1. RDF corpora

- Linguistic corpora are collections of texts with annotations: morphological, syntactic, semantic, etc.
- There are myriad ways to store and query corpora, as well as many tags even for the same language and the same type of annotation.
- Classic ways of corpora representation → low level of interoperability:
  - between different corpora
  - between corpora and other linguistic resources: dictionaries, thesauri, etc.
- Representing corpora as Linguistic Linked (Open) Data in RDF has many advantages:
  - Incorporation of any type of annotation on any stage of corpus development
  - Harmonising annotations between tagsets and storing multiple annotations for the same data
  - Linking to external resources, e.g. OntoLex-lemon dictionaries
- There are standards and evidence of representing corpora as Linked Data (NIF, CoNLL-RDF) but not of using it.
- Possible reason — the lack of user-friendly ecosystem.
- cqp4rdf is a first step on the way of creating the ecosystem for corpus-based linguistic research using LLOD corpus resources.

2. CQP query language

- CQP is a well-established and widely used query language for corpora.
- Ideal for extracting sequences of tokens with specific properties, i.e. a sequence of adjectives followed by one noun.
- The syntax is simple and concise:
  - Each token is denoted as { [tag-name] [pos="X"] [lemma="Y"] }.
  - Quantifiers such as +, *, ?, (n, m) can be applied: {[1,10]}.
  - Tokens can be filtered by their properties: (word="content" and pos="V"). String values are regular expressions: [pos="V.*"].
  - Segments are denoted in XML-style: <tag-name/> and are usually used in combination with special constructions, (not) containing (and) within tokens.
  - Tokens can be grouped by either: 1: [pos="A.*"] [lemma="and"] 2: [pos="P"] & 1: lemma=2: lemma returns constructions like more and more.
- We add namespaces to properties and segment names: conll:UPOS, nif:Sentence.
- List of namespases and optional declaration of possible properties are configured for a given corpus.
- This allows handling multiple tagsets or multiple corpora.

Examples

- A noun followed by a verb with 0 to 2 words in between: [conll:UPOS="NOUN"] [] [2] [conll:UPOS="VERB"]
- Pronouns with sequences of forms to have, to be or to do: [conll:UPOS="PRON"] [conll:LEMMA="have(be|do)"] [2]

3. cqp4rdf architecture

- cqp4rdf is designed to have modular architecture and consists of:
  - Frontend: interface where user interacts with corpora.
  - Backend: query conversion, getting and handling query results.
  - Administrative part: adding and configuring corpora.
- Decoupling the frontend gives the possibility to use it with better developed corpus manager interfaces.
- Operations specific to RDF representations of corpora are localised and are used only before querying the triplestore.
- In the current implementation, operations that modify data, e.g. linking or harmonising tagsets, are external to CQP.

4. Adding and configuring data

- One instance of cqp4rdf supports multiple corpora. Each has its section in the main configuration file, config.yaml.
- Global settings should include a pointer to a SPARQL endpoint of a triplestore with corpora.
- Adding a corpus is simple:
  - Insert RDF into a triplestore indicated in the config file.
  - Add a section describing a corpus into a configuration file. Necessary settings are the name, IRI with the corpus graph, and the list of prefixes.
- Additionally, it is possible to describe the data types and possible values for token properties, e.g. hide some fields or set a list of possible values for grammatical tags.

5. Current implementation

- A demo installation consists of a minimalistic frontend with a part of English Universal Dependencies corpus.
- Users can type a query and get results in a KWIC format, familiar for corpus linguists.
- Clicking on a token in the results, users can see all the properties of this token.
- The interface contains some example queries.
- Try it out at https://purl.org/tlod/cqp4rdf/ud, learn more at https://purl.org/tlod/cqp4rdf

6. Future directions

- In the current implementation, SPARQL queries are slow. → Benchmarking and optimising are needed.
- We could benefit from indexing to speed up common queries.
- Usually, storage and search is highly optimised for large corpora.
- Still, even for non-RDF corpus managers complex queries take time to execute.
- cqp4rdf works well for small corpora of less-resourced languages.
- Further experiments will show the range of its applicability.

- Adding and managing corpora via a dedicated interface would increase usability.
- Using CoNLL-RDF, we can import tab-separated corpus files (CoNLL format).
- Some data modification can be handled via SPARQL updates, they can be applied also using CoNLL-RDF.
- Providing a convenient way to apply predefined or custom updates will allow handling more advanced queries with less computational effort (i.e. adding multi-layered annotations for segments like sentences or named entities).

- To fully understand what is missing, we need use cases and corpus research done with cqp4rdf.
- To make this possible, one way would be to adapt existing corpus manager frontends.
- Meanwhile, we continue our tests with different tagsets and languages:
  - Eastern Armenian National Corpus,
  - Corpora of less-resourced languages collected by field linguists.