grewmatch

Queries on UD and PARSEME annotated corpora

Bruno Guillaume — LORIA / Inria Nancy Grand-Est

Join work with Guy Perrier, Guillaume Bonfante
and Grew-match users

UniDive Webminar • 2023 June, 19

Inria
INVENTEURS DU MONDE NUMÉRIQUE

Loria
Laboratoire lorrain de recherche
en informatique et ses applications

Today's presentation

- Show **practical examples** of **Grew-match** usage
- Not much details about the language syntax
  (tutorials and documentation are available)
- Examples on **UD** (**Universal Dependencies**), then on **PARSEME + UD**
  - **Exploration** of how a treebank is annotated
  - **Linguistic observations**
  - **Error mining**: find inconsistencies and potential errors

http://match.grew.fr

Request

Result

grewmatch | Tutorial | UD 2.12 | SUD 2.12 | UD Latest | SUD Latest | UD Auto | SUD Auto

Show corpora list | UD_English-LinES@2.12

```
pattern { N [lemma="build"] }
```

Clustering 1: No Key Whether

lemma upos xpos features textform/wordform sentences order: initial context

Search | Count

Basic | n-grams | Clustering | Misc

Request on graphs:

- Search for a `form`
- Search for a `lemma`
- Search for a `upos`
- Search for a dependency relation
- Search for both relations and tags
- Filter with NAP (Negative Application Patterns)

Request on metadata:

- Search for a `sent_id`
- Search in full text (with regexp)

13 occurrences [0.324s]

Save | TSV | CoNLL

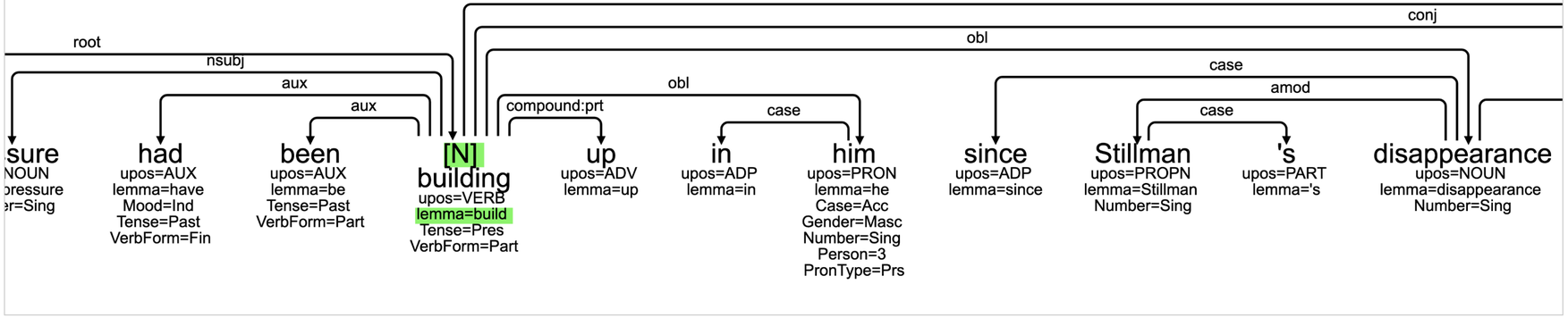
More results

1 / 10

- en_lines-ud-dev-doc2-3399
- en_lines-ud-dev-doc4-3682
- en_lines-ud-dev-doc8-4133
- en_lines-ud-test-doc3-4590
- en_lines-ud-train-doc1-79
- en_lines-ud-train-doc2-383
- en_lines-ud-train-doc3-1043
- en_lines-ud-train-doc3-1062
- en_lines-ud-train-doc3-1063
- en_lines-ud-train-doc4-1251

CoNLL | SVG

The pressure had been building up in him since Stillman's disappearance that morning, and it came out of him now as a torrent of words.



grewmatch + UD

- All examples are run on **UD_English-LinES** (version **2.12**)
  - 5,243 sentences and 94,217 tokens
  - *The majority of segments are from literature but there is also a section with online manual data and one section with Europarl data.*
  - No enhanced dependencies (easier to request and to read)

Explore how a treebank is annotated

How ***to*** is annotated?

Compute occurrences of the `form` ***to***

```
pattern { N [form="to"] }
```

to can have upos `ADP`

Can ***to*** be something else (not an `ADP`)?

```
pattern {
  N [form="to", upos<>ADP ]
}
```

to can also have upos `PART`

What else?

```
pattern {
  N [form="to", upos<>ADP|PART ]
}
```

Only 3 exceptions

Get all answers in one request

```
pattern { N [form="to"] }
```

`N.upos`

1284 PART | 904 ADP | 3 ADV

Explore how a treebank is annotated

How amod is used?

See some occurrences of the amod

```
pattern { N -[amod]-> M }
```

from an NOUN to a ADJ
(4,205 occurrences)

What else?

```
pattern { N -[amod]-> M }
without { N[upos = NOUN]; M[upos = ADJ] }
```

284 occurrences

Explore how a treebank is annotated

How an ADJ can be used?

What is the deprel used with an ADJ

```
pattern { M -> N; N [upos=ADJ] }
```

can be amod, xcomp…

See all possible values

```
pattern { e: M -> N; N [upos=ADJ] }
```

```
e.label
```

26 cases but
the first 4 gives 90%

What is the UPOS of the governor?

```
pattern { e: M -> N; N [upos=ADJ] }
```

```
M.upos
```

Mostly NOUN & VERB

See the correlation

```
pattern { e: M -> N; N [upos=ADJ] }
```

```
e.label
```

```
M.upos
```

e.label \ M.upos	734 NOUN	115 VERB	58 __undefined__	53 ADJ
765 amod	722	7		5
58 xcomp		58		
58 root			58	
53 conj	1	8		44

Explore how a treebank is annotated

How an ADJ can be used?

See the correlation `pattern { e: M -> N; N [upos=ADJ] }` `e.label` `M.upos`

e.label \ M.upos	734 NOUN	115 VERB	58 __undefined__	53 ADJ
765 amod	722	7		5
58 xcomp		58		
58 root			58	
53 conj	1	8		44

Explore how a treebank is annotated

How ***in front of*** is annotated?

Search the trigram of `lemma`s

```
pattern {
 N1 [lemma="in"];
 N2 [lemma="front"]; N1 < N2;
 N3 [lemma="of"]; N2 < N3
}
```

in front of is a `fixed` expression

Is it always annotated `fixed`?

```
pattern {
 N1 [lemma="in"];
 N2 [lemma="front"]; N1 < N2;
 N3 [lemma="of"]; N2 < N3
}
without { N1 -[fixed]-> N2; N1 -[fixed]-> N3 }
```

3 inconsistent annotations

Make linguistic observations

SVO?

VS?

```
pattern { V -[nsubj]-> S; V << S }
```

433 occurrences

SV or **VS**?

```
pattern { V -[nsubj]-> S }
```

```
S << V
```

SV: 94.6% and **VS** 6.4%

VO or **OV**?

```
pattern { V -[obj]-> O }
```

```
V << O
```

VO: 94.8% and **OV** 6.2%

SVO or what?

```
pattern { V -[nsubj]-> S; V -[obj]-> O }
```

```
S#V#O
```

SVO: 92.1%,
interesting exceptions

Error mining

the cc deprel

observe occurrences of cc

```
pattern { H -[cc]-> C }
```

(1) have a CCONJ as dependent
(2) are right-headed

Exceptions to (1)?

```
pattern { H -[cc]-> C; C [upos <> CCONJ] }
```

✅

Exceptions to (2)?

```
pattern { H -[cc]-> C; H << C }
```

❌ 8 annotations to be checked

Error mining

number agreement with subj

observe occurrences of subj
without number agreement

```
pattern {
  V -[nsubj]-> S;
  V.Number <> S.Number
}
```

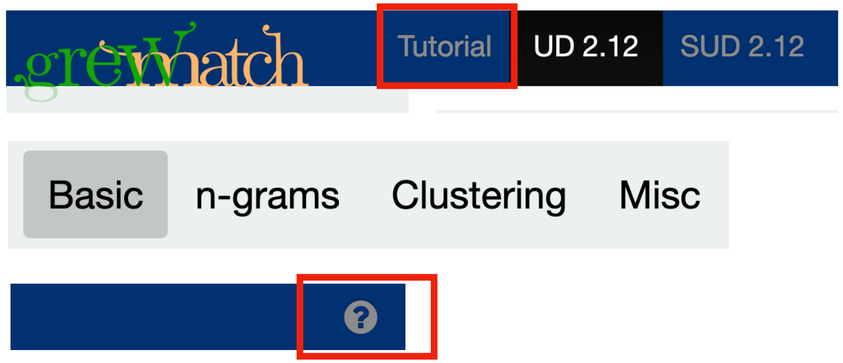
Explore and error mining: relation tables

- On each treebank, a set of **relations tables** (one per relation) is available
- equivalent to a double clustering of `upos` of the governor / `upos` of the dependent

Use:  and chose `amod` relation

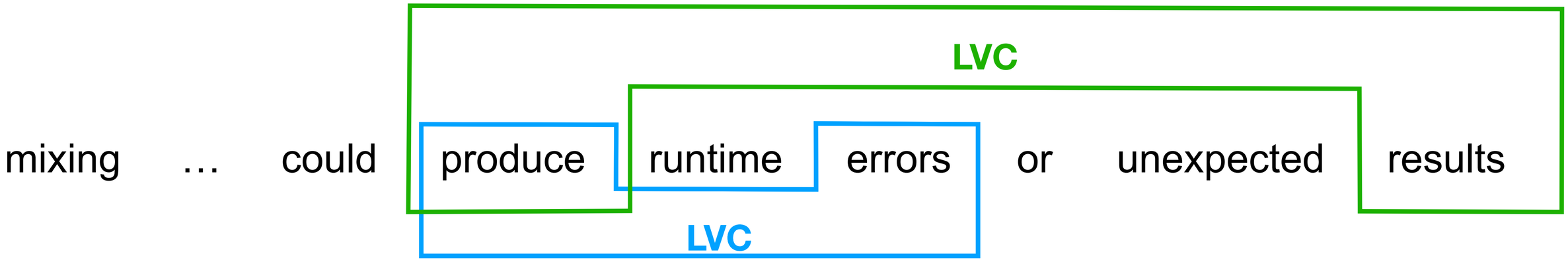
grewmatch + UD

- **Grew-match** is available on the **245 corpora** of UD 2.12 (and SUD)
- Automatically updated on some corpora (dev branch), available on request
- How to find help with the request language?
  - A **tutorial** available (top navbar, before UD)
  - **Snippets** on the right of the textarea
  - https://grew.fr/grew_match/help/



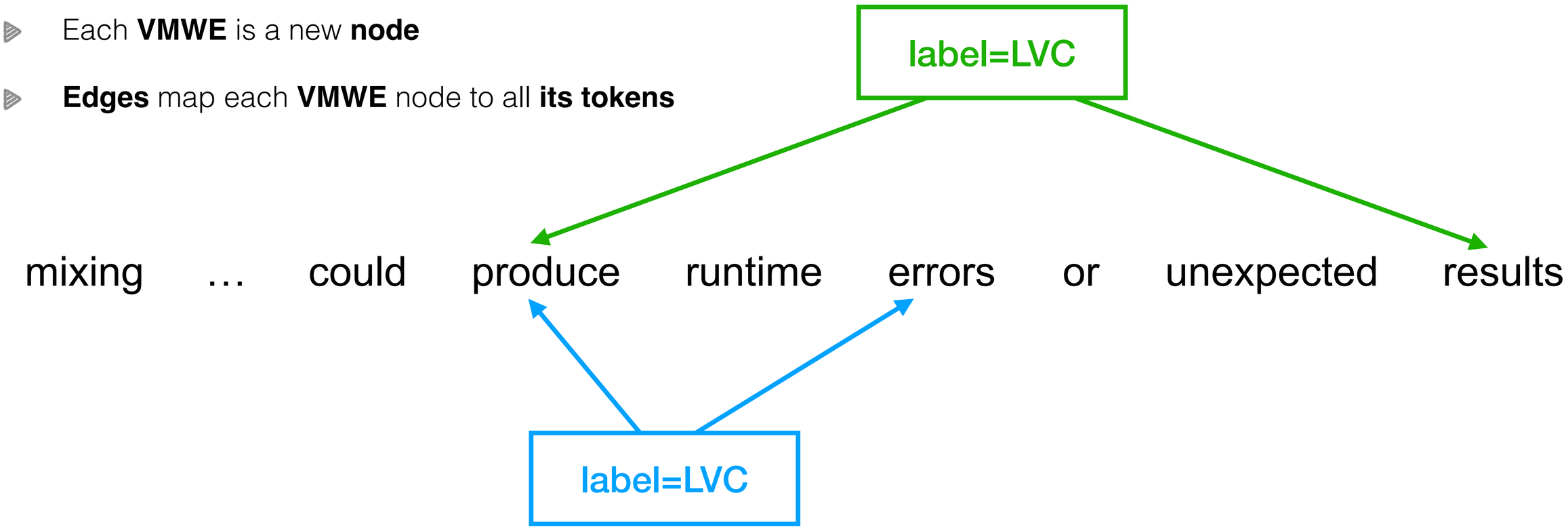
Q&A

grewmatch + PARSEME

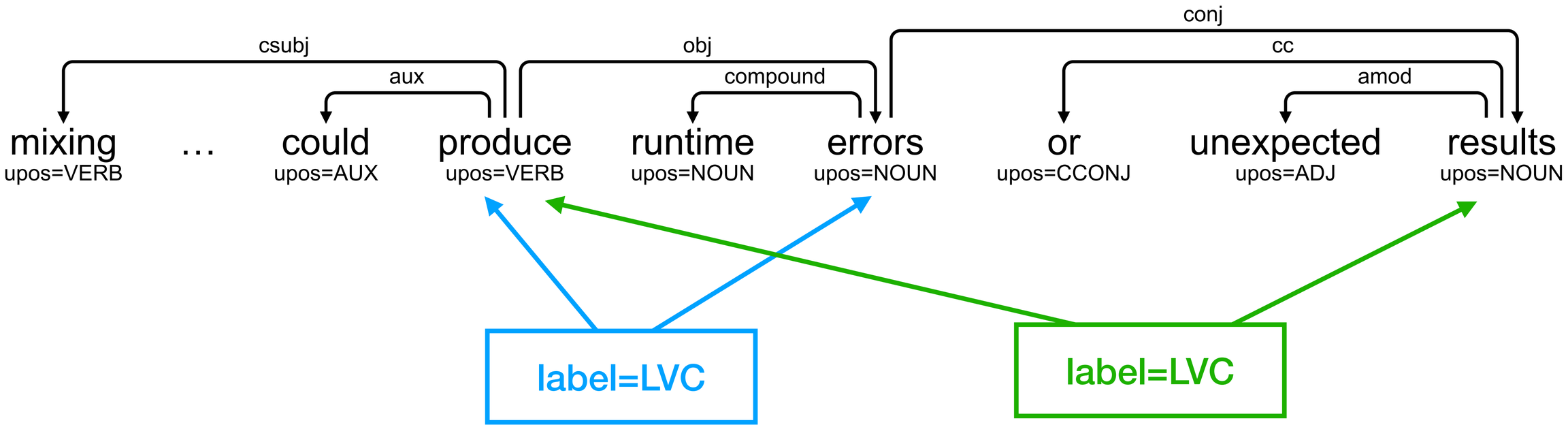


grewmatch + PARSEME

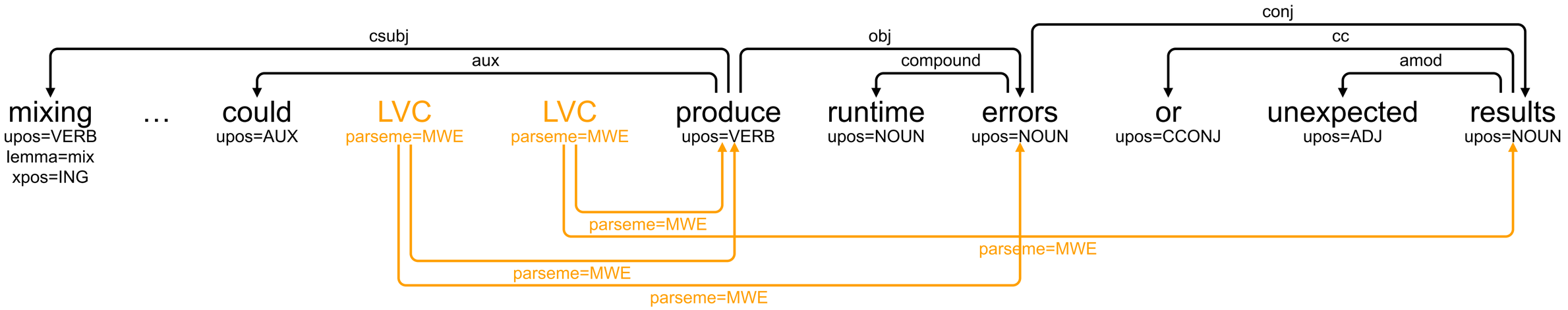
- Each **VMWE** is a new **node**
- **Edges** map each **VMWE** node to all **its tokens**



grewmatch + UD + PARSEME



grewmatch + UD + PARSEME



English Parseme data

Numbers of sentences

Treebank	Parseme	UD	Enhanced
PUD	201	1000	YES
LinES	3015	5243	NO
EWT	4221	16622	YES
TOTAL	7437	22865	

Explore how a treebank is annotated

How VID is used?

Examples of VID usage?

```
pattern { MWE [label="VID"] }
```

lemmas of VERB in VID?

```
pattern { MWE [label="VID"]; MWE -> V; V[upos=VERB] }
```

```
V.lemma
```

Explore how a treebank is annotated

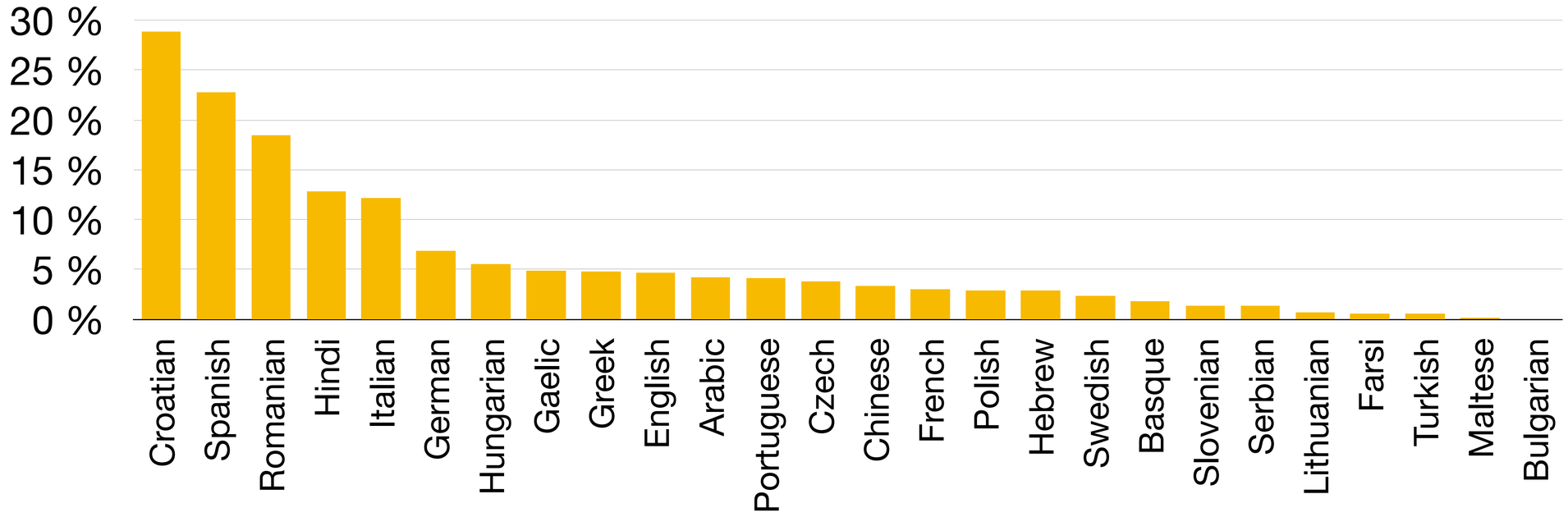
overlapping VMWE?

```
pattern { MWE1 [label] }
MWE2 [label]; MWE1 -> X; MWE2 -> X
```

is it frequent?

4.8% of VMWE overlap

in other languages?



Make linguistic observations

MVC usage in English?

Examples of MVC usage?

```
pattern { MWE [label="MVC"] }
```

Sizes of MVC?

```
pattern { MWE [label="MVC"] }
```

```
MWE.__out__
```

lemmas in MVC?

```
pattern { MWE [label="MVC"]; MWE -> N1; MWE -> N2; N1 << N2 }
```

```
N1.lemma
```

```
N2.lemma
```

Make linguistic observations

is *make* + `obj` a VMWE?

yes or no?

```
pattern { N1 [lemma="make"]; N1 -[obj]-> N2 }
```

```
MWE [label]; MWE -> N1; MWE -> N2
```

60% No, 40% Yes

If yes, what VMWE `label`?

```
pattern {
  N1 [lemma="make"]; N1 -[obj]-> N2;
  MWE [label]; MWE -> N1; MWE -> N2
}
```

`MWE.label`

64 LVC.full | 8 VID | 3 LVC.cause

If yes, `lemma` of `obj`

```
pattern {
  N1 [lemma="make"]; N1 -[obj]-> N2;
  MWE [label]; MWE -> N1; MWE -> N2
}
```

`N2.lemma`

42 clusters

Error mining

An VMWE must contain a VERB

Exceptions?

```
pattern {MWE [label]}
without {MWE -> V; V[upos=VERB]}
```

14 occurrences

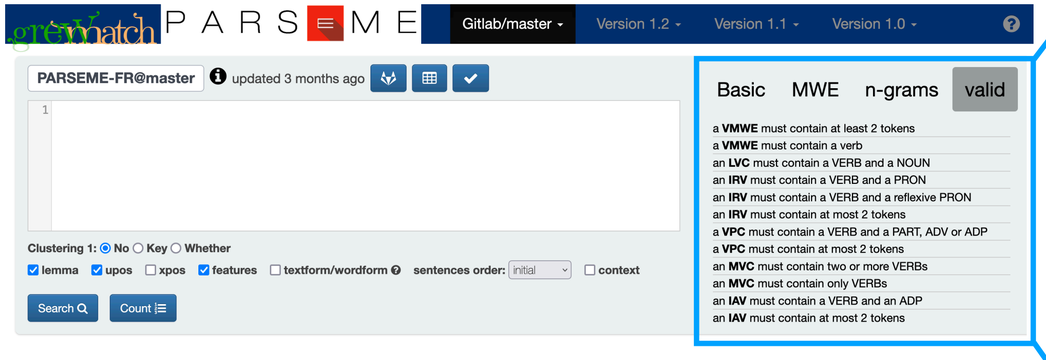
Taking AUX into account?

```
pattern {MWE [label]}
without {MWE -> V; V[upos=VERB|AUX]}
```

11 occurrences

Error mining: consistency with UD

Many other examples available in the online interface



Basic MWE n-grams **valid**

- a **VMWE** must contain at least 2 tokens
- a **VMWE** must contain a verb
- an **LVC** must contain a VERB and a NOUN
- an **IRV** must contain a VERB and a PRON
- an **IRV** must contain a VERB and a reflexive PRON
- an **IRV** must contain at most 2 tokens
- a **VPC** must contain a VERB and a PART, ADV or ADP
- a **VPC** must contain at most 2 tokens
- an **MVC** must contain two or more VERBs
- an **MVC** must contain only VERBs
- an **IAV** must contain a VERB and an ADP
- an **IAV** must contain at most 2 tokens

http://parseme.grew.fr

Error mining: consistency with UD

	Request	one_token	no_verb ↓	LVC	IRV	IRV_reflex	IRV_3	VPC	VPC_3	MVC	MVC_not_verb	IAV	IAV_3
Treebank		14126	11268	6222	856	3322	1603	13078	0	1912	684	289	1662
PARSEME-HU@1.3	18060	5745	5901	760				5654					
PARSEME-AR@1.3	2252	17	1302	835						3	7	85	3
PARSEME-PL@1.3	1837		836	612	193	193	3						
PARSEME-CS@1.3	3432		790	585	272	272	1513						
PARSEME-ZH@1.3	12923	5382	526	262				4542		1869	342		
PARSEME-BG@1.3	852	11	416	223	86	86	9					3	18
PARSEME-TR@1.3	1015	6	330	679									
PARSEME-HE@1.3	701	42	264	341				54					
PARSEME-GA@1.3	410	3	214	117				26				41	9
PARSEME-HR@1.3	738		146	57	24	24	1					83	403
PARSEME-DE@1.3	2699	1268	126	3	1	3	53	1245					
PARSEME-SV@1.3	3479	1616	92	2		237		1532					
PARSEME-SR@1.3	174		91	56	13	13	1						
PARSEME-IT@1.3	1502	9	65	41	11	1144	11	6		2	16	9	188
PARSEME-MT@1.3	202	13	59	128		1		1					
PARSEME-EL@1.3	275	1	26	221		1	1	11			14		
PARSEME-PT@1.3	1343	1	26	43	249	1021	3						
PARSEME-ES@1.3	502	2	23	4	1	8	1	1		32	298	5	127
PARSEME-LT@1.3	19		12	7									
PARSEME-EN@1.3	68	4	11	11				6		4	4	10	18
PARSEME-RO@1.3	979		5	3		206	2					13	750
PARSEME-EU@1.3	355		4	351									
PARSEME-FR@1.3	121	5	2	3	1	107	3						
PARSEME-FA@1.3	861	1	1	857		1	1						
PARSEME-HI@1.3	25			20						2	3		
PARSEME-SL@1.3	198			1	5	5	1					40	146

https://parseme.grew.fr/tables/?data=parseme/valid@1.3

http://parseme.grew.fr

- **Grew-match** is available on the **26 corpora** (versions **1.2**, **1.3**)
- All **master** versions are updated automatically from **gitlab**

Q&A

Other Grew related tools?

grew

https://grew.fr

Graph Rewriting for NLP

grew

https://pypi.org/project/grewpy

Python library: work in progress, not (yet) well-documented

ARBORATOR grew

https://arborator.github.io/

Online Annotation tool

- access to Grew-match **request** & Grew **rewriting**
- active development

What grewmatch cannot do?

- Search with an **unbound** dependency length ❌ `pattern { M -*-> N; N [upos=ADJ] }`
- Search for **disjunction** of patterns
- Run the same request in **multiple** treebanks at once
  - use **Command line interface** or the **Python** lib `grewpy`
  - some static **tables** available: **https://tables.grew.fr**

Conclusion

grew

and related tools https://grew.fr

mailing-list grew@inria.fr. To subscribe: https://sympa.inria.fr/sympa/info/grew

For **questions**, **issues** or **feature requests**: https://github.com/grew-nlp/grew/issues

Many thanks to Guy!

Grew would not exist without him!