Grobid-quantities Documentation

Release 0.4.2

Patrice Lopez <patrice.lopez@science-miner.com>, Luca Foppian

May 17, 2019

Contents

CHAPTER 1

Getting started

Building grobid-quantities and *JDK 1.8* (it ships Gradle wrapper out of the box).

1.1 Build and install

First install the latest development version of GROBID as explained by the documentation.

Copy the module quantities as sibling sub-project to grobid-core, grobid-trainer, etc.:

```
cp -r grobid-quantities grobid/
```

Try compiling everything with:

```
cd PATH-TO-GROBID/grobid/

./gradlew clean install copyModels
```

You should have the directories of the models *quantities* and *units* inside *../grobid-home/models*

Run some test:

```
cd PATH-TO-GROBID/grobid/grobid-quantities

./gradlew test
```

1.2 Start the service

Grobid quantities can be run as a service:

```
java -jar build/libs/grobid-quantities-{version}-onejar.jar server resources/config/
↪config.yml
```

Demo/console web app is then accessible at `http://localhost:8060`

Using `curl` POST/GET requests:

```
curl -X POST -F "text=I've lost two minutes." localhost:8060/service/
↪processQuantityText
```

Note that the model is designed and trained to work at *paragraph level*. It means that, for the moment, the expected input to the parser is a paragraph or a text segment of similar size, not a complete document. In case you have a long textual document, it is better either to exploit existing structures (e.g. XML/HTML elements) to segment it initially into paragraphs or sentences, or to apply an automatic paragraph/sentence segmentation, and then send separately to grobid-quantities the equivalent of a paragraph-size texts to be processed.

1.3 Training

To run the training:

- quantity model

```
cd PATH-TO-GROBID/grobid/grobid-quantities

./gradlew train_quantities
```

- unit model

```
./gradlew train_units
```

- value model

```
./gradlew train_values
```

the training and save the model produced in the latest iteration. 1000 iterations are largely enough.

CHAPTER 2

Annotation guidelines

2.1 Getting started

The first step of the annotation process is to generate training data from unlabeled documents based on the current models. The procedure is explained in details in *Training data* GROBID will create the training data corresponding to these documents in the right TEI format and with pre-annotations. The annotation work then consists of manually checking the produced annotations and adding the missing one. It is very important not to modify the text content in these generated files, not adding spaces or other characters, but only adding or moving XML tags.

When the training data has been manually corrected, move the file under the repository `resouces/dataset/quantities/corpus/` for retraining, or under `resouces/dataset/quantities/evaluation/` is the annotated data should be used for evalutation only. To see the different evaluation options, see GROBID documentation on training and evaluating.

2.2 Annotations

There are three types of measurements supported by grobid-quantities. Measurement corresponding to single value (or *atomic* value), to an interval (or range of values) or to a list. We do not distinguish conjunctive and disjunctive lists at the present time.

2.2.1 List of unit types

For the training annotation, the list of unit types (temperature, pressure, length, etc.) is controlled and based on SI definitions. This control is normally exhaustive and contains currently 50 types. The unit types are given in the file `` `src/main/java/org/grobid/core/utilities/UnitUtilities.java` ``. They are used to get the right transformation. The given names of the unit types has to be used when annotating measurement.

In the future, the list of units should however not be controlled and GROBID should support units never seen before.

For now it is admitted to annotate with `UNKNOWN` in case of doubt about the type. Examples:

- `m^2/kg` (specific surface area)

- `kD` (dissociation constant)
- `rad.m^-2`, in issue #51
- others in issue #53

2.2.2 Atomic values

We can distinguish two kinds of atomic values expressions:

Atomic values with units

Following TEI, the numeric value is identified with the element `<num>` and the unit with the element `<measure>` where the unit type is given by the attribute `@type`. The indicate of the unit name by the attribute `@unit` is optional: if present it might be used to augment the unit lexicon if not yet represented in the lexicon. A global `<measure>` element encodes the complete measurement (composed by the numeric value and the unit) associated with the measurement type given by the attribute `@type` `value`.

Example 1:

```
We monitored nutritional behaviour of amateur ski-mountaineering athletes during
↪<measure type="value"><num>4</num>
<measure type="TIME" unit="day">days</measure></measure> prior to a major competition␣
↪to compare it with official
recommendations and with the athletes' beliefs.
```

Example 2:

```
... <measure type="value"><num>20</num> <measure type="ENERGY" unit="MJ">MJ</measure>
↪</measure> (<measure
type="value"><num>4,800</num> <measure type="ENERGY" unit="kcal">kcal</measure></
↪measure>) for the shorter race route...
```

A percentage (and similar expression per mil and per ten thousand) has a unit type **Unit_Type.FRACTION**:

```
<measure type="value"><num>5</num> <measure type="FRACTION" unit="%">%</measure></
↪measure> of fat mass...
```

Atomic value without unit

They correspond to a count (implicit **Unit_Type.COUNT**). The numeric value is encoded with element `<num>` and the global `<measure>` element indicating the measurement type is added.

For example:

```
consists of <measure type="value"><num>two</num></measure> different race routes
```

The implicit **Unit_Type.COUNT** type will be infer by this particular encoding. Not that this encoding is only relevant to countable quantities.

2.2.3 Intervals

An interval introduces a range of values. We can distinguish two kinds of interval expressions:

1. Bounded value

Interval defined by a lower bound value and an upper bound value:

```
team races that can last from <measure type="interval"><num atLeast="4">4</num> to␣
↪more than <num atMost="12">12</num>
<measure type="TIME" unit="hour">h</measure></measure>
```

If the unit is mentioned twice, both units are annotated, example for `3 AU  rh <~ 5 AU`:

```
<measure type="interval"><num atLeast="3">3</num> <measure type="LENGTH" unit="AU">AU
↪</measure>  r h <num atMost="5">5</num> <measure type="LENGTH" unit="AU">AU</
↪measure></measure>
```

Note that an interval can be introduced by only one boundary value:

```
A rotor shaft according to any one of the preceding claims having a diameter of at␣
↪least <measure type="interval"><num
atLeast="1">1</num><measure type="LENGTH" unit="m">m</measure></measure>

[..]sky positions lie within a <measure type="interval"><num atMost="7">7</num>
↪<measure type="ANGLE" unit="°">°</measure>
</measure> radius of other planets[..]
```

2. Base and differential value

Take the example

```
4 women and 15 men, 30± 10 years, 176±7 cm, 70±9 kg, 15±5 % of fat mass, VO2max:␣
↪50±8 ml·kg1·min1 and 21 of race A
```

after two "counts", four measurements express intervals following this form.

```
<measure type="value"><num>4</num></measure> women and <measure type="value"><num>15</
↪num></measure> men,
```

Similarly as in the previous interval case, an attribute in element `<num>`, here `@type`, characterizes the *base* value and the *differential/range* value.

```
<measure type="interval"><num type="base">30</num> ± <num type="range">10</num>
↪<measure type="TIME" unit="year">years</measure></measure>,
<measure type="interval"><num type="base">176</num> ± <num type="range">7</num>
↪<measure type="LENGTH" unit="cm">cm</measure></measure>,
<measure type="interval"><num type="base">70</num> ± <num type="range">9</num>
↪<measure type="MASS" unit="kg">kg</measure></measure>,
<measure type="interval"><num type="base">15</num> ± <num type="range">5</num>
↪<measure type="FRACTION" unit="%">%</measure></measure> of fat mass
```

If the quantity is expressed only in term of range (without base) it can be implicitly assumed that the base=0, see example ± 10 years

```
<measure type="interval">± <num type="range">10</num><measure type="TIME" unit="year">
↪years</measure></measure>
```

If the interval has a base without a range, it's annotated with only the base (issue *#64 <https://github.com/kermitt2/grobid-quantities/issues/64>*):

```
a certain temperature interval1 around <measure type="interval"><num type="base">4 0</
↪num> <measure type="TEMPERATURE" unit="°C">°C</measure></measure>
```

Notes about intervals

- Interval markers such as `more than`, `less than`, and so on, are left outside the annotation when it's possible (see issue #35). Example:

```
more than <measure type="interval"> <num atLeast="2">2</num> </measure>
```

- An interval can be bounded with quantities expressed in different unit multiples (see issue #45). For the sentence `radii between 10 µm and 1 cm` the result will be:

```
grains with radii between <measure type="interval"><num atLeast="10">10</num>
↪<measure type="LENGTH" unit="µm">µm</measure> and <num atMost="1">1</num>
↪<measure type="LENGTH" unit="cm">cm</measure></measure>
```

- The `From ... to` markers are **not necessarily introducing an interval**, example:

```
the rate was reduced from <measure type="value"><num>1.87</num></measure> to
↪<measure type="value"><num>0.82</num></measure>
```

- When interval boundaries are given alphabetically, the <num> attribute must be converted in numbers:

```
between <measure type="interval"><num atLeast="1">one</num> and <num atMost="10">
↪ten</num></measure>
```

- Rational numbers are writter with a slash (/) (issue *#66 <https://github.com/kermitt2/grobid-quantities/issues/66>*)

```
at least <measure type="interval"><num atLeast="2/3">two-thirds</num></measure>␣
↪of wins for their favourite team
```

2.2.4 Lists

Lists introduce series of values. The unit can be expressed per value or for several values at the same time. A `<measure>` element encloses the whole list of values including their units:

```
<measure type="list"><measure type="ENERGY" unit="cm^-1">cm-1</measure>: <num>3440</
↪num>(br), <num>1662</num>,
<num>1632</num>, <num>1575</num>, <num>1536</num>, <num>1498</num>, <num>1411</num>,
↪<num>1370</num>, <num>1212</num>,
<num>1006</num>, <num>826</num>, <num>751</num></measure>

<measure type="list"><num>1.27</num> <measure type="LENGTH" unit="Å">Å</measure> for␣
↪1H5Y, <num>1.52</num>
<measure type="LENGTH" unit="Å">Å</measure> for 1KA9, and <num>1.69</num> <measure␣
↪type="LENGTH" unit="Å">Å</measure>
</measure> for 1THF
```

List can be disjunctive, conjunctive, or a combination. We do not distinguish the different kinds of list at the present time:

```
batches of <measure type="list"><num>three</num> or <num>four</num></measure>␣
↪observations

for flexural samples the size is <measure type="list"><num>100</num> <measure type=
↪"LENGTH" unit="mm">mm</measure>
 x <num>100</num> <measure type="LENGTH" unit="mm">mm</measure> x <num>400</num>
↪<measure type="LENGTH" unit="mm">mm
 </measure></measure>
```

If there are no intermediary values, it's an argument for deciding to annotate an element as a list, for example this ranked list (issue *#65 <https://github.com/kermitt2/grobid-quantities/issues/65>*):

```
for every lower position in the general classification the prize money was more or␣
↪less halved between
 the first <measure type="value"><num>seven</num></measure> ranked riders: from
↪<measure type="list">
 <measure type="CURRENCY" unit="euro">€</measure> <num>450,000</num> for the winner␣
↪over <measure type=
 "CURRENCY" unit="euro">€</measure> <num>200,000</num> for the runner-up to <measure␣
↪type="CURRENCY"
 unit="euro">€</measure> <num>100,000</num> for the third ranked rider, and so on to
 <measure type="CURRENCY" unit="euro">€</measure> <num>11,500</num></measure> for the␣
↪rider ranked in seventh place.
```

2.2.5 Additional items

Dates

Dates are time measurements, they are thus also encoded in the training data as a complement to the other _TIME_ expressions involving time units. In TEI P5, the dates are marked with a specific element `<date>` which can be contained in an element `<measure>`. The encoding is then straightforward for atomic values (with attribute `@when`), intervals (with attribute `@from-iso` and `@to-iso` in case on min-max intervals) and lists:

```
Comet C/2013 A1 (Siding Spring) will have a close encounter with Mars on <measure␣
↪type="value"><date when=
"2014-10-19">October 19, 2014</date></measure>.

The arrival time of these particles spans a <measure type="interval"><num type="range
↪">20</num>-<measure
type="TIME" unit="min">minute</measure> time interval centered at <date type="base"␣
↪when="2014-10-19T20:09">
October 19, 2014 at 20:09 TDB</date></measure>


Observations took place from <measure type="interval"><date from-iso="2014-10-19">
↪October 19, 2014</date> to
<date to-iso="2014-10-25">October 25, 2014</date></measure>.

the emergence of this sport in the <measure type="interval"><date from-iso="1980" to-
↪iso="1989">1980 s</date>
</measure>


Observations were performed on <measure type="list"><date when="2013-10-29">October␣
↪29, 2013</date>, on <date
```

(continues on next page)

(continued from previous page)

```
when="2014-01-21">Jan 21, 2014</date>, and on <date when="2014-03-11">March 11, 2014</
↪date></measure>.
```

Time tag (and difference with Date tag)

- if only the part of a date is expressed (for example the time of a day), but we can infer the date, a complete date is implicit and the context can make it being fully quantified.

For example `20:10 UTC` will be annotated:

```
<measure type="value"><date when="2014-10-19T20:10Z">20:10 UTC</date></measure>
```

With UTC inside the annotation which is important to know exactly the "time" measure.

- for a time expression not linked to a date, like the expression of an "hour", it's appropriate to annotate with the tag `<time>`, to distinguish from the `<date>` case (see issue #48):

```
To sleep well, relax everyday at <measure type="value"><time when="21:00:00">21:00</
↪time></measure>
```

```
It consists of infusing the [...] drugs in the following order: leucovorin between
↪<measure type="interval"><time from-iso="10:30:00">10 h 30</time> and <time to-iso=
↪"21:20:00">21 h 20</time></measure> [...]
```

Special cases

Frozen quantity expressions like *decade* **or** *Room temperature*

- **Room temperature** (Raumtemperatur, température ambiante, ...) is used very frequently in chemistry and related fields. It can be considered as 20 °C (293 Kelvin), although not defined in a standard manner (https://de.wikipedia.org/wiki/Raumtemperatur).

  ```
  <measure type="value"><measure type="TEMPERATURE">Raumtemperatur</measure></
  ↪measure>
  ```

- **Decade** (issue #52)

  ```
  over  <measure type="interval"><num atLeast="2">two</num> <measure type="TIME" >
  ↪decades</measure></measure>
  ```

2.2.6 Miscellaneous / Examples

Plus (+) and minus (-) signs

The + and - signs must be put **inside** the `<num>` tag. Examples:

```
a recent study [...] showed that cycling efficiency was lower (<measure type="value">
↪<num>11</num><measure type="FRACTION" unit="%">%</measure></measure>) and energy␣
↪cost of running was greater (<measure type="value"><num>+11</num><measure type=
↪"FRACTION" unit="%">%</measure></measure>) in the master compared with young␣
↪triathletes
```

Units without values

Case where it's not annotated: When we refer to the units as such, to express something about the units, we are not using the units to quantify something with a value:

```
and r H are the geocentric and heliocentric distances in cm and AU, respectively, and␣
↪F comet and F
```

Like here for the units: `cm` and `AU`.

Case where it's annotated: We could have units expressed without values, when the value is implicit:

```
that can extend <measure type="interval"><measure type="LENGTH" unit="mm">millimeters
↪</measure></measure> or even <measure type="interval"><measure type="LENGTH" unit=
↪"cm">centimeters</measure></measure> from the cell body
```

here the value of millimeters and centimeters is unspecified (e.g. equivalent to `several`), but we have a quantity and more precisely an interval. See issue #31

Unprecise quantifiers

When used with units, quantifers like `few`, `several`, `a couple`, `a large amount of` is annotated, and whatever quantifies even imprecisely :

```
the reference solution becomes distinct from the ballistic solution only a <measure␣
↪type="value"><num>couple</num> of <measure type="TIME" unit="week">weeks</measure></
↪measure> before the encounter.
```

Determiners are left outside (*a <measure type="value"><num>couple</num> of <measure type="TIME" unit="week">weeks</measure></measure>*). See issue #34

X-fold

Quantifiers like `two-fold`, `sevenfold` meaning "two times/part", "seven times/part" are annotated, to capture the full expression of quantity including this notion of "part":

```
allowing a <measure type="value"><num>sevenfold</num></measure> compaction of the␣
↪length

LUCA-HisF displays a clear <measure type="value"><num>two-fold</num></measure>␣
↪symmetry
```

Constants

Precise number (for example `c` , the speed of light in vacuum) and imprecise numbers (for example π which has an infinite number of decimals) are annotated. See issue #37

Example:

```
`decelerating from <num>5</num><measure type="VELOCITY" unit="% c">% c</measure>`
```

Exponents for powers of ten

Exponents notation might be lost in documents, for example 10 power -6 in pdf becomes `10 6`. The correct exponents are written in the attribute when there is one, 10 power -6 will be written `10^-6`. Example in interval:

```
<measure type="interval"><num atMost="10^-6">10 6</num></measure>
```

See issue #38

Numbers which seems to be only tags but are in fact quantifying

For example expressions like *at day 21* or *between day 56 and day 91*, which are really quantifying and for which range queries can be expressed.

OCR errors

OCR errors are annotated as if they were the correct sequences, since they are realistic noise. For example:

```
20 aC -> <measure type="value"><num>20</num> <measure type="TEMPERATURE" unit="°C">°C
↪</measure></measure>
2.5 • -> <measure type="value"><num>2.5</num><measure type="ANGLE" unit="°">°</
↪measure></measure>
```

2.2.7 Out of scope

Only **expressions of quantities** are annotated, which can use numbers or alphabetical words.

Some numbers are also used for other stuff like markers, call-out, section number, identifiers, index, reference expressions, formula parameters, ill-encoded characters, etc. and all these cases are out of scope. See issue #36

Some sequences not annotated (not commented)

Markers, call-out, section number, numerical bullet points, identifiers, index, reference expressions, formula parameters

Examples:

Reference markers:

```
lower than those derived by Vaubaillon et al. (2014) and Moorhead et al. (2014)␣
↪computing the corresponding impact probabilities (Milani et al. 2005)
```

Figure/table titles, and other numbers who don't quantify anything:

```
Figure 1 shows the residuals of C/2013 A1's observations
[Figure 1 about here.]
Table 1 contains the orbital elements of the computed solution.
our new orbit solution (JPL solution 46)
```

Inline formulas, like:

```
a minimum point of v 2 = |v| 2 under the constraint that the particle reaches Mars, i.
↪e., (ξ, ζ)(r, β, v) = (0, 0).
```

Some sequences to be commented out

Ill-encoded characters

Examples:

Sequence that would be usually annotated but contain encoding problems / characters in the free unicode range, like:

```
```

2.2.8 Quantified object

Currently work in progress The quantified object (or substance) is the object for which the measurement is expressed. For example *A mixture of 10kg of silicon nitride powder*. The object is the *silicon nitride powder* which is attached to the measurement of 10 with unit Kg.

Cf. issue #19

The quantified object model currently rely on a simplistic approach that involve the dependency parsing of the sentence and the identification of the head in the phrase with some heuristic.

In this section we discuss the guidelines for annotating training data used for a CRF model to replace the current implementation.

In the training data are identified the measurement (and their type) and the quantified object.

For example:

```
<p>A mixture of 10kg of silicon nitride powder.</p>
```

can be annotated as:

```
<p>A mixtured of <measure type="value" ptr="#1235324324321">10kg</measure> of
↪<quantifiedObject id="1235324324321">silicon nitride powder</quantified Object>.</p>
```

The quantified object is identified by its ID and linked to the measure via the attribute *ptr="#ID"*.

NOTE This implementation allows the linking of objects directly attached on the left or right of the measurement, for the time being far entities are not supported.

How to annotate?

Annotating the quantifiedObject is a complicated task, because it requires a clear definition to avoid miunderstanding. Firstly the question each annotator should ask is "What is being measured?".

2.2.9 Case not yet supported

The following cases are not annotated at this stage. **The sentence when these cases occur should be put in comments** for the moment.

Sigma estimation

```
We selected the A 1 uncertainty so that its range would span from 0 au/d 2 to twice␣
↪the nominal value at 3&#x3C3;.
```

Intervals embedded in intervals

```
[..]only Mars is near enough that the orbital motion can extend a single viewing␣
↪window from 45 days to as much as 60 to 90 days.

For the wide scenario the uncertainty goes from 45 min down to 1-2 min.
```

Note: one possibility would be to only mark the external boundaries of the interval.

```
[..]only Mars is near enough that the orbital motion can extend a single viewing␣
↪window from <measure type="interval">
<num atLeast="45">45</num><measure type="TIME" unit="day">days</measure> to as much␣
↪as 60 to <num atMost="90">90</num>
<measure type="TIME" unit="day">days</measure></measure>.

For the wide scenario the uncertainty goes from <measure type="interval"><num atLeast=
↪"45">45</num>
<measure type="TIME" unit="days">min</measure> down to 1-<num atMost="2">2</num>
↪<measure type="TIME" unit="min">min</measure></measure>.
```

List of intervals

```
No significant difference in running and total times was observed between the age␣
↪groups 25 to 34 and 35 to 44 years
```

Atomic value expressed with different values and units

```
The current Hawaii Ironman triathlon record is 8:54:202 for females
```

```
[a] male athlete was able to finish [an] ultra-marathon in a time of 19 h 44 min
```

Unit embedded in numerical value

For example `92°.5` wich would require to embed `<measure>` in `<num>` (issue #49). Or also `126 withdrawals out of 162 riders`.

Discontinuous cases

Interval quantities whith base and range multiplied as a whole by a power of ten (see issue #42):

```
A1 (Siding Spring) will pass Mars with a close approach distance of 1.35 ± 0.05 × 10␣
↪5 km
```

or like:

```
The gas production rates, Q(CO 2 ) = (3.52> ± 0.03) × 10 26 molecules s 1
```

Other cases

```
This process takes place between t = 30[ 12 h] and t = 140[ 12 h]
```

2.2.10 Various examples

XML examples (ended with the name of the file between brackets)

```
Note that only <measure type="value"><num>47 out of the 76</num></measure> patients␣
↪experienced a first accident.[hal-00643787.training.tei.xml]
```

(continues on next page)

(continued from previous page)

```
Branson Sonifier W-250D, <measure type="value"><num>2 x 2</num> <measure type="TIME"
↪unit="min">min</measure></measure> in <measure type="value"><num>15</num> <measure
↪type="TIME" unit="s">sec</measure></measure> intervals. [hal-00924047.training.tei.
↪xml]

ranging from <measure type="interval"><num atLeast="1.14">1.14</num> <measure type=
↪"LENGTH" unit="Å">Å</measure> to <num atMost="1.43">1.43</num> <measure type="LENGTH
↪" unit="Å">Å</measure></measure>.[hal-00924047.training.tei.xml]

the emission maxima shifted from <measure type="value"><num>345</num> <measure type=
↪"LENGTH" unit="nm">nm</measure></measure> to <measure type="value"><num>325</num>
↪<measure type="LENGTH" unit="nm">nm</measure></measure> [hal-00924047.training.tei.
↪xml]

[...] (<measure type="list"><num>72</num> for compressive test and <num>72</num></
↪measure> for flexural test) [hal-00962359]

Patients with SS have a <measure type="interval"><num atLeast="20">20</num>-<num
↪atMost="40">40 fold</num></measure> increased risk of developing lymphoma [hal-
↪00987664]
```

CHAPTER 3

Training data

As the rest of GROBID, the training data is encoded following the TEI P5. See *Annotation guidelines* for detailed explanations and examples.

3.1 Generation of training data

Training data generation works the same as in GROBID, with executable name `createTrainingQuantities`, for example:

```
java -jar build/libs/grobid-quantities-{version}-onejar.jar trainingGeneration -dIn ~/
↪grobid/grobid-quantities/src/test/resources/ -dOut ~/test/
```

Help can be invoked with

```
java -jar build/libs/grobid-quantities-{version}-onejar.jar trainingGeneration --help
```

The input directory can contain PDF (.pdf, scientific articles only), XML/TEI (.xml or .tei, for patents and scientific articles) and text files (.txt).