Graph-based multi-layer querying in Parseme Corpora

Bruno Guillaume

Graph-based multi-layer querying in Parseme Corpora

Bruno Guillaume
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
Bruno.Guillaume@inria.fr

Abstract

We present a graph-based tool which can be used to explore Verbal Multi-Word Expression (VMWE) annotated in the Parseme project. The tool can be used for linguistic exploration on the data, for helping the manual annotation process and to search for errors or inconsistencies in the annotations.

1 Introduction

The Parseme project (Monti et al., 2018) proposes a large set of annotated data with Verbal Multi-Word Expressions (VMWE). In version 1.2 (Ramisch et al., 2020), 14 languages were covered but with older versions and ongoing work[1], there are now data in 26 languages (See Table 4 in appendix for list of languages and the number of sentences for each language). In the last release, Parseme 1.3, only "verbal" Multi-Word Expressions are annotated; the annotation of other categories is planed for future releases.

Parseme data is published with associated morpho-syntactic annotations, in accordance with the Universal Dependencies (de Marneffe et al., 2021) framework. Some parseme annotations are directly built on data available in the UD project. In this case, we have both high-quality morpho-syntactic annotations and VMWE annotations on the same data. Other parts of the Parseme data, which are not built on existing UD data, are accompanied by an automatic morpho-syntactic annotation, obtained with UDPipe (Straka et al., 2016), thus also following the UD annotation framework. This means that all annotated data from the Parseme project can be considered as multi-layer annotated data, with morphosyntactic annotations encoded following UD and VMWE.

In this article, we propose an encoding of the two annotation layers in a common structure, using a graph encoding of both UD and VMWE annotations. With this encoding, it is possible to use graph-based tools to work with the data. In this work, we use our GREW tool (Guillaume, 2021) to make queries on the two layers.

The Parseme 1.3 data will be released on http://hdl.handle.net/11372/LRT-5124. At the time of the final version of the paper, these data are not available. The experiments reported in the paper are done on a preliminary version of the data provided by the Parseme team. We cannot exclude minor differences between the data we used in our observations and the official 1.3 data.

In Section 2, we explain the encoding. The next sections give examples of usage with general observations in Section 3, applications to error mining in Section 4 and some more comprehensive study of the consistency between UD and Parseme annotation layers in Section 5.

2 Graph encoding

The two annotation layers (UD and VMWE) are stored in a common technical format (CUPT)[2], but it is not straightforward to consider both in the same structure. In UD, each sentence is split in a sequence of tokens and each Parseme annotation consists in identifying a subset of the tokens of the sentence which correspond to a VMWE. In addition to the subset, a tag is given to each VMWE.

The Parseme guidelines[3] describes the set of tags and their definitions. The tagset contains three universal tags: LVC.FULL, LVC.CAUSE for light verb constructions and VID for verbal idioms. Three quasi-universal categories are also defined: VPC.FULL, VPC.SEMI for verb-particle constructions and MVC for multi-verb constructions. A few other tags are used in the corpora: the IAV for inherently adpositional verbs, presented

[1] https://gitlab.com/parseme/corpora

[2] http://multiword.sourceforge.net/cupt-format

[3] https://parsemefr.lis-lab.fr/parseme-st-guidelines/1.3/

Proceedings of the 19th Workshop on Multiword Expressions (MWE 2023), pages 58–64

as experimental in the Parseme guidelines and the tag LS.ICV for inherently clitic verbs (currently only used in Italian data). Some development versions of the data makes also use of the special tag NOTMWE which, as its name indicates, does not encode a VMWE, it is used in the consistency checking mechanism[4].

There are two difficulties in the encoding.

- A VMWE is not always a span of the original text, or in other words it does not always contains a subset of consecutive tokens of the sentence. For instance, in the sentence *Take a look !!!*[5], the subset containing *Take* and *look* is annotated with the tag LVC.FULL the token *a* between the two elements not being part of the VMWE.

- The second problem is that the same token can be included in more than one VMWEs. In the sentence *[...] to get rid of the moral burden [...]*, two subsets are annotated independently: the 3 elements *get*, *rid* and *of* are tagged as IAV (Inherently adpositional verbs) and the 2 elements *get* and *rid* are tagged as MVC (Multi-verb constructions). Such VMWE annotations will be called *overlapping* VMWEs.

In order to take into account theses difficulties, we propose to encode the two layers in a single graph structure. Our graphs contain two kinds of nodes and two kinds of edges:

- UD nodes and UD edges which encode the lexical tokens and the dependency relations between tokens

- Parseme nodes and Parseme edges which encode VMWEs: each VMWE is represented by a new node with a feature named `label` which stores the tag. The node associated with a VMWE is linked with Parseme edges to all the UD token it contains.

Figure 1 shows a simplified picture of the encoding of the two overlapping VMWEs in the sentence *[...] to get rid of the moral burden [...]*. UD nodes and relations are drawn in black whereas Parseme nodes and edges are drawn in blue and below the sentence.

[4]Script `consistencyCheckWebpage.py` available in https://gitlab.com/parseme/utilities

[5]English examples come from the English Parseme corpus: https://gitlab.com/parseme/parseme_corpus_en

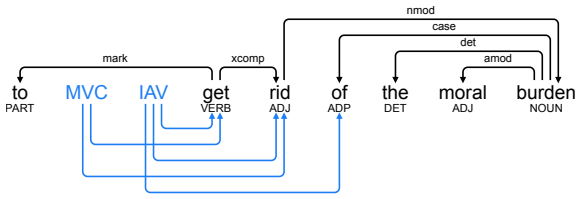

Figure 1: Graph representation (simplified) of two overlapping VMWE annotations

Note that Parseme nodes are not really inserted in the linear structure of the sentence. By convention, these nodes are drawn before the first token of the subset, just to ease the reading of the figures.

3 Multi-layer queries

The benefit of having the two annotation layers in the same structure is that it is possible to make queries which refer to both layers and then to make cross observations. We use the GREW tool which allows to write graph queries that can be executed on the Parseme corpora represented has above. The tool is available in an online web interface: GREWMATCH[6] on a predefined set of treebanks. A Python library, named GREWPY, is also available to use queries in scripts.

We give a few examples of GREW queries on the Parseme graph encoding.

3.1 VWMEs by types

Using the fact that all Parseme nodes have a feature named `label` (and that UD nodes do not have such a feature), the simple request below returns the set of all annotated VMWEs.

```
1 pattern { MWE [label] }
```

In the request, `MWE` is an node identifier. The query can be rephrased as: "search for any node having a feature named `label` and call this node `MWE`".

GREW proposes a mechanism to cluster the output of a query following some criterion. With the clustering key `MWE.label`, the set of solutions of the previous query is clustered in a partition of subsets according to the value of the feature `label` of the node `MWE`.

In Table 1 in appendix, each line correspond to the size of the clusters obtained for each language in the Parseme data.

[6]http://parseme.grew.fr

3.2 VWMEs by sizes

We keep the same basic request used in the previous subsection. In GREW, the clustering key `MWE.__out__` splits the occurrences returned by a request depending on the number of outgoing edges on the node `MWE`. Following the encoding described in Section 2, this corresponds to the number of tokens implied in the VMWE. Table 2 in appendix reports the sizes of the clusters obtained with this clustering for all languages.

According to the notion of Multi-Word Expression, we do not expect to have one-token annotation as VMWE. All languages have a few number of such VMWEs (above 50 occurrences) except for four languages: Hungarian, Chinese, Swedish and German.

In Hungarian data, there are 5745 one-token VMWEs; this is probably linked to the fact that Hungarian is an agglutinative language; among the cases, the same noun with lemma *bekezdés* '*paragraph*' appears 995 times, it is tagged as VPC.FULL and it is built from the verb *bekezd* '*to indent*' and with a noun-forming suffix *-és*[7].

There are 5382 cases in Chinese, but Chinese, not using whitespaces, is well know to be a language in which tokenization is challenging.

In Swedish, there are 1614 occurrences and 1268 occurrences in German. In both languages, the major part of cases are particle verbs. In German, the four most frequent lemmas are: *einsetzen* '*to insert*', *anbieten* '*to offer*', *ankündigen* '*to announce*', *mitteilen* '*to share*'. Unlike English where particles of particle verbs always remain separate words (*put off*, *to put off*), German particles of infinitival forms are fronted and spelled as one word, joining the main verb (*abschrecken* '*to put off*', *abgeschreckt* '*be put off*'). whereas particles of finite verb forms are positioned behind the main verb and spelled as separate words (*schreckt* ab 'put off') just as in English. In Swedish, just as in German, there are many particle verbs that alternate between realizing the particle as a separate word and as a prefix. This shows that the notion of tokenisation is considered quite differently in both projects.

Apart from one-token annotations, the size 2 is the most common setting in all languages. Size 6 and higher are quite rare and the maximum is reached by Hebrew with one VMWE containing 13 tokens.

[7] https://en.wiktionary.org/wiki/bekezdés

3.3 Ratio of overlapping VMWEs

With a graph request, we can distinguish VMWE annotations with or without overlapping. The request below corresponds to the "without overlapping" case:

```
1 pattern {
2   MWE1 [label]
3 }
4 without {
5   MWE2 [label];
6   MWE1 -> X; MWE2 -> X
7 }
```

Lines 1-3 is a request for any VWME. Lines 4-7 use the `without` construction of GREW which filters the output of a query: each occurrence of the basic query (line 1-3) which satisfies the constraint expressed in the `without` part is filtered out. In our example, cases where some `MWE2` exists, which shares a token `X` with the one previously found `MWE1` are removed. The result of the full query is then only the non-overlapping VMWEs.

For the "with overlapping" case, the request is the same where the keyword `without` is replaced by the keyword `with` (line 4). Table 3 shows the ratio of overlapping VMWEs for each language.

4 Error mining

One of the common usage of GREW and mainly GREW-MATCH is error mining. By looking at all examples of a given query, we can spot inconsistencies and potential annotation errors. A first example of error mining is to explore the occurrences of one-token VMWEs (see Subsection 3.2) which are unexpected and require manual inspection. Let us see a few other examples.

4.1 VMWEs without any verb

We can test whether each annotation does contain a verb. This is expected as the current version of Parseme focuses on "Verbal" Multi-Word Expressions. The following request searches for Parseme VMWEs without any verb, according to UD annotation (UD uses the two POS tags AUX and VERB for the verbal forms).

```
1 pattern {
2   MWE [label];
3 }
4 without {
5   MWE -> V;
6   V[upos=VERB|AUX]
7 }
```

Table 4 in appendix gives the numbers of occurrences in each language for this request (column **no_verb**). The median of the number of occurrences in the 26 treebanks is 91.5, with two treebanks above 1000 occurrences. The two exceptions are Hungarian (we have already seen that many nouns are tagged as VMWE because there are built from a verb and a noun-forming affix) and Arabic (where we also observe many cases of noun describing an action, build on a verbal root). This shows that the definition of what is a "verb" in Parseme is not fully aligned with the UD policy.

4.2 Inherently reflexive verbs

Parseme consider the tag IRV for Inherently reflexive verbs. In the meantime in UD, there is the feature `Reflex=Yes` which can be used on reflexive pronouns (but this is not mandatory). We can expect that a VMWE annotated as IRV contains such a reflexive pronoun. The request above allows to search for the exceptions to this rule.

```
pattern {
  MWE [label = "IRV"];
}
without {
  MWE -> P;
  P[upos=PRON, Reflex=Yes]
}
```

Table 4 in appendix gives the numbers of occurrences in each language for this request (column **IRV_no_reflex**). For the three highest numbers are 1144 in Italian, 1021 in Portuguese and 237 in Swedish. This is due to the fact that these three languages does not annotate feature `Reflex` in the UD data. Romanian and French have both annotations IRV in Parseme and `Reflex=Yes` on pronoun in UD, but there are many inconsistencies.

Is is worth noting that Slovenian also appeared in the problematic languages at the submission time. but it was due to a bug in the data which was found thanks to the current work and which was fixed in the mean time.

5 Consistency UD/Parseme

In this section, we give a few examples of requests which can be used in GREW-MATCH to explore how some specific class of VWME is annotated in one treebank.

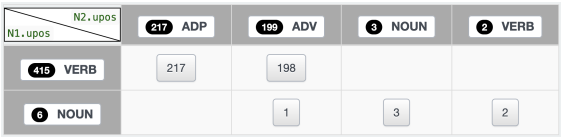

Figure 2: POS of the tokens used in the VPC construction in English

5.1 Verb-particle constructions in English

The example runs on Verb-particle constructions (VPC) and on English data. According to Parseme guidelines, two subtags must be used: VPC.FULL for fully non-compositional VPC and VPC.SEMI for semi-non-compositional.

First, we can have a look at the distribution of this kind of VMWE according to the number of tokens implied[8]. We observed 421 occurrences (368 VPC.FULL and 53 VPC.SEMI)[9] of this label, all of them having exactly two tokens.

Another request[10], specifying the two tokens `N1` and `N2`, can display the distribution of the POS of the tokens in the Figure 2 which shows that two constructions `VERB-ADP` and `VERB-ADV` covers all but 6 cases.

Exploring further[11], we observed in the 217 `VERB-ADP` cases, a large majority (202) of annotation where the `VERB` is linked to the `ADP` with relation `compound:prt`. Other cases are: no direct relation between the two nodes (10 cases), relation `advmod` (3 cases), `compound` (2 cases). Similarly[12], we observed in the 198 `VERB-ADV` cases, a majority (118) of annotation where the `VERB` is linked to the `ADP` with relation `compound:prt`. Other cases are: relation `advmod` (75 cases), no direct relation between the two nodes (2 cases), `compound`, `obl` and `xcomp` (1 case fo each).

These irregularities in the annotation would require a careful inspection by a native English speaker but we can already see a bunch of annotation inconsistencies either in the UD annotation layer or in the Parseme one.

[8] http://parseme.grew.fr/?custom=63f1ee845234a

[9] The numbers of this section are based on requests done on 2023/02/19, they may changed when the data is updated. Requests on a stable data from a release will be provided for final version.

[10] http://parseme.grew.fr/?custom=63f1eeda64fe8

[11] http://parseme.grew.fr/?custom=63f1f03dea172

[12] http://parseme.grew.fr/?custom=63f1f0d94e4ec

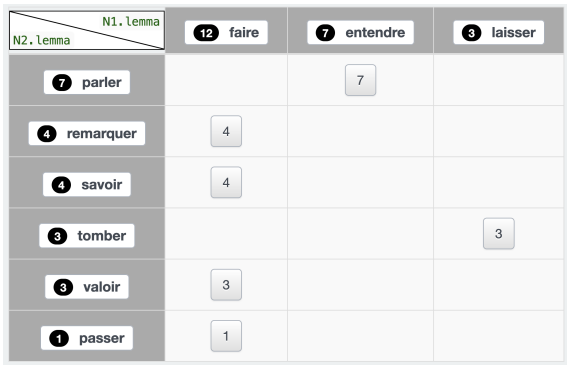

Figure 3: Lemmas used with MVC label in French data. Translations of columns lemmas are '*to do*', '*to hear*' and '*to let*'. Translations of rows lemmas are '*to speak*', '*to remark*', '*to leave*', '*to fall*', '*to be worth*' and '*to pass*'.

5.2 MVC in French

There are 22 occurrences of the MVC in the French data, all having two tokens. All are continuous except one containing a negation *on n'**entendra** plus **parler** de...* '*one will no more hear about...*'. The Figure 3 shows the distribution of lemmas of the two tokens `N1` and `N2` (following the linear order).

The syntactic annotation is regular with lemma *faire* '*to do*' for `N1` as a causative auxiliary of `N2` and for the two other lemmas (*entendre* '*to hear*' et *laisser* '*to let*'), an `xcomp` relation from `N1` to `N2`.

By searching the corresponding lemmas, we found a few annotation errors or annotation inconsistencies.

6 Conclusion

We have shown in this paper that using a graph encoding to represent a multi-layer annotation in a common structure is useful and can be exploited for different purposes, like error mining or linguistic exploration of the data. This methodology opens new perspectives for corpora maintenance and is complementary to existing tools like the UD validation script[13] and the Parseme consistency checking. Using the same idea, it would be possible to encode other annotation layers, like the ones available in a corpus like the GUM corpus[14] (Zeldes, 2017).

[13] https://universaldependencies.org/validation-rules.html

[14] https://gucorpling.org/gum/

Acknowledgments

We would like to thank the reviewers for their helpful comments. Thanks also to Kim, Joakim and Khensa for their help on German, Swedish and Arabic respectively.

References

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Bruno Guillaume. 2021. Graph Matching and Graph Rewriting: GREW tools for corpus exploration, maintenance and conversion. In *EACL 2021 - 16th conference of the European Chapter of the Association for Computational Linguistics*, Kiev/Online, Ukraine.

Johanna Monti, Savary Agata, Marie Candito, Verginica Barbu Mititelu, Bejček Eduard, Cap Fabienne, Čéplö Slavomir, Silvio Ricardo Cordeiro, Eryiğit Gülşen, Voula Giouli, Maarten van Gompel, HaCohen-Kerner Yaakov, Kovalevskaitė Jolanta, Krek Simon, Liebeskind Chaya, Carla Parra Escartín, Lonneke van der Plas, Qasemizadeh Behrang, Ramisch Carlos, Federico Sangati, Stoyanova Ivelina, and Vincze Veronika. 2018. Parseme multilingual corpus of verbal multiword expressions.

Carlos Ramisch, Agata Savary, Bruno Guillaume, Jakub Waszczuk, Marie Candito, Ashwini Vaidya, Verginica Barbu Mititelu, Archna Bhatia, Uxoa Iñurrieta, Voula Giouli, Tunga Güngör, Menghan Jiang, Timm Lichte, Chaya Liebeskind, Johanna Monti, Renata Ramisch, Sara Stymne, Abigail Walsh, and Hongzhi Xu. 2020. Edition 1.2 of the PARSEME shared task on semi-supervised identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pages 107–118, online. Association for Computational Linguistics.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

A Example Appendix

Language	IAV	IRV	LS.ICV	LVC.cause	LVC.full	MVC	VID	VPC.full	VPC.semi
Arabic	581	0	0	303	2678	5	1182	0	0
Basque	0	0	0	214	3152	0	880	0	0
Bulgarian	90	3223	0	222	1909	0	1260	0	0
Croatian	1388	1193	0	147	880	0	293	1	0
Chinese	0	0	0	177	1214	3826	973	0	4629
Czech	0	10000	0	0	2923	0	1613	0	0
English	71	0	0	51	333	51	187	368	53
Farsi	0	1	0	0	3435	0	17	0	0
French	0	1501	0	97	1878	22	2157	0	0
German	0	322	0	33	311	0	1437	1744	194
Greek	0	1	0	179	5293	51	2841	143	0
Hebrew	0	0	0	223	1049	0	1108	153	0
Hindi	0	0	0	26	641	306	61	0	0
Hungarian	0	0	0	401	1143	0	104	5156	956
Irish	187	0	0	118	200	0	106	28	20
Italian	497	1144	37	174	734	33	1484	105	2
Lithuanian	0	0	0	25	479	0	308	0	0
Maltese	0	1	0	1	700	2	518	4	0
Polish	0	3688	0	314	2478	0	833	0	0
Portuguese	0	1021	0	127	3954	18	1306	0	0
Romanian	3340	3826	0	182	516	0	1644	0	0
Serbian	0	564	0	69	402	0	269	0	0
Slovenian	710	1626	0	64	239	0	724	0	0
Spanish	511	714	0	81	392	713	327	1	0
Swedish	0	237	0	10	417	0	441	1461	589
Turkish	0	0	0	0	3583	5	4141	0	0

Table 1: Numbers of occurrences of VMWEs with their labels.

Language	1	2	3	4	5	6	7	8	9	10	11	12	13
Arabic	17	3673	946	91	11	10	0	1	0	0	0	0	0
Basque	0	4164	70	12	0	0	0	0	0	0	0	0	0
Bulgarian	11	5974	604	102	13	0	0	0	0	0	0	0	0
Croatian	0	3182	640	75	3	2	0	0	0	0	0	0	0
Chinese	5382	5224	136	35	15	14	6	5	1	0	1	0	0
Czech	0	11178	2571	664	97	18	8	0	0	0	0	0	0
English	4	1001	73	25	7	3	0	1	0	0	0	0	0
Farsi	1	3004	404	38	4	2	0	0	0	0	0	0	0
French	5	4353	1048	180	34	28	6	1	0	0	0	0	0
German	1268	1976	644	129	15	7	1	0	1	0	0	0	0
Greek	1	6253	1511	523	166	31	9	7	5	1	1	0	0
Hebrew	42	1781	584	87	21	5	8	2	2	0	0	0	1
Hindi	0	961	15	46	9	1	1	0	1	0	0	0	0
Hungarian	5745	2010	5	0	0	0	0	0	0	0	0	0	0
Irish	3	477	152	21	5	1	0	0	0	0	0	0	0
Italian	9	2693	1118	288	64	27	11	0	0	0	0	0	0
Lithuanian	0	683	99	21	7	1	0	1	0	0	0	0	0
Maltese	13	680	391	100	32	3	4	1	1	0	1	0	0
Polish	0	6550	653	88	13	6	0	2	0	0	0	1	0
Portuguese	1	5449	650	263	32	20	6	4	0	1	0	0	0
Romanian	0	8009	1368	74	45	12	0	0	0	0	0	0	0
Serbian	0	1151	128	17	4	3	1	0	0	0	0	0	0
Slovenian	0	2732	531	72	21	4	2	1	0	0	0	0	0
Spanish	2	2089	569	69	10	0	0	0	0	0	0	0	0
Swedish	1614	1336	188	14	3	0	0	0	0	0	0	0	0
Turkish	6	7233	445	41	4	0	0	0	0	0	0	0	0

Table 2: Numbers of tokens of VMWEs.

Language	Yes	No
Bulgarian	0.0%	100.0%
Maltese	0.16%	99.84%
Turkish	0.57%	99.43%
Farsi	0.58%	99.42%
Lithuanian	0.74%	99.26%
Serbian	1.38%	98.62%
Slovenian	1.4%	98.6%
Basque	1.77%	98.23%
Swedish	2.31%	97.69%
Hebrew	2.88%	97.12%
Polish	2.95%	97.05%
French	3.04%	96.96%
Chinese	3.38%	96.62%
Czech	3.78%	96.22%
Portuguese	4.17%	95.83%
Arabic	4.27%	95.73%
English	4.67%	95.33%
Greek	4.82%	95.18%
Irish	4.86%	95.14%
Hungarian	5.54%	94.46%
German	6.9%	93.1%
Italian	12.19%	87.81%
Hindi	12.86%	87.14%
Romanian	18.46%	81.54%
Spanish	22.82%	77.18%
Croatian	28.86%	71.14%

Table 3: Ratio of VMWEs which overlap with another annotation.

Language	# sentences	no_verb	irv_no_reflex
Arabic	7483	1302	0
Basque	11158	4	0
Bulgarian	21599	416	2
Croatian	6133	146	2
Chinese	48929	526	0
Czech	49431	790	0
English	7436	11	0
Farsi	3617	1	1
French	20961	2	107
German	8996	126	3
Greek	26175	26	1
Hebrew	19200	264	0
Hindi	1684	0	0
Hungarian	6159	5901	0
Irish	1705	214	0
Italian	15728	65	1144
Lithuanian	11104	12	0
Maltese	10600	59	1
Polish	23547	836	0
Portuguese	32062	26	1021
Romanian	56664	5	206
Serbian	3586	91	0
Slovenian	27825	0	5
Spanish	5515	23	8
Swedish	6026	92	237
Turkish	22306	330	0

Table 4: Numbers of occurrences of VMWEs without any verbal token (column `no_verb`) and of VMWEs tagged IRV without any reflexive pronoun (column `irv_no_reflex`).