˜ã °ã ‘ï¼‰ã ¨ã ¯ã€ ã‚³ãƒ³ãƒ"ãƒ¥ãƒ¼ã‚¿ã §æ–‡å—ã‚'è¡¨ç¤ºã ™ã‚‹éš›ã «ã€ æ£ã —ã è¡¨ç¤ºã •ã‚Œã ªã „ç ¾è±¡ã ®ã "ã ¨ã€‚

- ä¾‹ï¼šã€Œæ–‡å—åŒ–ã ‘ã€ ã Œã€ ã€Œ Ã¦â€"â€¡Ã¥Ââ€"Ã¥Å'â€"ÃƒÂ â€˜ ã€ ã‚„ã€Œè ï¿½ï¿½æ€œå–§ç"Ÿï¿½ã€ ã ¨è¡¨ç¤ºã •ã‚Œã‚‹ã ªã ©ã€‚

ã€Œæ–‡å—åŒ–ã ‘ã€ ã ¨ã „ã †è¨€è'‰ã ¯ã€ ã‚³ãƒ³ãƒ"ãƒ¥ãƒ¼ã‚¿ç'°å¢ƒã §åŽŸå‰‡ã ¨ã —ã ¦ãƒžãƒ«ãƒ ãƒ ã‚¤ãƒˆæ–‡å—ã‚'ä½¿ç"¨ã —ã ªã „æ¬§ç±³ç‰ã ®ãƒ©ãƒ†ãƒ³ã‚¢ãƒ«ãƒ•ã‚¡ãƒ™ãƒƒãƒˆä½¿ç"¨è¨€èªžã «ã Šã „ã ¦ã ¯è©²å½"ã ™ã‚‹ç"¨èªžã Œå˜åœ¨ã —ã ªã ‹ã £ã Ÿã "ã ¨ã ‹ã‚‰ã€ æ—¥æœ¬èªž ã ® â€œ**Mojibake**â€ ã ¨ã „ã †è¨€è'‰ã Œ ã ®ã ¾ã ¾é€šç"¨ã ™ã‚‹ã‚ˆã †ã «ã ªã £ã Ÿã€‚â†'#Mojibake

ç›®æ¬¡ [é žè¡¨ç¤º]

1 ä¸»ã ªåŽŸå›
  1.1 æ–‡å—ã‚³ãƒ¼ãƒ‰ã ®å¤§ã ã •
  1.2 æ–‡å—ã‚³ãƒ¼ãƒ‰ã ®é •ã „

Linguistic peryton #1: "symbols"

"Symbols"

Rather than "symbols", we should talk about *specific symbols in specific blocks of specific planes*, such as the Mathematical Alphanumeric Symbols.
These appear in our data as glyphs that a human recognises as related to one or more specific letter/character, but that a computer 'sees' as different from what we humans consider equivalent: corpus linguistics tools operate different decisions on how to interpret them and group them, and this may cause some serious issues. Other "symbols" include ligatures, IPA alphabet, and much more.

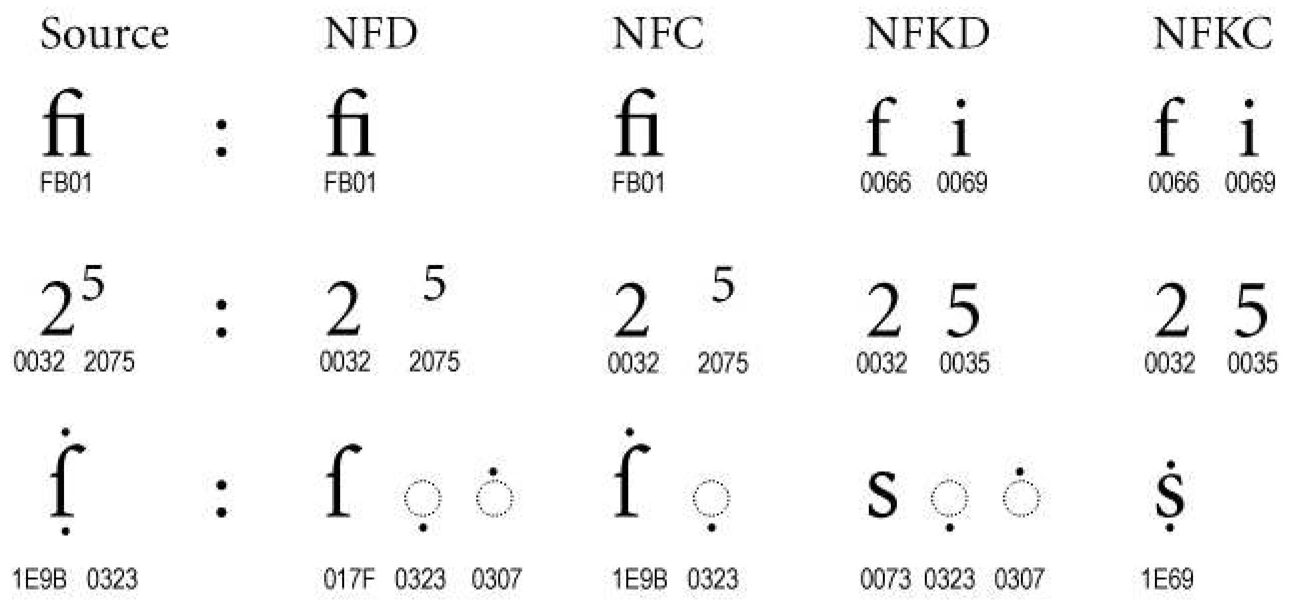
symbol/glyph	**character name**	**code**	**block**	**plane**
ﬀ	Latin Small Ligature Ff	`U+FB00`	Alphabetic Presentation Forms	Basic Multilingual Plane
ﬁ	Latin Small Ligature Fi	`U+FB01`	Alphabetic Presentation Forms	Basic Multilingual Plane
𝔽	Mathematical Double-Struck Capital F	`U+1D53D`	Mathematical Alphanumeric Symbols	Supplementary Multilingual Plane
f	Latin Small Letter f	`U+0066`	Basic Latin	Basic Multilingual Plane
F	Latin Capital Letter F	`U+0046`	Basic Latin	Basic Multilingual Plane

Unicode Normalization Forms

> "The Unicode Standard defines two formal types of equivalence between characters: *canonical* equivalence and *compatibility* equivalence. **Canonical** equivalence is a fundamental equivalency between characters or sequences of characters which represent the same abstract character, and which when correctly displayed should always have the same visual appearance and behavior. […] **Compatibility** equivalence is a weaker type of equivalence between characters or sequences of characters which represent the same abstract character (or sequence of abstract characters), but which may have distinct visual appearances or behaviors." (Unicode Inc. 2023)
>
> "**Unicode Normalization Forms** are formally defined normalizations of Unicode strings which make it possible to determine whether any two Unicode strings are equivalent to each other. Depending on the particular Unicode Normalization Form, that equivalence can either be a canonical equivalence or a compatibility equivalence." (Unicode Inc. 2023)

Form	Description
Normalization Form D (**NFD**)	Canonical Decomposition
Normalization Form C (**NFC**)	Canonical Decomposition, followed by Canonical Composition
Normalization Form KD (**NFKD**)	Compatibility Decomposition
Normalization Form KC (**NFKC**)	Compatibility Decomposition, followed by Canonical Composition

Unicode Normalization Forms /2



So normalising the previously discussed 𝔽𝕣𝕚𝕕𝕒𝕪 with NFKC becomes `Friday`; good!

⚠ However **NFKC is destructive**, meaning that it is not possible to reconstruct the original characters (cf. W3C 2021)!

Normalization? We got you covered!

Normalization is nothing new in corpus linguistics; think of e.g. spelling variations. As such, solutions have been devised to:

1. Make normalization feasible and (semi-)automatic
2. Preserve the original spelling while including its normalized version

VARD (Variant Detector; Baron and Rayson 2008) is one such tool that allows to produce a normalized corpus in XML format

```
<normalized orig="idk">I don't know</normalized>,
<normalized orig="wud"> would</normalized> you say this is correct?
```

Normalization? We got you covered!

Normalization is nothing new in corpus linguistics; think of e.g. spelling variations. As such, solutions have been devised to:

1. Make normalization feasible and (semi-)automatic
2. Preserve the original spelling while including its normalized version

VARD (Variant Detector; Baron and Rayson 2008) is one such tool that allows to produce a normalized corpus in XML format

```
<normalized orig="idk">I don't know</normalized>,
<normalized orig="wud"> would</normalized> you say this is correct?
```

🧪 So why not use the same approach for "symbols"?

```
It's <normalized type="nfkc" orig="𝔽𝕣𝕚𝕕𝕒𝕪">Friday</normalized>! And on Friday I'm in love
```

Linguistic peryton #2: emojis

Emojis

Even emojis are part of Unicode: they are characters represented by `code points`, which are then turned into *glyphs* when the text is displayed.

Here's the reference list for emoji in Unicode

For example

> Back with my babies \ud83d\ude0d

Let's decode it

Emojis /2

The increasing number of emojis is regulated by different versions of the Unicode standard, meaning that emojis are in constant evolution.
One such evolution is the ability of defining sex or skin tones for a number of emojis. The latest is v15.1; here's the changelog.

emoji	character name	emojis combination	characters names
👩🏿	Woman: Dark Skin Tone	👩 + 🏿	Woman + Dark Skin Tone
👩🏼	Woman: Medium-Light Skin Tone	👩 + 🏼	Woman + Medium Light Skin Tone
👯‍♀️	Women with Bunny Ears	👯 + ♀	People with Bunny Ears + Female Sign

Emojis /3: Behind the scenes

emojis	character code
👩🏿	`U+d83dU+dc69U+d83cU+dfff`
👩	`U+d83dU+dc69`
🏿	`U+d83cU+dfff`
👩🏼	`U+d83dU+dc69U+d83cU+dffc`
👩	`U+d83dU+dc69`
🏼	`U+d83cU+dffc`
👯‍♀️	`U+d83dU+dc6fU+200dU+2640U+fe0f`
👯	`U+d83dU+dc6f`
♀️	`U+2640U+fe0f`

Emojis /3: Behind the scenes

emojis	character code
👩🏿	`U+d83dU+dc69`**`U+d83cU+dfff`**
👩	`U+d83dU+dc69`
🏿	`U+d83cU+dfff`
👩🏼	`U+d83dU+dc69`**`U+d83cU+dffc`**
👩	`U+d83dU+dc69`
🏼	`U+d83cU+dffc`
👯‍♀️	`U+d83dU+dc6f`**`U+200d`**`U+2640U+fe0f`
👯	`U+d83dU+dc6f`
♀️	`U+2640U+fe0f`

- **`U+200d`** (or `\u200d` using different notation): aka **Z**ero-**W**idth **J**oiner (**ZWJ**), is used to combine emojis and output a modified version where e.g. the gender is different
- Emoji Modifier Sequences: **`U+d83cU+dfff`** (equivalent to `U+1f3ff`) or **`U+d83cU+dffc`** (equiv. to `U+1f3fc`) are used to modify skin tones

Transliterating emojis

Emojis may be included in a corpus as transliterated elements, meaning that what is "accessible" to the corpus tool is not the emoji itself, but rather the unique textual description that identifies said emoji. These descriptions are defined by the Unicode Consortium (CLDR; see Full Emoji List), and can be e.g. included as enclosed by curly brackets, with an underscore separating each word of the description (cf. Di Cristofaro 2023:283-285).
Even though corpus tools that support emojis (e.g. SketchEngine) do not correctly identify the ones with **Modifier Sequences** and/or **ZWJ**, the Python module `emoji` can be used to pre-process our data correctly (and its online documentation is very rich)!

Transliterating emojis

Emojis may be included in a corpus as transliterated elements, meaning that what is "accessible" to the corpus tool is not the emoji itself, but rather the unique textual description that identifies said emoji. These descriptions are defined by the Unicode Consortium (CLDR; see Full Emoji List), and can be e.g. included as enclosed by curly brackets, with an underscore separating each word of the description (cf. Di Cristofaro 2023:283-285).
So for example, the emoji 🤗 would be rendered as

> {smiling_face_with_open_hands}

to allow the corpus tool to treat it as one single element (token); and using XML it may be enclosed in a `<w>` tag as exemplified below.

```xml
<w pos="FU" wclass="X" hw="smiling_face_with_open_hands" usas="Z99">
  {smiling_face_with_open_hands}
</w>
```

Transliterating emojis

Emojis may be included in a corpus as transliterated elements, meaning that what is "accessible" to the corpus tool is not the emoji itself, but rather the unique textual description that identifies said emoji. These descriptions are defined by the Unicode Consortium (CLDR; see Full Emoji List), and can be e.g. included as enclosed by curly brackets, with an underscore separating each word of the description (cf. Di Cristofaro 2023:283-285).
Or, back to our example

> It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love 💗👯

```
And on Friday I'm in love
<w pos="FU" wclass="X" hw="growing_heart" usas="Z99">
  {growing_heart}
</w>
<w pos="FU" wclass="X" hw="women_with_bunny_ears" usas="Z99">
  {women_with_bunny_ears}
</w>
```

Linguistic peryton #3: missing spaces

Missing spaces, old friends

Since the majority of tokenizers use the presence of (white)spaces as delimiter between to elements (in English, at least), when two or more words are not separated by (white)spaces the tagger may (and oftentimes it will) treat them as one single token.
That's why we may want to apply **word segmentation** (e.g through the Python module `python-wordsegment` cf. Di Cristofaro 2023:285-288) to our texts.

> When did I become such a girl… #overanalyzingeverything #thisisntlikeme

```
When did I become such a girl...
<exhashtag orig="overanalyzingeverything">
  overanalyzing everything
</exhashtag>
<exhashtag orig="thisisntlikeme">
  this isn't like me
</exhashtag>
```

But emojis are not words, so if something like

> It's 𝔽𝕣𝕚𝕕𝕒𝕪! And on Friday I'm in love💗

appears, then we must find another way (hint: check the *Extra* section of these slides).

More on Unicode and encodings

Unicode and encodings: advanced readings

The Unicode cookbook for linguists: Managing writing systems using orthography profiles by Steven Moran and Michael Cysouw (Available in Creative Commons)

The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!) by Joel Spolsky

What every programmer absolutely, positively needs to know about encodings and character sets to work with text by David C. Zentgraf

`emoji` Python module online documentation

Unicode® Standard Annex #15 | Unicode Normalization Forms

Any other *linguistic peryton*?

I fear so...

Work in progress

Here be dragons (and perytons)

Linguistic perytons in corpora A, B, C

folder	variationSelectors	emojis	tokens	nkfc	nkfcNormalised
calzedonia	0	458	8,512	19	19
carolinaherrera	0	985	50,113	78	79
champion	0	51	1,603	4	4
chloe	0	7	20,058	11	11
columbia1938	0	72	5,582	188	188
dkny	0	53	5,064	2	2
forever21	0	2,078	22,975	778	800
hm	0	212	15,586	5	5
miumiu	0	0	21,085	9	9
monsoon	0	70	16,673	32	33
patagonia	0	0	716	4	4
ugg	0	753	18,400	63	65
wrangler	0	293	7,397	42	42

The statistical significance of *linguistic perytons*

Starting from a dataset of approx. 200,000 tokens collected from Instagram, 6 different corpora have been built, each one representing a different *pre-processing procedure*.

	original symbols	**processed symbols**
original emojis	`oe_os`	`oe_ps`
original emojis retokenised	`oer_os`	`oer_ps`
processed emojis	`pe_os`	`pe_ps`

For each corpus, a **wordlist** (with no cut-off threshold) and **collocations** (stat=LogDice, LL=infinite, min.freq. = 1, min.range = 1) **for each word** of the corpus are calculated using 3 (possibly 4) corpus tools:

- AntConc (v4.2.4)
- LancsBox (v6.0.0)
- LancsboxX (v3.0.0)
- SketchEngine

Mixed models are employed to evaluate if and how significant is each procedure - and therefore, the impact that each type of peryton has on the collocates' rank and effect, and on the words' rank and frequency.

Know your data *AND* your tool(s)!

Along with an evaluation of the statistical significance of perytons, a list of each corpus tools' characteristic is being compiled to offer researchers a way to better understand what may happen when creating/analysis a corpus. Some preliminary - and random - notes:

- **AntConc**:
  - Accepts any encoding as long as it is defined by the user, and does not operate any conversion
  - Does not recognise emojis
  - Does not "recognise" symbols
- **LancsBox**:
  - Converts any input encoding into UTF-8
  - Limited support for emojis
  - Does not "recognise" symbols
- **LancsBoxX**:
  - Since it relies on XML, it is certain that the input data is UTF-8
  - Uses PyMUSAS (based on Spacy) to annotate the data, hence emojis and symbols become *linguistic perytons* if not pre-processed
- **SketchEngine**:
  - Converts any input encoding into UTF-8
  - Has partial support for emojis (e.g. **ZWJ** ones are not supported)
  - Operates NFKC on symbols (good but...)

A proposal: not a standard, but some guidelines

Some guidelines

1. Detail the Operating System, its `locale` and its encoding (e.g. `en_GB.UTF-8` for Unix-like OSes and Windows >= 10, `2057 UTF-8` for Windows < 10)
2. Detail the software (libraries, modules, etc…) employed to collect and pre-process the data, together with their versions
3. If possible, pre-tokenize the corpus contents (using e.g. Spacy or PyMUSAS) and document the procedures employed (including what mentioned in points 1-2)
4. Detail the corpus tool employed, along with the settings chosen (if the default ones are used, this should be made explicit)
5. ⚗ Provide a corpus-fingerprint through e.g. the profile function of `segments`

`segments` profile output

Grapheme	frequency	mapping
	22	
n	9	n
i	7	i
d	7	d
I	6	I
'	6	'
o	6	o
F	4	F
r	4	r
a	4	a
y	4	y
t	3	t
s	3	s
!	3	!

Grapheme	frequency	mapping
A	3	A
m	3	m
l	3	l
v	3	v
e	3	e
	2	
𝔽	2	𝔽
𝕣	2	𝕣
𝕚	2	𝕚
𝕕	2	𝕕
𝕒	2	𝕒
𝕪	2	𝕪
💗	1	💗
👯	1	👯

Extra: destroy all ~~monsters~~ perytons!

Python to the rescue

The *linguistic perytons* discussed so far can be pre-processed - thus producing a *linguistic perytons*-free corpus - through the use of some Python code.
The slides that follow present a series of checks (executed through a series of `if`, `elif`, and `else`) that are run on each token - as identified by PyMUSAS/Spacy.

Block #1: "symbols"

First, we check if a token contains "symbols"; if so, we normalize it through NFKC, including the non-normalised version as attribute to the tag that encloses the normalised one

```
if not unicodedata.is_normalized("NFKC", tok.text):
    normalised = unicodedata.normalize("NFKC", tok.text_with_ws)
    tag_normalised = spacy_nlp(normalised)
    for norm_token in tag_normalised:
        create_tag(norm_token, tag, e, n, spacy_nlp)
```

Block #2: variation selectors

Beside all the emojis characteristics discussed so far, there is (at least) another one that may produce *linguistic perytons*: variation selectors since the corpus tool (or the tagger) may count it as one token.
These are unicode characters in the range `\uFE00 - \uFE0F` - such as `\uFE0E` and `\uFE0F` checked in the code below - that are used to e.g. distinguish between the black and white version ❤ of the heart emoji and its coloured one ❤️:

> ❤ = `\u2764`
> ❤️ = `\u2764\uFE0F`

```
# now check for the presence of variation selectors \uFE0E and \uFE0F
elif len(str(tok.text)) < 2 and (ord(tok.text) == 65038 or ord(tok.text) == 65039):
    continue
```

Block #3: missing spaces

Now we check if there are occurrences where a word and one emoji (or multiple ones) are not separated by a white space; the code below is applied to the text after being retokenized as part of a different procedure - not documented here! - through the following method:

```
retokenize = spacy.load("en_core_web_sm", enable=["tokenizer"])
```

Then each token of the retokenized text is processed as follows:

```
elif retok._.has_emoji:
        fixed_text = ""
        emoji_indexes = {}
        indexes_skip = []
        # get a list of all the emojis in the text
        all_em = emoji.emoji_list(post_text)
        # add all emojis dictionaries (match_start, match_end, emoji) to the dictionary emoji_in
        for n_em, one_em in enumerate(all_em, start=1):
            emoji_indexes[n_em] = one_em
            # here I need to check if the match_start has already been included in the indexes to
            # may happen if two emojis are not separated by a space (or a character recognised as
            if one_em["match_start"] in indexes_skip:
                indexes_skip.remove(one_em["match_start"])
            # now I add the indexes to the list of ones to skip, adding +1 to the start and end s
            # and after making sure that contiguous emojis are not skipped using the above indexe
            indexes_skip.extend(
                [
                    i
                    for i in list(
                        range(
                            one_em["match_start"] + 1,
                            one_em["match_end"] + 1,
                        )
                    )
                ]
            )
```

Block #4: emojis

Now we check if the token is an emoji, and if so we transliterate it through the `emoji` Python module, while assigning some dummy values for the POS and USAS tags.

```python
elif emoji.purely_emoji(tok.text):
    new_tag = etree.SubElement(tag, "w")
    new_tag.attrib["pos"] = "FU"
    new_tag.attrib["wclass"] = "X"
    new_tag.attrib["hw"] = f'{demojize(tok.text).replace("{","").replace("}","")}'
    new_tag.attrib["usas"] = "Z99"
    new_tag.text = f"{demojize(tok.text_with_ws)}"
```

Block #5: everything else

At last, if a token is neither a symbol, nor a variation selector, nor an emoji, we treat is a "simple" word and tag it accordingly.

```python
else:
    new_tag = etree.SubElement(tag, "w")
    new_tag.attrib["pos"] = f"{tok.pos_}"
    new_tag.attrib["wclass"] = f"{tok.tag_}"
    new_tag.attrib["hw"] = f"{tok.lemma_}"
    # Since a tag may be composed of multiple USAS tags, we only get the first one
    new_tag.attrib["usas"] = f"{tok._.pymusas_tags[0]}"
    new_tag.text = f"{tok.text_with_ws}"
```

Thank you!

mdicristofaro@unimore.it
https://infogrep.it
https://catlism.github.io

3rd - 7th June 2024 in Modena, there is Summer School in Digital Humanities for PhD students:
https://www.summerschooldigitalhumanities.unimore.it/

References

Baron, Alistair, and Paul Rayson. ‘VARD 2: A Tool for Dealing with Spelling Variation in Historical Corpora’. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*, 15. Aston University, Birmingham, 2008.

Burke-Spolaor, S., Matthew Bailes, Ronald Ekers, Jean-Pierre Macquart, and Fronefield Crawford III. ‘Radio Bursts with Extragalactic Spectral Characteristics Show Terrestrial Origins’. *The Astrophysical Journal* 727, no. 1 (December 2010): 18.

Di Cristofaro, Matteo. *Corpus Approaches to Language in Social Media*. Routledge Advances in Corpus Linguistics. New York: Routledge, 2023.

Moran, Steven, and Michael Cysouw. The Unicode Cookbook For Linguists: Managing Writing Systems Using Orthography Profiles. Zenodo, 2018.

References /2

Petroff, E., E. F. Keane, E. D. Barr, J. E. Reynolds, J. Sarkissian, P. G. Edwards, J. Stevens, et al. ‘Identifying the Source of Perytons at the Parkes Radio Telescope’. *Monthly Notices of the Royal Astronomical Society* 451, no. 4 (21 August 2015): 3933–40.

Spolsky, Joel. ‘The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)’. Joel on Software (blog), 8 October 2003.

Winter, Bodo. *Statistics for Linguists: An Introduction Using R*. 1st ed. New York, NY: Routledge, 2019.

Zentgraf, David C. ‘What Every Programmer Absolutely, Positively Needs to Know About Encodings and Character Sets to Work With Text’. Kunststube, 27 April 2015.