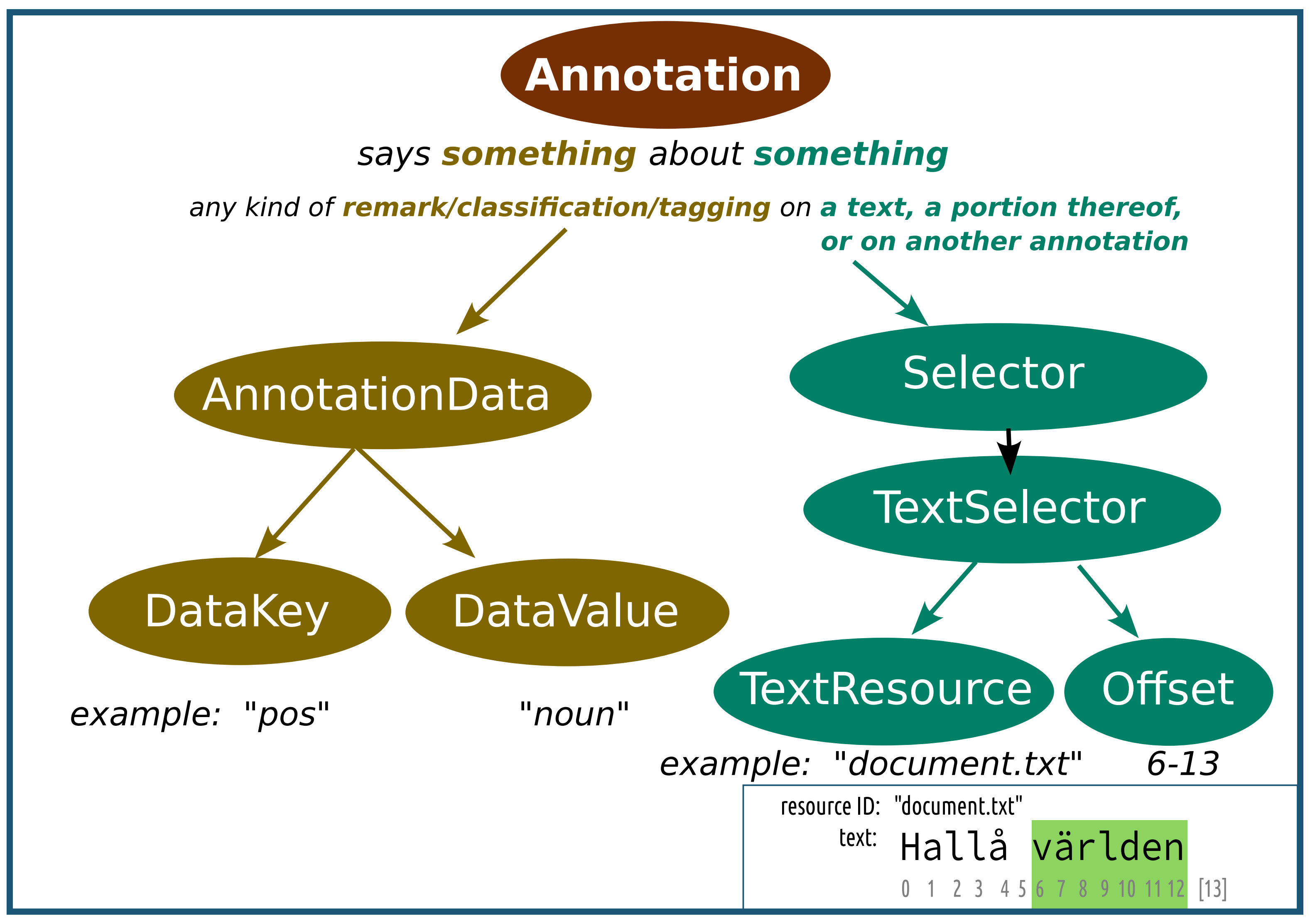
Clariah

STAM

httpa://annotation.github.io/stam

KNAW Humanities Cluster

by Maarten van Gompel



Data Model

***STAM** is a standalone data model for **stand-off annotation** on **text**. It allows you to describe annotations on text in your own terms. STAM **does not prescribe any vocabulary**.*

Texts are kept as-is (utf-8 plain text), devoid of any special markup, and annotations target text segments via character offsets. This approach can be called radical stand-off. Any further annotation paradigm decisions are up to you. STAM aims to be **generic** and **flexible**.

Annotation is the central notion in STAM; *almost everything is an annotation*, each has **annotation data** and selects a **target**. The data vocabulary is all up to you; not even the notion of a word, token or sentence is predefined. Annotations can represent whatever you want; be it linguistic, structural, presentational, editorial or otherwise.

Practical Tooling

STAM comes with **practical programming** libraries and **tools** to work with annotations on text. These aim to be fairly low-level standalone software with **minimal dependencies**, high **reusability**, and not reliant on any wider infrastructure.

- **stam-rust** is the main library written in Rust with a focus on performance (*Rust API*).
- **stam-python** is a Python library built on-top of the Rust library (*Python API*).
- a tutorial **"Standoff Text Annotation for Pythonistas"** is available as a jupyter notebook.
- **stam-tools** is a set of command-line tools to work with STAM.

```
$ pip install stam
```

```
$ cargo install stam-tools
```

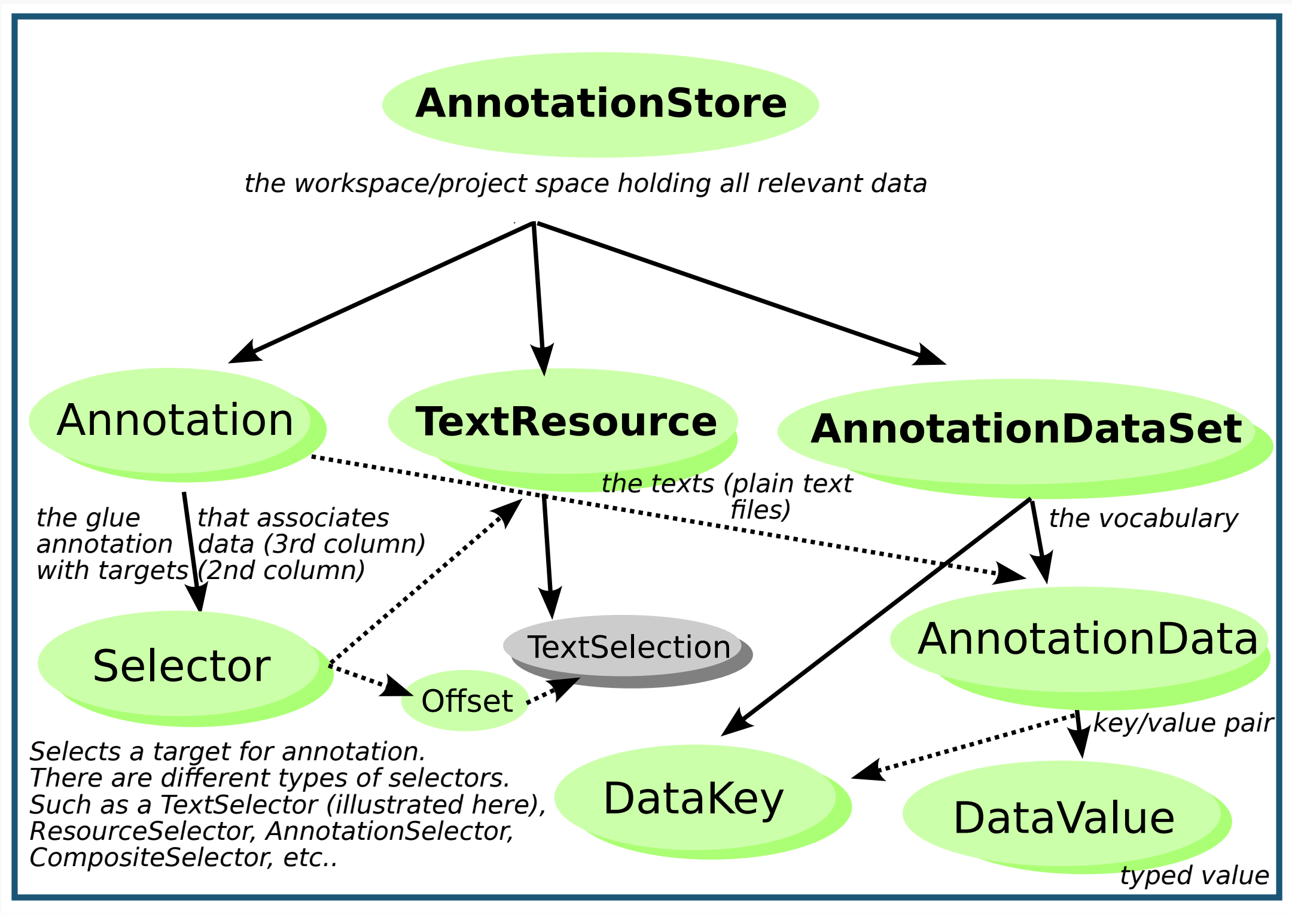
All are available as free open-source software under the GNU General Public Licence v3. The tools are aimed at developers and technical researchers and data scientists. You can build upon STAM to construct your own applications that deal with annotation on text, including but not limited to for purposes such as natural language processing (NLP).

```python
from stam import *

store = AnnotationStore(id="example")
resource = store.add_resource(
                 id="document.txt",
                 text="Hallå världen")
annotation = store.annotate(
   target=Selector.textselector(
              resource,
              Offset.simple(6,13)),
   data={"key": "pos", "value": "noun",
         "set": "testset" } )

store.set_filename(
              "example.stam.store.json")
store.save()

for annotation in store.annotations():
    for data in annotation.data():
        print(f"{annotation},
              {data.key()}={data.value()}")
```


AnnotationStore
the workspace/project space holding all relevant data
Annotation
TextResource
AnnotationDataSet
the texts (plain text files)
the glue annotation with targets
that associates data (3rd column) (2nd column)
the vocabulary
Selector
TextSelection
Offset
AnnotationData
key/value pair
Selects a target for annotation. There are different types of selectors. Such as a TextSelector (illustrated here), ResourceSelector, AnnotationSelector, CompositeSelector, etc..
DataKey
DataValue
typed value

Formal specification & extensibility

There is a core STAM model that is kept **minimal** yet **expressive**, and several optional **extensions** that define additional modelling capabilities you may or may not use. All are documented in a **formal specification**, independent of any implementations.

STAM provides a **solid generic foundation** for stand-off text annotation upon which other initiatives can build. The model is **specialised** for text annotation, not general knowledge graphs.

A **STAM Query Language** is defined for querying:

```
SELECT ANNOTATION ?a WHERE
       RESOURCE "mytext.txt";
       DATA "myvocab" "pos" = "noun";
```

Simplicity, reusability & interoperability

STAM provides a canonical **STAM JSON format**, a **STAM CSV format** and an optimised binary format. Data can also be easily imported from and exported to simple ad-hoc formats like TSV. The model itself is independent of any serialisation formats.

```json
{
  "@type": "AnnotationStore", "@id": "example",
  "resources": [ { "@type": "TextResource", "@id": "document.txt", "text": "Hallå världen" }],
  "annotationsets": [ { "@type": "AnnotationDataSet", "@id": "testset",
  "keys": [ { "@type": "DataKey", "@id": "pos" }],
  "data": [ { "@type": "AnnotationData", "@id": "!D0", "key": "pos", "value": { "@type": "String", "value": "noun"
  "annotations": [{
      "@type": "Annotation", "@id": "!A0",
      "target": {
        "@type": "TextSelector",
        "resource": "document.txt",
        "offset": { "@type": "Offset", "begin": { "@type": "BeginAlignedCursor", "value": 6 },
            "end": { "@type": "BeginAlignedCursor", "value": 13 } }
      },
      "data": [ { "@type": "AnnotationData", "@id": "!D0", "set": "testset" } ]
```

STAM may act as a pivot model in conversions between different annotation formats, paradigms and vocabularies. It does not seek to replace existing projects but offers a common foundation in which other vocabularies may be expressed, allowing you to benefit from a common machinery.

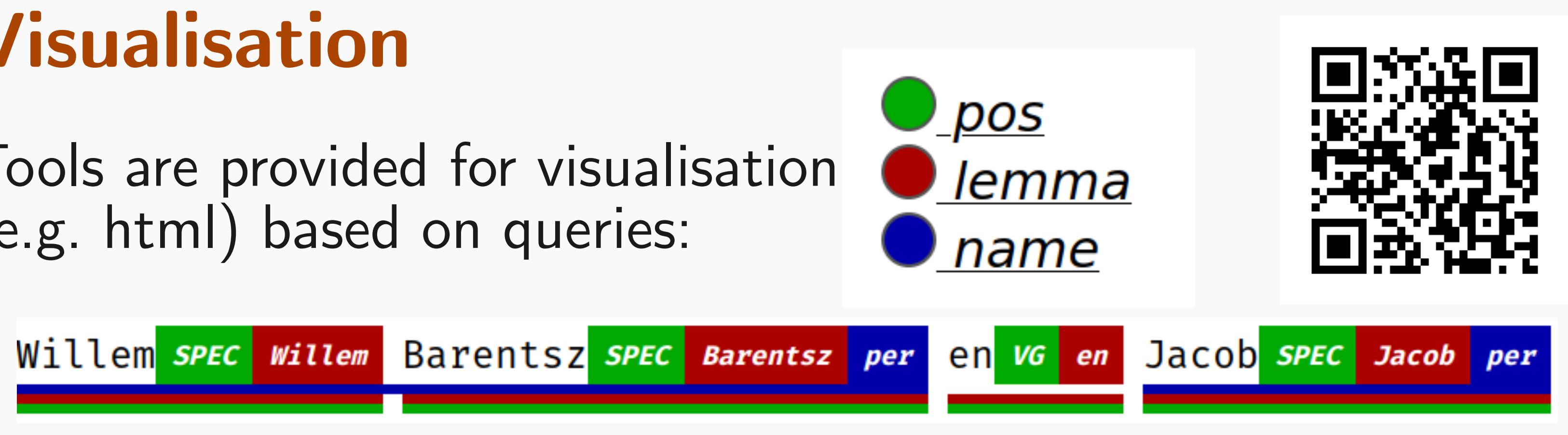
We seek a fair degree of **interoperability** with **W3C Web Annotations**, linked open data in general, **FoLiA**, **Text Encoding Initiative (TEI)**, **CoNLL-U** and **Text Fabric**. Various converters and mappings to this end have been implemented, such as a W3C Web Annotation exporter and a generic XML importer that effectively **untangles** an XML file with inline annotations/markup (like TEI) and splits it into plain text and pure stand-off annotations referencing that text.

Computing with Annotations

- Conversion between character offsets in a relative vs absolute frames, bytes and unicode points.
- Computation of spatial relationships between text selections/annotations, e.g. embedding, overlap, adjacency, and retrieval of annotations bases on such relationships.
- Common regular-expression based text search
- Automatic alignment of two or more similar texts (Smith-Waterman/Needleman-Wunsch) and automatic transposition of annotations from one text to another, following such an alignment.
- Validation of the integrity of stand-off annotations. Do they still point to the right text?

Visualisation

Tools are provided for visualisation (e.g. html) based on queries:


pos
lemma
name
Willem SPEC Willem
Barentsz SPEC Barentsz per
en VG en
Jacob SPEC Jacob per