Language Leaders Guide for ST 1.1 (2018)

PARSEME shared task on automatic identification of verbal multiword expressions 1.1

Table of Contents

What is the role of a Language Leader (LL)?

What are the tools available for language teams in the shared task?

How to use and adapt the annotation guidelines to your language?

How to organize your annotator team?

How to choose your corpus?

How to technically prepare a new corpus for annotation?

How to re-annotate a corpus that was already annotated with FLAT?

How to modify your corpus during annotation?

How to update statistics about your corpus?

How to download files?

How to modify the annotation team?

How to modify the file assignment?

How to perform consistency checks and prepare the final corpora?

How to prepare a double-annotated subcorpus for calculating the inter-annotator agreement?

How to adjudicate annotations?

1. What is the role of a Language Leader (LL)?

The Language Leader (LL) is the person who coordinates the annotation effort for a given language in the PARSEME shared task on identification of verbal multiword expressions.

The shared task languages are organized into Language Groups (LGs), related to language families (e.g. Romance languages, Slavic languages, etc). Each language belongs to a language group, and each language group has one or more Language Group Leaders (LGLs). The organization of the teams and contact information of the organizers, technical experts, LLs and LGLs is on the shared task organigram. In order to communicate with LGLs and other members of the shared task, we advise that LLs register and use our communication tools.

The main tasks carried out by the LL are:

- Recruiting and training annotators
- Maintaining the list of guidelines examples in the language
- Collecting feedback on the guidelines and transferring it to the LGLs
- Choosing and preparing the corpus for annotation
- Coordinating the annotations
- Performing the consistency checks after the annotation
- Preparing the CoNLL-U files
- Administering the Gitlab issues assigned to the language
- Co-authoring publications related to the shared task

2. What are the tools available for language teams in the shared task?

Annotation is performed on FLAT, a web platform which allows textual annotation and categorization of MWEs, including overlapping and discontiguous units. More details can be found on the FLAT user guide.

The shared task infrastructure also includes communication tools. Individual communication or communication that only concerns a small group of people are usually done by email or, if required, in videoconference meetings (e.g. skype). The contact information of the shared task members can be found on the shared task organigram. For issues concerning the annotation guidelines and their problems and enhancements, we have an issue system on gitlab. Therefore, we advise all LLs and those annotators who would like to participate in the discussions to register to gitlab.com and send their username to the organizers. We will include you in the parseme gitlab group and you will be able to create, answer and solve issues. For issues related to ongoing annotation and the FLAT platform, LLs and annotators can use the Telegram Helpdesk group.

For sharing data and scripts, we use gitlab.com repositories, including:

- parseme/sharedtask-guidelines, hosting the HTML guidelines and issues page
- parseme/utilities, hosting scripts useful for LLs before and after annotation
- parseme/sharedtask-data, hosting the corpora that are publicly available to participants

A detailed description of the repositories and tools can be found here.

3. How to use and adapt the annotation guidelines to your language?

The annotation guidelines specify how annotators should identify and categorize verbal MWEs. They were conceived to be universal, that is, all languages should be able to use the same guidelines. Therefore, annotator teams should try to follow them strictly, regardless of their previous background or the particular linguistic theory they follow. There should be as little personal interpretation as possible.

However, some languages may have phenomena that are not correctly covered or not covered at all by the guidelines. Moreover, the guidelines are far from being perfect and doubts may arise for unclear tests, borderline cases, etc. Therefore, LLs and annotators can (and should!) participate in enhancing the guidelines. They can contact their LGLs directly about an issue, or they can create an issue on the gitlab guidelines repository.

The annotation guidelines in HTML version can host examples in each participating language. To toggle the display of examples in a given language, click on the language code on the top of the screen. To add new examples in your own language, follow the instructions.

- Deadline for filling in examples in your language: **31 July 2017**
- Contact: Carlos and Silvio

4. How to organize your annotator team?

Annotator teams for a given language may be of variable size. A team should ideally have at least 2 annotators (e.g. the LL + another annotator) so as to allow double annotation of a subset of files, for the sake of inter-annotator agreement (IAA) calculation.
The following steps are necessary to prepare your team, if you are using the FLAT annotation platform:

1. Ask Behrang to register each annotator to FLAT. When this is done, inform the FLAT group administrator (Carlos or Agata), who will add each new annotator to your language group on FLAT.
2. Document the names and FLAT usernames of each annotator in the organigram.
3. Make sure your annotators are familiar with the annotation guidelines and with the FLAT user's guide.
4. Make sure they know how to view language-specific examples in the guidelines and complete them via the multilingual example table.

5. How to choose your corpus?

Each LL is in charge of choosing and preparing the corpus for annotation. We give some hints and best practices to optimize this task.

- Size: the objective is to have a final corpus of at least 3,500 annotated VMWEs per language. Since the density of VMWEs highly depends on the particular language, as well as text choice and genre, no reliable estimation of the corpus size is terms of number of tokens can be given. For example, in Brazilian Portuguese, we found on average 1 VMWE for every 6 sentences in the shared task corpus of edition 1.0. Generally speaking, we expect the **density** of VMWEs to be **rather low**.

- Genre: the corpus should contain newspaper texts, Wikipedia pages or alike, dedicated to no specific technical domain (general news rather than sports, weather forecasts, economics, etc.).
- Translationese issues: The corpus should be written in the original (rather than translated from another language)
- Source: in the perspective of future editions of the shared task on joint MWE identification and parsing, a corpus that is already (manually) syntactically annotated would be preferable.
- License issues: the corpus should be free from copyright issues so as to allow publication under a free license such as Creative Commons.
- Pre-selection strategies: In order to highlight the existing VMWE-related issues, and to avoid bias in annotation and evaluation, the corpus should contain both positive and negative examples of MWE occurrences. Therefore, it should be a running text rather than a set of automatically pre-selected sentences that would maximize the number of MWE occurrences. If the automatic pre-selection cannot be avoided, make sure that you also pre-select negative examples.
- Noisy data: The corpus should be kept in its original state. Notably, possible spelling, grammar or punctuation errors should not be corrected. On the other hand, texts from highly non-normalized sources such as tweets, SMS, chats or spontaneous speech transcriptions should be avoided.
- Pre-annotations: existence of automatically performed pre-annotations is allowed but you will need to beware the bias which they introduce. Make sure that the annotators do not overestimate the system's performances, and that they review the whole text, not only the pre-annotated candidates proposed by the system. A particularly useful pre-annotation concerns automatically underlining all verbs present in the corpus. A dedicated tag in FLAT can be defined for this purpose, so that verbs are highlighted for annotators on FLAT.
- Morphological and syntactic data - if you have access to a corpus that fulfills the above conditions, and additionally has already been annotated for morphology, lemmas and/or syntax, you are strongly encouraged to provide these data. These will be distributed within the ST closed track, whenever available, so as to enable a better VMWE identification quality. You can also perform an automatic morphological and/or syntactic annotation of the selected corpus prior to the VMWE annotation either with your own tools or with generic libraries like UDPipe for instance. The technical team can help with any difficulties you may find in this step.
- Dialects: for languages with large dialects (English, Spanish etc.), select corpora from the dialects for which there is at least one native annotator in the ST language team.

6. How to technically prepare a new corpus for annotation?

Once the corpus has been chosen, it should undergo several preparation steps in order to get ready for annotation:

1. Organize your corpus into a set of files. There is no hard restriction on the size of individual files, the FLAT web interface is optimized so as to handle big files with no delay. But for the reasons of a better distribution among annotators it would be interesting to split big files into smaller fragments (500-1000 sentences).
2. Ensure that all data uses UTF-8 encoding. Use conversion tools such as *iconv* if this is not the case.
3. Name your files according to the following convention:
   <language-code>-<free-text>.<extension>
   Example: EN-Herald-Tribune-23.tsv
   According to the format of the file (see below), the possible extensions are:

.txt, .tsv, .folia.xml.

The filenames should always start with a letter, and have no occurrence of the following characters: , !, ", #, $, %, &, ', (,), *, +, ,, /, :, ;, <, =, >, ?, @, [, \,], ^, `, {, |, }, ~.

Please follow these conventions, otherwise the number of already annotated VMWEs cannot be automatically calculated.

4. Convert the files into the parseme-tsv format or parseme-tsv-pos format. The latter is useful if part-of-speech or VMWE pre-annotations are available. Notably, retaining verbal POS tags allows verbal tokens to be automatically highlighted in FLAT, which can greatly speed up the manual annotation.
5. If lemmas, morphological tags and/or syntactic relations are available for the same files (see Section 5 above), they should be provided in CoNLL-U files, perfectly aligned with the parseme-tsv files, i.e. having the same number of lines, the same tokenization, and the same token in the same line.
    a. One solution here is to create parseme-tsv-pos based on a CoNLL-U (for example, if you are annotating an UD corpus, that is already in CoNLL-U format).
    b. Another solution is to use UDPipe to create CoNLL-U files for your corpus (see *"How to prepare the final annotated (training and test) corpus"* below).
    c. In both cases, the script checkSentenceMatching.py is available to check and enhance the consistency of your parseme-tsv and CoNLL-U files (see *"How to prepare the final annotated (training and test) corpus"* below).
6. Place your parseme-tsv(-pos) files and your CoNLL-U files (if any) into the relevant file directory (your-language-group/your-language/**1-TSV-untouched** and your-language-group/your-language/**4-conll**). This step is optional for the LLs managing their files locally.
7. Assign the files to annotators in the final annotation sheet (follow the link for your language family, select the sheet for your language. This step is optional for the LLs managing their files locally.
8. Take care of assigning a subset of files to two annotators, to enable the IAA calculation.
9. Upload the files into your annotators' namespaces in FLAT or send these instructions to annotators so that they can do it themselves. To add a file on FLAT, follow these steps:
    a. Log in to the server (using the right configuration, depending on your language)
    b. You should now see the namespaces of all the annotators in your language team. If this is not the case, contact the technical support.
    c. Select the namespace of one of your annotators. Click on Upload Document. Select the TSV file. Select the PARSEME TSV input format. Click on Upload. The document (converted to the XML Folia format) should now appear in the annotator's namespace.
10. Inform the annotators that the corpus is ready for annotation.
11. You can repeat these steps when new files become available.

7. How to re-annotate a corpus that was already annotated with FLAT?

Language teams may choose to re-annotate or check the annotations of files that were already annotated with FLAT in previous editions of the shared task. Several options are possible here, and the decision on how to proceed should be taken on a case by case basis.

For re-annotating (parts of) the corpus with FLAT, this can be performed either by making a copy of the original Folia files or by re-importing the files from the released PARSEMETSV corpus. We advise using the latter option because then you guarantee that you start from the very last version, generated after all consistency checks were performed. In this case, the procedure is quite simple:

1. Split the PARSEMETSV file into fragments of 500-1000 sentences. Attention, do NOT use a command such as Linux's *split* because the PARSEMETSV format has one token, and not one sentence per line. Rather use *awk* or some more sophisticated scripting language for that. The technical team can help if needed.
2. Do not forget to follow the file naming conventions and all other instructions that apply to a new corpus.
3. Import each file to FLAT as explained above: the existing annotations will show on the interface and you can edit them and add new ones.

The above steps are particularly useful if you need to annotate some parts of the corpus from scratch, e.g. because you want to extend your corpus or because you judge some enhancements of the guidelines substantial enough to require a full re-annotation (hopefully not!).
If, however, you judge that the modifications to be performed on the previously annotated corpus are limited to the existing annotations, another option is to use the consistency checking scripts. They allow you to automatically locate potential problems and then correct them manually or automatically (for simple corrections). In this case, just follow the instructions for the preparation of the final corpus. Remember that the *folia2parsemetsv.py* script, in spite of its name, also takes PARSEMETSV files directly as *--input*, so you do not need to have access to the original folia files to work with it.

8. How to modify your corpus during annotation?

It might occur that, while annotating a corpus, you realize that it contains some incorrect or spurious data (e.g. an accidentally inserted sentence or word, etc.). Correcting such errors is not currently possible with FLAT, and must be done manually as a post-annotation step directly in the downloaded file (in the Folia or parseme-tsv format).

9. How to update statistics about your corpus?

1. When preparing a new file for annotation, insert its number of tokens and sentences in the final annotation sheet (see the sheet with your language name). This step is optional for the LLs managing their files locally.
2. Periodically, visit the document index of your annotators in FLAT and:
   a. Mark in the final annotation sheet which files have been completed. The *Files* sheet will automatically update the statistics about the advancement of your corpus annotation.
   b. Check the numbers of the annotated VMWEs on the statistics page and copy them to the final annotation sheet.

   These 2 steps are optional for the LLs managing their files locally.

10. How to download files?

Downloading the annotated files is not necessary, since they remain on the server and can be downloaded by the shared task technical team. But you can also do it for your own use.

1. Go to the document index, select the namespace of an annotator. Download a (complete) file to the file directory (jour-language-group/your-language/**2-Folia-complete**) or to your local directory.
2. The folia2p

3. arsemetsv.py script converts an annotated Folia file into the parseme-tsv format. You may use it to convert your files and place them into the file directory (jour-language-group/your-language/**1-TSV-complete**) or to your local directory. But this can also be done automatically at the end of all annotations, by the technical experts.

11. How to modify the annotation team?

1. Adding a new annotator - follow the steps from Section 2 (FLAT registration, username documentation, training)
2. Deleting an annotator - check which files assigned to the annotator are complete and document them in the final annotation sheet. Mark the untouched files assigned to this annotator as unassigned. This step is optional for the LLs managing their files locally.

12. How to modify the file assignment?

1. Adding a new file - follow steps 2-10 from Section 6, assign the file to an annotator in the final annotation sheet. This step is optional for the LLs managing their files locally.
2. Re-assigning - if file F is assigned to annotator A1, and you wish to assign it to annotator A2, login to FLAT and go to A1 document index, select F, select A2 next to Move. Click on Move.

13. How to perform consistency checks and prepare the final corpora?

The following steps require:

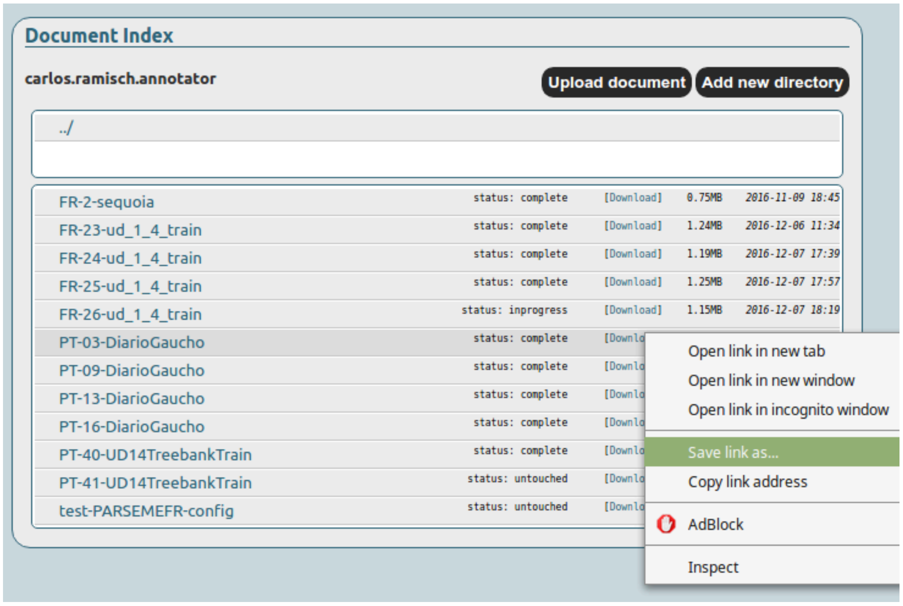
- python 3
- Linux
- The latest version of the pynlpl library
- The development versions of the libxml2 and libxslt1 libraries

To fulfil these requirements, you may run the following commands:

```
sudo apt-get install python3-pip
sudo apt-get install python-dev libxml2-dev libxslt1-dev zlib1g-dev
sudo pip3 install pynlpl
          or
sudo pip3 install --upgrade pynlpl
```

1. Download all Folia XML files from FLAT for your language, from all annotators' document index
    a. Unfortunately, each file must be downloaded individually, it's a bit tedious. Alternatively, you can ask a member of the technical team to provide a .zip with all files at once.
    Note: you must download the files as XML. If your browser wants to download an HTML file, click on

[Download] to open the XML page and then right-click and select "Save as".



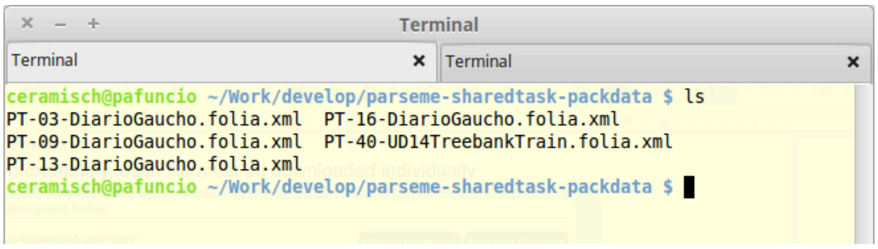
Document Index

carlos.ramisch.annotator

../

FR-2-sequoia	status: complete	[Download]	0.75MB	2016-11-09 18:45
FR-23-ud_1_4_train	status: complete	[Download]	1.24MB	2016-12-06 11:34
FR-24-ud_1_4_train	status: complete	[Download]	1.19MB	2016-12-07 17:39
FR-25-ud_1_4_train	status: complete	[Download]	1.25MB	2016-12-07 17:57
FR-26-ud_1_4_train	status: inprogress	[Download]	1.15MB	2016-12-07 18:19
PT-03-DiarioGaucho	status: complete	[Downlo		
PT-09-DiarioGaucho	status: complete	[Downlo		
PT-13-DiarioGaucho	status: complete	[Downlo		
PT-16-DiarioGaucho	status: complete	[Downlo		
PT-40-UD14TreebankTrain	status: complete	[Downlo		
PT-41-UD14TreebankTrain	status: untouched	[Downlo		
test-PARSEMEFR-config	status: untouched	[Downlo		

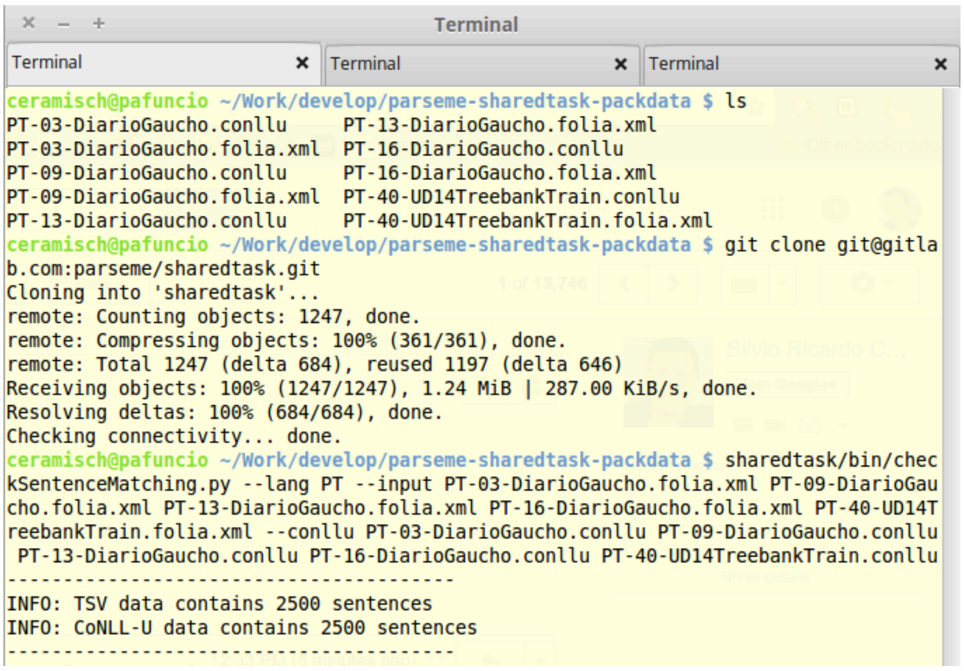
b. Place all files in the same folder, for instance, a new folder named *parseme-sharedtask-packdata*



```
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ ls
PT-03-DiarioGaucho.folia.xml  PT-16-DiarioGaucho.folia.xml
PT-09-DiarioGaucho.folia.xml  PT-40-UD14TreebankTrain.folia.xml
PT-13-DiarioGaucho.folia.xml
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ 
```

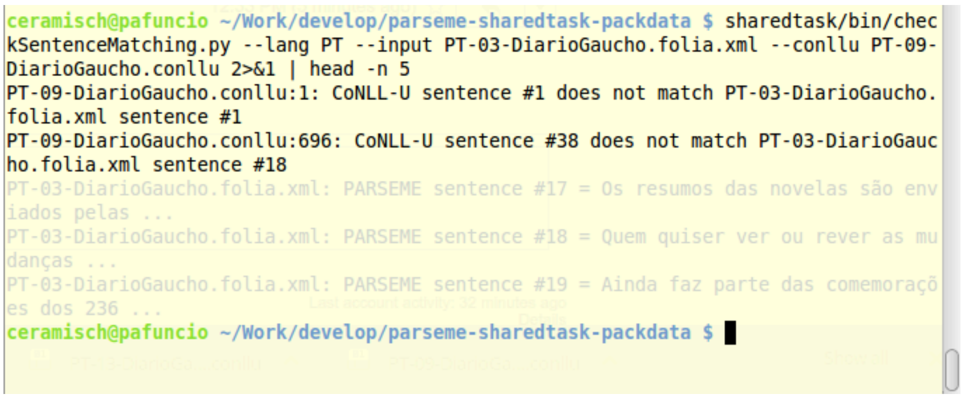
2. Collect or generate all corresponding CoNLL-U files, if they are available:
    a. The proper CoNLL-U format is **required** (with 10 columns, each column having the same semantics as in CoNLL-U.
    b. The UD tagset (for POS-tags, morphological features, and/or dependency information) is **recommended** but not required.
    c. If you do not have a CoNLL-U file, you can generate it using a parser like UDPipe. We can help if necessary, write to Silvio and Carlos.
3. Make sure that the sentences in the Folia files and their corresponding CoNLL-U files are aligned. You can check this by running the checkSentenceMatching.py script. This script indicates problematic unaligned sentences (it performs internal token-alignment in order to avoid spurious warnings).
    a. Attention: the script requires libraries from the *parseme/utilities* repository, and in particular those in the folder *lang-leaders*. Please, download or **pull the latest version** of the repository before going on. If you do not have access to the repository, register to GitLab.com and send us your username.
    b. Attention: the shared task scripts require Python3 and won't work with Python2.x. You should run them by simply calling the script directly.

When you run the script on correctly aligned files, the result should be as shown below.



```
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ ls
PT-03-DiarioGaucho.conllu     PT-13-DiarioGaucho.folia.xml
PT-03-DiarioGaucho.folia.xml  PT-16-DiarioGaucho.conllu
PT-09-DiarioGaucho.conllu     PT-16-DiarioGaucho.folia.xml
PT-09-DiarioGaucho.folia.xml  PT-40-UD14TreebankTrain.conllu
PT-13-DiarioGaucho.conllu     PT-40-UD14TreebankTrain.folia.xml
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ git clone git@gitla
b.com:parseme/sharedtask.git
Cloning into 'sharedtask'...
remote: Counting objects: 1247, done.
remote: Compressing objects: 100% (361/361), done.
remote: Total 1247 (delta 684), reused 1197 (delta 646)
Receiving objects: 100% (1247/1247), 1.24 MiB | 287.00 KiB/s, done.
Resolving deltas: 100% (684/684), done.
Checking connectivity... done.
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ sharedtask/bin/chec
kSentenceMatching.py --lang PT --input PT-03-DiarioGaucho.folia.xml PT-09-DiarioGau
cho.folia.xml PT-13-DiarioGaucho.folia.xml PT-16-DiarioGaucho.folia.xml PT-40-UD14T
reebankTrain.folia.xml --conllu PT-03-DiarioGaucho.conllu PT-09-DiarioGaucho.conllu
 PT-13-DiarioGaucho.conllu PT-16-DiarioGaucho.conllu PT-40-UD14TreebankTrain.conllu
----------------------------------------
INFO: TSV data contains 2500 sentences
INFO: CoNLL-U data contains 2500 sentences
----------------------------------------
```

In case of problems, the script indicates where the problems are, as shown below:



```
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ sharedtask/bin/chec
kSentenceMatching.py --lang PT --input PT-03-DiarioGaucho.folia.xml --conllu PT-09-
DiarioGaucho.conllu 2>&1 | head -n 5
PT-09-DiarioGaucho.conllu:1: CoNLL-U sentence #1 does not match PT-03-DiarioGaucho.
folia.xml sentence #1
PT-09-DiarioGaucho.conllu:696: CoNLL-U sentence #38 does not match PT-03-DiarioGauc
ho.folia.xml sentence #18
PT-03-DiarioGaucho.folia.xml: PARSEME sentence #17 = Os resumos das novelas são env
iados pelas ...
PT-03-DiarioGaucho.folia.xml: PARSEME sentence #18 = Quem quiser ver ou rever as mu
danças ...
PT-03-DiarioGaucho.folia.xml: PARSEME sentence #19 = Ainda faz parte das comemoraçõ
es dos 236 ...
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $
```

4. Perform the annotation consistency check. This step can greatly improve the coherence and thus the quality of annotations. Therefore, we recommend that you stop annotation sooner, if required, but use 1-2 days to perform this minimal consistency checks:
   a. Run folia2consistencyCheckWebpage.py. This script generates a webpage (below: webpage1.html) that groups all VMWEs based on their occurrences.
      If CoNLL-U files are available, the script is to be called in this way:

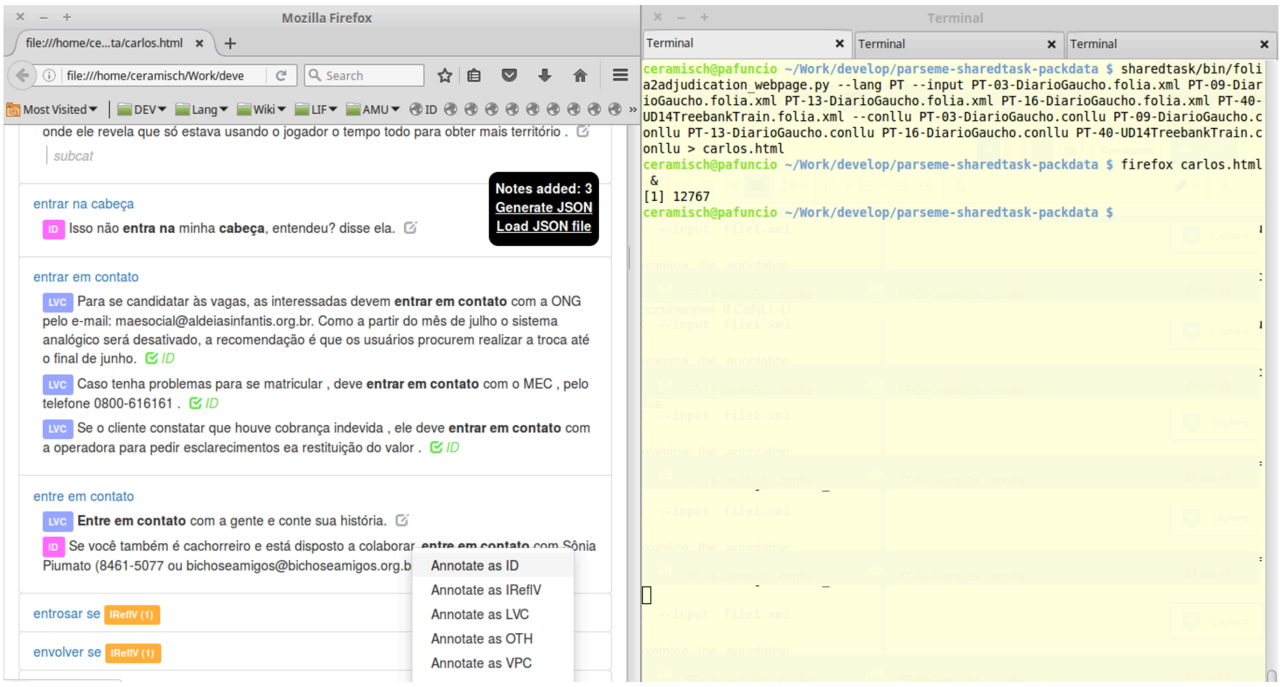
```
./folia2consistencyCheckWebpage.py --lang=XX  --find-skipped
       --input  file1.folia.xml  file2.folia.xml  ..  fileN.folia.xml
       --conllu file1.conllu  file2.conllu  ..  fileN.conllu
       > webpage1.html
```

      If no CoNLL-U files are available, the script is called like this:

```
./folia2consistencyCheckWebpage.py --lang=XX  --find-skipped
       --input  file1.folia.xml  file2.folia.xml  ..  fileN.folia.xml
       > webpage1.html
```

In both cases, XX should be replaced by your language code: EN, FR, DE, etc.

We strongly suggest using the option --find-skipped to locate unannotated sequences of tokens that were annotated somewhere else. This can be useful to detect some silence in your data. However, it only locates tokens that are not too far apart, and many will be false positives (so each one should be manually checked).



```
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ sharedtask/bin/foli
a2adjudication_webpage.py --lang PT --input PT-03-DiarioGaucho.folia.xml PT-09-Diar
ioGaucho.folia.xml PT-13-DiarioGaucho.folia.xml PT-16-DiarioGaucho.folia.xml PT-40-
UD14TreebankTrain.folia.xml --conllu PT-03-DiarioGaucho.conllu PT-09-DiarioGaucho.c
onllu PT-13-DiarioGaucho.conllu PT-16-DiarioGaucho.conllu PT-40-UD14TreebankTrain.c
onllu > carlos.html
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ firefox carlos.html
 &
[1] 12767
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $
```

onde ele revela que só estava usando o jogador o tempo todo para obter mais território .

subcat

Notes added: 3
Generate JSON
Load JSON file

entrar na cabeça

ID Isso não **entra na** minha **cabeça**, entendeu? disse ela.

entrar em contato

LVC Para se candidatar às vagas, as interessadas devem **entrar em contato** com a ONG pelo e-mail: maesocial@aldeiasinfantis.org.br. Como a partir do mês de julho o sistema analógico será desativado, a recomendação é que os usuários procurem realizar a troca até o final de junho. ✓ ID

LVC Caso tenha problemas para se matricular , deve **entrar em contato** com o MEC , pelo telefone 0800-616161 . ✓ ID

LVC Se o cliente constatar que houve cobrança indevida , ele deve **entrar em contato** com a operadora para pedir esclarecimentos ea restituição do valor . ✓ ID

entre em contato

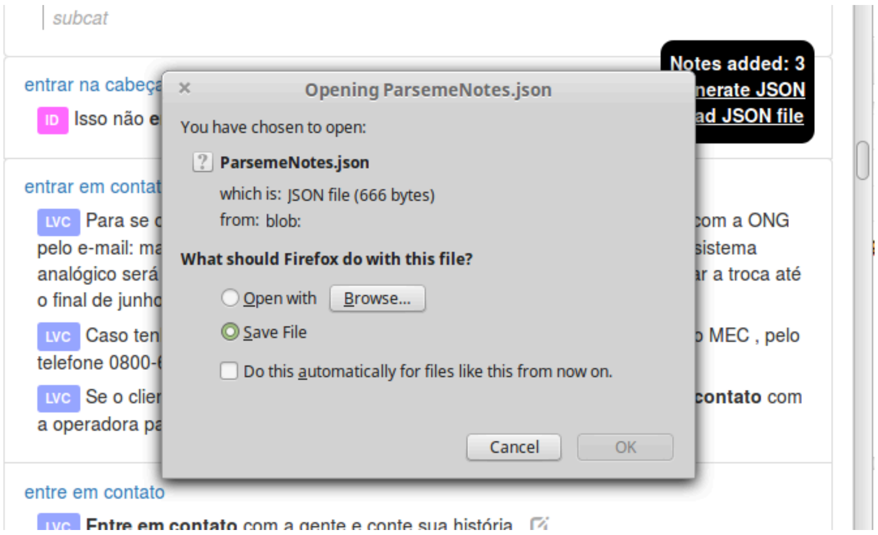
LVC **Entre em contato** com a gente e conte sua história.

ID Se você também é cachorreiro e está disposto a colaborar **entre em contato** com Sônia Piumato (8461-5077 ou bichoseamigos@bichoseamigos.org.b

entrosar se IReflV (1)

envolver se IReflV (1)

b. Open webpage1.html in a browser, examine the annotation inconsistencies, indicate the modifications if needed by clicking on the edition icon after each sentence, and create a JSON file with modifications by clicking on "Generate JSON". You can stop and restart when you want, since it is possible to save and load the JSON file with the modifications at any time. Save modifications often ;-)



Opening ParsemeNotes.json

You have chosen to open:

ParsemeNotes.json

which is: JSON file (666 bytes)

from: blob:

What should Firefox do with this file?

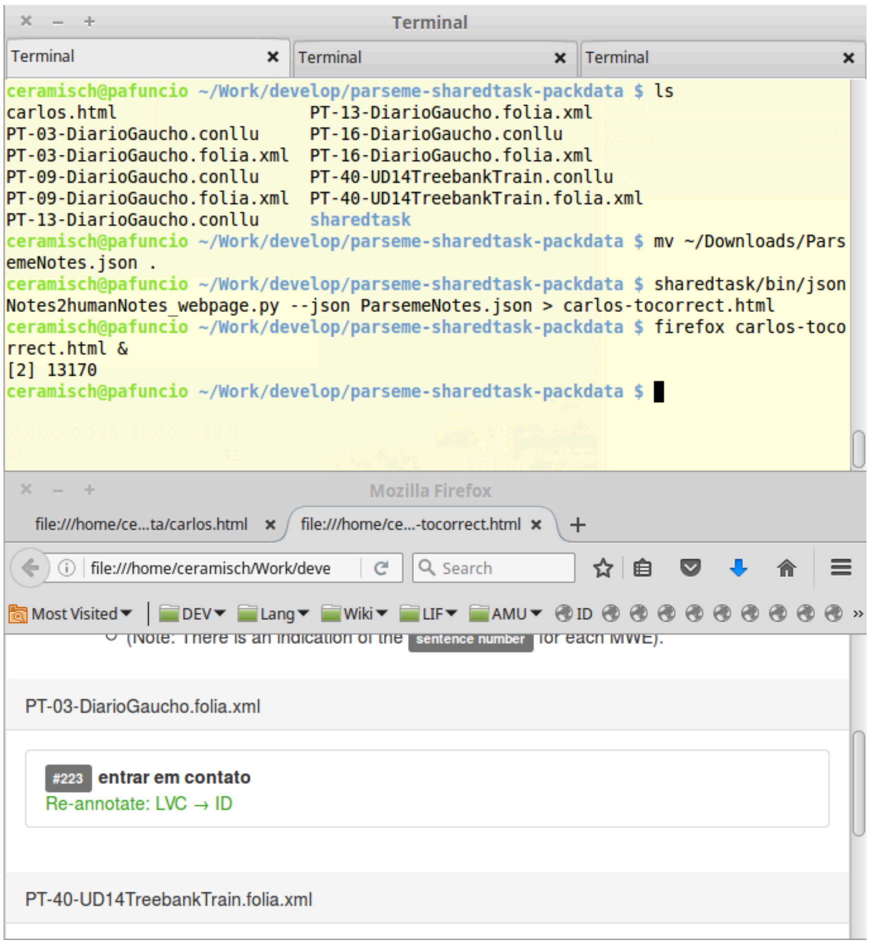
Open with [Browse...]

[✓ Save File]

Do this automatically for files like this from now on.

c. You can the script jsonNotes2ReannotationWebpage.py with the previously generated JSON file as a parameter to the flag `--json-input`. This produces `webpage2.html` with names of files and the list of sentences to be changed for each file. **However, to speed up the**

application of the modifications, we advise you to use the automatic correction option proposed below (item (d)).



```
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ ls
carlos.html                  PT-13-DiarioGaucho.folia.xml
PT-03-DiarioGaucho.conllu    PT-16-DiarioGaucho.conllu
PT-03-DiarioGaucho.folia.xml PT-16-DiarioGaucho.folia.xml
PT-09-DiarioGaucho.conllu    PT-40-UD14TreebankTrain.conllu
PT-09-DiarioGaucho.folia.xml PT-40-UD14TreebankTrain.folia.xml
PT-13-DiarioGaucho.conllu    sharedtask
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ mv ~/Downloads/Pars
emeNotes.json .
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ sharedtask/bin/json
Notes2humanNotes_webpage.py --json ParsemeNotes.json > carlos-tocorrect.html
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ firefox carlos-toco
rrect.html &
[2] 13170
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ 
```

Terminal
Mozilla Firefox
file:///home/ce...ta/carlos.html
file:///home/ce...-tocorrect.html
file:///home/ceramisch/Work/deve
Search
Most Visited
DEV
Lang
Wiki
LIF
AMU
ID

(Note: There is an indication of the sentence number for each MWE).

PT-03-DiarioGaucho.folia.xml

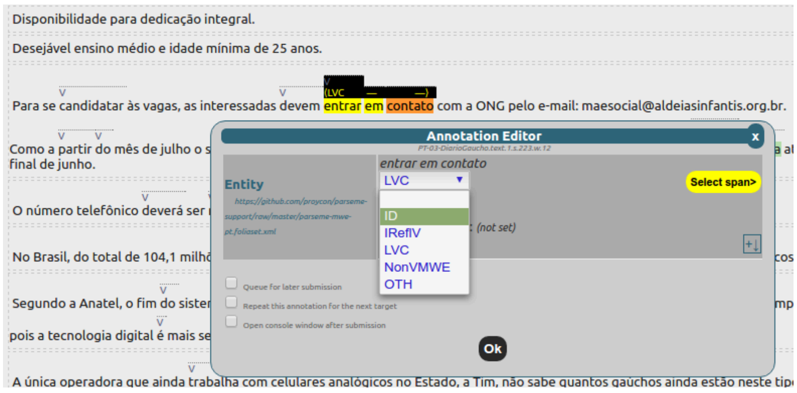
#223 **entrar em contato**
Re-annotate: LVC → ID

PT-40-UD14TreebankTrain.folia.xml

d. A certain number of simple modifications can be performed automatically. This includes changing labels and removing words from the span. If you want to apply modifications automatically, saving a lot of time, you should proceed as follows:

   i. Re-run step (c) above, this time indicating to jsonNotes2ReannotationWebpage.py the names of all folia xml files, using the `--xml-input` option. The output `webpage2.html` file will only show the modifications that cannot be made automatically (there is a button to see what MWEs will be able to be automatically annotated). We will start by performing the *manual* reannotation.

   ii. Modifications indicated on this webpage are usually special cases, and cannot be automatically applied. You should go to FLAT, open the indicated file, locate the sentence and manually apply the modifications required in `webpage2.html`.



Disponibilidade para dedicação integral.

Desejável ensino médio e idade mínima de 25 anos.

(LVC — —)
Para se candidatar às vagas, as interessadas devem entrar em contato com a ONG pelo e-mail: maesocial@aldeiasinfantis.org.br.

Como a partir do mês de julho o s … a al
final de junho.

O número telefônico deverá ser

No Brasil, do total de 104,1 milhõ … cos

Segundo a Anatel, o fim do sister … mp

pois a tecnologia digital é mais se

A única operadora que ainda trabalha com celulares analógicos no Estado, a Tim, não sabe quantos gaúchos ainda estão neste tipo

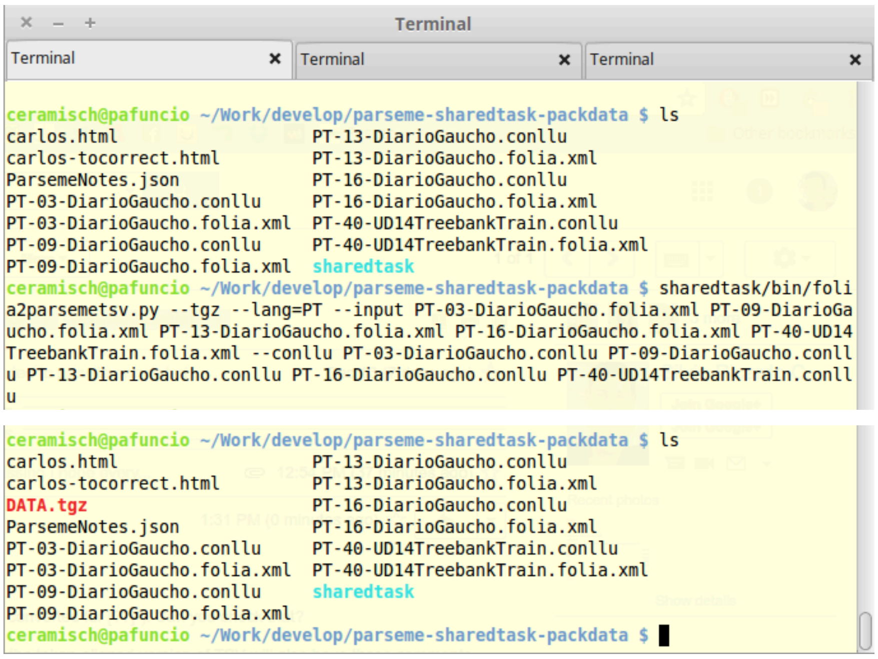
Annotation Editor
PT-03-DiarioGaucho.text.1.s.223.w.12
entrar em contato
Entity
https://github.com/proycon/parseme-support/raw/master/parseme-mwe-pt.foliaset.xml
LVC
ID
IReflV
LVC
NonVMWE
OTH
Select span>
: (not set)
Queue for later submission
Repeat this annotation for the next target
Open console window after submission
Ok

  iii. Download all Folia XML files again (since they have just been manually reannotated), *replacing* the previous ones.
  iv. Run jsonNotes2ReannotationWebpage.py passing these new XML files to the `--xml-input` option. Also specify the `--generate-xml` flag. This will re-generate a spurious HTML list of modifications that cannot be made automatically, but you can ignore it (you have just performed these modifications manually). This script will also create a folder named `AfterAutoUpdate/` with the final data.

5. Apply folia2parsemetsv.py script to all Folia files in `AfterAutoUpdate/`:

```
./folia2parsemetsv.py --tgz  --lang=XX --input file1.folia.xml file2.folia.xml ..
        fileN.folia.xml --conllu file1.conllu file2.conllu .. fileN.conllu
```

   If the CONLL files are given, the output file is adjusted with regard to contractions and comments (lines starting with "#").The output is a DATA.tgz archive with 2 files:
   a. A *.parsemetsv* file with all VMWE annotations
   b. A *.conllu* file with all CoNLL-U data aligned with the VMWE data



```
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ ls
carlos.html                 PT-13-DiarioGaucho.conllu
carlos-tocorrect.html       PT-13-DiarioGaucho.folia.xml
ParsemeNotes.json           PT-16-DiarioGaucho.conllu
PT-03-DiarioGaucho.conllu   PT-16-DiarioGaucho.folia.xml
PT-03-DiarioGaucho.folia.xml  PT-40-UD14TreebankTrain.conllu
PT-09-DiarioGaucho.conllu   PT-40-UD14TreebankTrain.folia.xml
PT-09-DiarioGaucho.folia.xml  sharedtask
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ sharedtask/bin/foli
a2parsemetsv.py --tgz --lang=PT --input PT-03-DiarioGaucho.folia.xml PT-09-DiarioGa
ucho.folia.xml PT-13-DiarioGaucho.folia.xml PT-16-DiarioGaucho.folia.xml PT-40-UD14
TreebankTrain.folia.xml --conllu PT-03-DiarioGaucho.conllu PT-09-DiarioGaucho.conll
u PT-13-DiarioGaucho.conllu PT-16-DiarioGaucho.conllu PT-40-UD14TreebankTrain.conll
u

ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ ls
carlos.html                 PT-13-DiarioGaucho.conllu
carlos-tocorrect.html       PT-13-DiarioGaucho.folia.xml
DATA.tgz                    PT-16-DiarioGaucho.conllu
ParsemeNotes.json           PT-16-DiarioGaucho.folia.xml
PT-03-DiarioGaucho.conllu   PT-40-UD14TreebankTrain.conllu
PT-03-DiarioGaucho.folia.xml  PT-40-UD14TreebankTrain.folia.xml
PT-09-DiarioGaucho.conllu   sharedtask
PT-09-DiarioGaucho.folia.xml
ceramisch@pafuncio ~/Work/develop/parseme-sharedtask-packdata $ █
```

6. Write a README file describing your data, based on the README for your language in the PARSEME edition 1.0. If your language was not part of edition 1.0, you can use the Portuguese README.md as a starting point. The README should contain the description of the corpus, tagsets used, annotation choices, licence, authors, etc.
7. Provide the DATA.tgz and the REDAME.md files per email to the shared-task organizers (parseme-st-core@nlp.ipipan.waw.pl).

8. Also, please indicate in your email which part of the data your TRIAL data comes from. This is important because we will split your data into TRAINING and BLIND/TEST, and we cannot include the TRIAL data into the BLIND/TEST data. We will take sentences corresponding to around 500 VMWEs for BLIND/TEST, and the remainder will be TRAINING data.
    a. Please do **not** commit files to gitlab yourself. We will perform some automatic checks, split training and blind/test, commit them to private `parseme/sharedtask-data-dev`, ask you to check it, then push it to public repository `parseme/sharedtask-data`.

14. How to prepare a double-annotated subcorpus for calculating the inter-annotator agreement?

The Inter-Annotator Agreement (IAA) is a classical measure to assess the methodology of an annotation campaign, and of the resulting annotated corpus. In order to calculate it, a fraction of the final corpus needs to be annotated by **two independent annotators**. The IAA results are to be reported on the shared task description paper (co-authored by all ST organizers and LLs).
Please, follow the following steps and send the double-annotated files to the organizers. Please, also notify **which of the scenarios** 1-4 below you followed.

1. If good fractions of the shared task corpus in your language have **already** been **double-annotated** by **exactly two annotators**.
    a. **Select** a set of double-annotated files which
        i. Contain at least **200-300 VMWEs** annotated by each annotator (FLAT statistics can help).
        ii. Were annotated close to the end of the annotation campaign (as to ensure the best possible quality).
        iii. Contain **original annotations**, which have not been adjudicated (see Section 15) or unified in any way. Namely, for the IAA it is important to work on results of **independent annotations**. We assume that common documentation on difficult cases and discussions among annotators, which were taking place during the annotation process, do not violate the principle of independent annotation.
    b. **Convert** the selected files to the cupt format and **merge** them so as to obtain only 2 files with the same number of lines and the same sentences in the same order. Each file should contain the annotations of the same annotator. Name the files `XX.double.1.cupt` and `XX.double.2.cupt`, where XX is your language code.
    If your annotations were done in FLAT, both the conversion and the merge can be done by the parsemetsv2cupt.py script. Run the two commands below adapting the names of the input files. It is important that the `--input` files are passed to both scripts in the same order to ensure perfect alignment. Please also provide corresponding CONLL-U files, since we need to know which tokens are verbs to calculate IAA:

```
./parsemetsv2cupt.py    --lang=XX    --input    file1-annotator1.folia.xml
      file2-annotator1.folia.xml      --conllu     file1-annotator1.conllu
      file2-annotator1.conllu > XX.double.1.cupt
./parsemetsv2cupt.py    --lang=XX    --input    file1-annotator2.folia.xml
      file2-annotator2.folia.xml      --conllu     file1-annotator2.conllu
      file2-annotator2.conllu > XX.double.2.cupt
```

   c. **Check** again that `XX.double.1.cupt` and `XX.double.2.cupt` have the same sentences in the same order. If they are not in the same order (or if some file has extra sentences that do not appear in the other), this should be manually fixed.
   d. If you have performed any **adjudication/correction** of the original annotations (see Section 15), you may also send us a third file `XX.double.gold.cupt` obtained in the same way as above in b-c above. This will allow us to also assess the amount of corrections performed for the final corpus.
   e. **Send** the 2 files to parseme-st-core@nlp.ipipan.waw.pl before **April 20, 2018**.

2. If good fractions of the shared task corpus in your language have **already** been **double-annotated** by **more than two annotators**.
   a. Follow the previous steps a-d for each of the annotators who annotated some fraction of your corpus.
   b. Apply the evaluation script of each pair of the resulting files:

      ```
      ./evaluate.py XX.double.1.cupt XX.double.2.cupt
      ```
   c. Select the pair of files which gives the **best F-measure** score, i.e. the best IAA in the task of VMWE identification (but not classification).
   d. Send this all `.cupt` files to parseme-st-core@nlp.ipipan.waw.pl before **April 20, 2018**.

3. If your corpus has **not** yet been **double-annotated** but you have at least **2 annotators** in your team.
   a. Select a set of files annotated by one annotator (A1, the one you consider particularly reliable), which:
      i. Contain at least **200-300** annotated **VMWEs** (the FLAT statistics can help).
      ii. Were annotated close to the end of the annotation campaign (co as to ensure the best possible quality).
      iii. Contain **original annotations**, i.e. have not been adjudicated, consistency-checked (see Section 15) or unified in any way.
   b. Retrieve the **non-annotated** versions of the same file and submit the same files for annotation by a different annotator (A2). If you use FLAT, this boils down to uploading the non-annotated files to the file index of annotator A2.
   c. When the annotation is ready, download the new files and follow steps 1b-e above before **April 20, 2018**.

4. If your corpus has **not** yet been **double-annotated** but you are the **only annotator** in your language - follow the steps in point 3 above, choosing files that you have previously annotated yourself. If you use FLAT, make sure the new files bear different names so as not to override the previous annotations. Don't forget to indicate that you are measuring self-IAA (scenario 4) in your email.

15. How to adjudicate annotations?

Two different operations have been called "adjudication" in the context of PARSEME:

1. Consistency checks (since edition 1.0).

   a. The goal is to compare all occurrences of the same MWE across the whole corpus, and then re-assign MWE categories so as to increase the consistency these occurrences.
   b. The whole corpus may have been annotated by one or multiple annotators, but each Folia XML file must only have been annotated by a single annotator (i.e. avoid parallel annotations of the same file by different annotators, as the script has not been tested for this use case).
   c. This is part of the guidelines for preparing the final annotated corpus: see Section 13.4.

2. Multi-annotator adjudication (since edition 1.1).
   a. The goal is to compare parallel annotations of the same file by different annotators (double-annotations), and to decide which annotation to be kept officially.
   b. This has only been implemented for double-annotation, so you need exactly two parallel version of your Folia XML file.
   c. Note: this tool is still experimental. Feedback is more than welcome.

   d. In order to perform adjudication, you need to create a webpage. Run this command:
```
./folia2annotatorAdjudicationWebsite.py --lang=XX
      --annotator-1 fileX-version1.folia.xml
      --annotator-2 fileX-version2.folia.xml
      >AdjudicationPage.html
```
   e. You must then perform the adjudication and generate a ParsemeNotes.json file.
   f. Once the file ParsemeNotes.json is generated, you should follow the same instructions as for the "consistency checks": see Section 13.4.
      i. You will use ParsemeNotes.json as input to jsonNotes2ReannotationWebpage.py, to manually or automatically re-annotate the file from annotator-1.
      ii. In the example above, the file from annotator-1 is called fileX-version1.folia.xml, so you would use `--xml-input fileX-version1.folia.xml`.

16. How to run UDPipe?

(Note: this section on UDPipe is experimental. It might be moved elsewhere in the future).

If your data does not originally have CoNLL-U information, you will likely want to generate one automatically. This can be done with UDPipe, using a pre-trained model for your language.

1. Download UDPipe models
   a. Note: the links below contain models for multiple languages, so get ready to download about 1GB of data in a big zip file (even though the model to your language will only have around 20 to 100MB)
   b. Links:
      i. Latest: https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2364/allzip
      ii. (Other models here)
2. Run UDPipe
   a. Assuming that MODELPATH is the path to your language's model (e.g. udpipe-ud-2.0-170801/romanian-ud-2.0-170801.udpipe)
   b. Run the following command:
```
1.1/lang-leaders/pre-annot/run_udpipe.sh MODELPATH file.parsemetsv
```

c. If this is the first time you run this script, it will automatically download and compile UDPipe (so it may take a couple of extra minutes to run).