FRIEDRICH-ALEXANDER-UNIVERSITÄT
ERLANGEN-NÜRNBERG

PROFESSUR FÜR WISSENSREPRÄSENTATION UND
-VERARBEITUNG

Meaning Extraction and Semantic Services in STEM-Documents

A case study on Quantity Expressions and Units

Master Thesis in Computer Science

Author:
Ulrich Rabenstein

Supervisor:
Prof. Dr. Michael Kohlhase

June 7, 2017

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 7. Juni 2017

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The quantification of the environment is an essential part of human life and it is thus not surprising that the first human measurements were done at least 30,000 years ago [Hau01, Page 9ff]. This was detected in 1936 by Karel Absolon, a Czech paleontologist, who found radius bones with a regular pattern of notches. He interpreted them as counting signs – possibly to measure the amount of prey. It took several thousand years to develop a more complex system of measurement based on comparison to a reference. In former times, length, for instance, was measured in comparison to parts of the human body, like in the case of the units feet and cubit. Without a fixed value for these terms, we can regard the information that, say, a house has a height of 10 feet only as a loose estimate. The exact value depends on the length of the compared foot which is obviously impractical as a common system. This problem can be solved by defining exact values for the terms. Like this, we get fixed *units* – like the modern unit "foot" as a measure for length. Now we can use *quantity expressions*, say again "10 feet", to describe exact distances. However different units with different values arose in different societies. Today there are two predominant systems of measurement, the imperial system and the metric system. The latter is nowadays officially accepted in nearly all countries, but the former is still in use in many countries that were part of the old British Empire. The mixture of both systems already caused expensive errors like the crash of the Mars Climate Orbiter in 1999 [Boa99]. It happened because a part of the program calculated the impulse of the spacecraft in imperial units, namely in pound-seconds, while a different module expected the impulse in newton-seconds – a metric unit – as an input. The resulting miscalculation lead to a wrong position of the spacecraft which finally caused the crash.

Efforts for a uniform and modern system of measurement resulted in the International System of Units (abbreviated as SI) which was resolved in 1960 [fWM06]. It is based on seven basic physical properties – the *dimensions* – and seven corresponding units: Length is measured in meter, mass in kilogram, time in seconds, electric current in ampere, temperature in kelvin, the amount of substance in mole and luminous intensity in candela (see also Table 1). The value of the basic units is derived mostly from natural constants. For instance, the second is defined as

> "the duration of 9 192 631 770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium 133 atom." [fWM06, Page 113]

and the meter as

> "the length of the path travelled by light in vacuum during a time interval of 1/299 792 458 of a second." [fWM06, Page 112].

SI base unit	Physical Dimension
meter (m)	length
kilogram (kg)	mass
second (s)	time, duration
ampere (A)	electric current
kelvin (K)	temperature
mole (mol)	amount of substance
candela (cd)	luminous intensity

Table 1: The SI Base Units, their Symbols and their corresponding Physical Dimensions. This table corresponds to Table 1 in [fWM06, Page 116].

Factor	Name (Symbol)	Factor	Name (Symbol)
10^{1}	deca (da)	10^{-1}	deci (d)
10^{2}	hecto (h)	10^{-2}	centi (c)
10^{3}	kilo (k)	10^{-3}	milli (m)
10^{6}	mega (M)	10^{-6}	mirco (μ)
10^{9}	giga (G)	10^{-9}	nano (n)
10^{12}	tera (T)	10^{-12}	pico (p)
10^{15}	peta (P)	10^{-15}	femto (f)
10^{18}	exa (E)	10^{-18}	atto (a)
10^{21}	zetta (Z)	10^{-21}	zepto (z)
10^{24}	yotta (Y)	10^{-24}	yocto (y)

Table 2: SI Prefixes – compare Table 5 in [fWM06, Page 121].

For convenience, the system also contains many units that can be derived from the basic units without any additional numerical factor. For instance watt, a unit for power, is defined in terms of basic units as $1\ \mathrm{W} = 1\ \mathrm{kg} \cdot \mathrm{m}^2 \cdot \mathrm{s}^{-3}$ and ohm, a unit for electrical resistance, is defined as $1\ \Omega = 1\ \mathrm{W} \cdot \mathrm{A}^{-2} = 1\ \mathrm{kg} \cdot \mathrm{m}^2 \cdot \mathrm{s}^{-3} \cdot \mathrm{A}^{-2}$. Tables 2 to 4 in [fWM06, Page 117ff] list derived units and their definition in SI base units. We omit a separate table here. Decimal fractions and multiples of SI units can be formed with a fixed set of SI prefixes, which are shown in Table 2 and range from 10^{-24} to 10^{24}. The kilogram is the only SI-unit which already contains a prefix in its normal form as its value is defined by the mass of the international kilogram prototype.

Nevertheless, not only imperial units – like "miles", "square mile feet" and "degree Fahrenheit" – are still widely used, but also very specialized units in their certain fields – such as "solar masses" in astronomy, "atomic mass units" in particle physics and "electron volt" in high energy physics. This is because they are coupled to the phenomena under discussion and thus more intuitive. Compare, for instance, the information that a planet has a mass of approximately "$9.9455 \cdot 10^{29}\ \mathrm{kg}$" to its equivalent of "0.5 solar

masses". However, for people who are not familiar with these units, their occurrence can also complicate the understanding of texts. As an example a recipe in fluid ounces and quarts requires some calculations to translate it to, say, milliliters for a person who is used to metric units. Inaccuracies and errors in such calculations not only cause unpleasant-tasting dishes, but, as we have already seen, can cause dramatic errors like the crash of the Mars Orbiter in more complicated settings. Computer programs help to prevent such problems and many simple unit conversion tools and web-services are already widely available. However they not only require the user to leave his document in order to open the conversion service, but also to manually enter the data which is another source of errors. The automatic conversion of whole documents from one system of measurement to another can help users to prevent such errors and to stay more focused, because they do not have to switch to a conversion service. But it is a much more difficult task, which we will, among others, investigate in this thesis.

1.2 Related Work

The aspect of quantity expressions in scientific documents has already been investigated from different perspectives and we give a brief overview on them in this section.

Stiv Sherko has already worked on the detection of quantity expressions in his bachelor thesis [She15]. He uses the arXMLiv corpus [SKG+10] which is a transformation of the e-print archive *arXiv* to XML. His thesis includes a categorization of typesetting possibilities for quantity expressions, which, however, is limited to only three cases and thus omits many cases, both from the typesetting point of view and from the unit point of view. For instance, quantity expressions like "30°" do not occur in his thesis at all. He also presents a rule-based system for the extraction of semantics of quantity expressions which uses context free grammars. Of course this system is based on his categorization and therefore has the same strong limitations in scope and accuracy.

In his bachelor thesis, Tom Wiesing has built a semantic search engine for quantity expression [Wie15] which finds units across different representations, say, it finds "$25\,\mathrm{m/s}$" when searching for "$90\,\mathrm{km/h}$". The search engine relies on getting input data from an external application. Units are encoded in and can be converted via the extensible framework for knowledge representation MMT [RK13]. His work is an interesting prototype, but has some important limitations: For instance, one cannot search for units in a given context. Say, we can search for "$25\ \mathrm{m}^2$", but not for a formula like "$x^2 = 25\ \mathrm{m}^2$". In addition, his formalization treats prefixed units – like "kilometer" – as separate independent entities and therefore has some overhead compared to a generic solution that allows to construct prefixed units from base units – e.g. "meter" – and a set of prefixes. In the final version of Wiesing's thesis, Sherko's application is used as a source for input data. The range of the search engine is thus strongly limited as it inherits Sherko's limitations. Furthermore the front-end of the application is a rather ad hoc solution with a low level of user friendliness.

The authors of [CGL11], Mihai Cîrlănaru, Deyan Ginev and Christoph Lange, consider the aspect of semantic publishing – especially for quantities and units. They review existing and propose new methods for directly including semantics in LaTeX, which involves additional packages like "SIunits" [WH] and "sTeX" [Koh08]. Additionally they present a system for the automatic conversion of units, which, however, only works on semantically published documents.

The World Wide Web Consortium (W3C) published a note on the correct representation of units in the mathematical markup language MathML [DH03], which permits to include mathematical content in HTML5 documents. The note states that "unit symbols are written in roman (upright) type" and explains how to ensure this in MathML. Additionally, it proposes a method to identify unit symbols with their corresponding definitions and discusses the conversion of units in MathML.

In [SD08], James Davenport and Jonathan Stratford present the challenges of handling units and of creating a unit converter using the semantic markup language OpenMath [SOD+04]. They describe their solutions for the encoding of prefixes as well as the problem of converting between units without fixed conversion factor like "months" and "days". The authors conclude by stating that it is possible to build a unit converter in OpenMath which differs from existing ones by its extensibility.

Marcus Foster observed the ambiguity of the SI notation and proposed changes to make it unambiguous [Fos09]. He addresses that expressions like "lm", "hA" and "mN" have multiple possible meanings. The first might denote both "lumen" and "liter-meter". For the latter two expressions, the first character can either stand for a prefix or a unit resulting in the possible meanings "hour-ampere" and "hectoampere" for the second expressions as well as "millinewton" and "meter-newton" for the last one. His suggested changes include the mandatory usage of explicit multiplication for compound units, say, "$\mathrm{m} \cdot \mathrm{N}$" for "meter-newton" to make the SI system unambiguous and thus guarantee correct parsing.

1.3 Contribution of the Thesis

In this thesis, we describe a framework for the detection of quantity expressions in documents from the fields of science, technology, engineering and mathematics (STEM). These documents have originally been typeset using LaTeX and were then converted to HTML5 with mathematical markup in the form of presentation and content MathML. Additionally, we present *semantic services* – services that are based on the meaning of the underlying text – upon the detected quantity expressions.

For that, we first review the phenomenology of quantity expressions and units in STEM documents and create a classification for them. Next we describe a rule-based implementation which detects that certain sequences of characters are quantity expressions, infers their meaning and represents this in a standardized format; here content MathML. We thus refer to such challenges as *meaning extraction*. We then use the ex-

tracted content representations to offer three semantic services. The first is a service for the automatic conversion of quantity expression. This allows us to simply right click on an expression in the document and to convert it to a unit of our choice, say "3 hours" to "10800 seconds". We can also convert all quantity expressions in a given document to their corresponding SI base units, which is a prototypical implementation of a global conversion. Additionally, we improve the scope of Tom Wiesing's semantic search engine for quantity expressions by converting the extracted semantic information to suitable input data for his system. Like this, we can search for formulae containing, say, "42 kilometers", and our results also include equivalent expressions like "26 miles". The last semantic service is an enhancement for screen reading programs for blind users, which allows the programs to read the meaning of a quantity expression instead of or in addition to its presentation, say "1500 Volt" instead of "1500 V".

1.4 Structure of the Thesis

We start by reviewing the phenomenology of quantity expressions in documents that were typeset using LaTeX (Section 2). In Section 3, we briefly explain technologies and applications which are necessary for this thesis. Using this terminology, we discuss the research problem in Section 4 and then describe the software architecture, the rule-based implementation as well as its evaluation (Section 5). We explain the semantic services in Section 6 and discuss future work in Section 7 before we conclude the thesis (Section 8).

2 Phenomenology

There are various different ways of expressing the same quantity expression, even when it consists of the same units. Compare, for instance, "5 m", "5 meter" and "five meter", which are all equivalent ways of stating the exactly same information. The documents, we are working with, were originally written in LaTeX which also offers a wide variety of possibilities to typeset the same expression. Thus the goal of this section is to present concrete examples of quantity expressions and units, which the author observed during his work on this thesis, and to introduce a categorization for them (Section 2.1). We will also refer to these examples as running examples throughout the thesis. In Section 2.2, we discuss the observations and derive requirements for the implementation. We restrict the scope of this thesis to a subset of the observed quantity expressions at the end of this section.

Nr.	Rendered Result	LaTeX-source	Reference
1	$5GHz$	`$5GHz$`	[Ser98]
2	5GHz	`$5\text{GHz}$`	
3	$10^{-10}s$	`$10^{-10}s$`	[Ser98]
4	6μm	`$6 \mu \text{m}$`	[RSM+98]
5	0.45 eV	0.45 eV	[TTGV98]
6	200 μm	200 `$\mu$`m	[VBR98]
7	25MΩ	`$25$M$\Omega$`	[VBR98]
8	$0.6M_{\odot}$	`$0.6M_\odot$`	[ZHM+98]
9	0.6 $M_{\odot}$	0.6 `$M_\odot$`	[ZHM+98]
10	10^{-4} $M_{\odot}$	10`$^{-4}$` M`$_\odot$`	[BBGHK92]

Table 3: Examples of the Category of Simple Multiplicative Quantity Expressions.

Nr.	Rendered Result	LaTeX-source	Reference
1	$10^{20}/s$	`$10^{20}/s$`	[Iwa98]
2	10^{17}/s	`$10^{17}$/s`	[Iwa98]
3	20 fm/c	20 fm/c	[LK98]

Table 4: Examples of the Category of Simple Divisive Quantity Expressions.

2.1 Categorization of Quantity Expressions and Units

We distinguish quantity expressions from a syntactic and a semantic point of view and for instance have a category containing relatively plain expressions with units without superscripts (i.e. "$1\ \mathrm{m}$"; syntactically) and a category containing range expressions for arbitrary units (i.e. "$1 - 2\ \mathrm{kg}$"; semantically). This intentionally does not lead to a strictly disjoint categorization, as the examples demonstrate. For every example, we show the presentational part of the quantity expressions as well as its LaTeX source and a reference to the document it occurred in. It is sufficient to present short LaTeX snippets here, since the same observations apply to the HTML code which is created from the LaTeX source. The most basic category is the just mentioned one for *simple multiplicative quantity expressions* (Table 3). We regard a quantity expression as simple, if it contains any kind of numeric expression followed by one or more unit symbols in a multiplicative way (i.e. "3 Nm"). This excludes unit symbols, which are in superscript, like "30°", and we also exclude written-out unit symbols (i.e. meter) and textual numbers for this class.

Instead there is a separate category for textual unit symbols and quantity expressions with textual numbers (Table 7). In addition to the first category, we introduce one for *simple divisive quantity expressions* (Table 4), which differs only by the fact that units may also occur in a divisive way (i.e. "4 m/s"). We extend the simple multiplica-

Nr.	Rendered Result	LaTeX-source	Reference
1	$1.0 \cdot 10^{17}$W/cm^2	`$1.0\cdot10^{17}\text{W/cm}^{2}$`	[RSM+98]
2	1.0Wcm$^{-2}\mu$m^2	`$1.0\mbox{Wcm}^{-2}` `\mu\text{m}^{2}$`	[RSM+98]
3	4.8×10^9 cm^{-2}	`$4.8\times 10^{9}$` cm`${}^{-2}$`	[YLS+98]
4	0.5 eV/Å^3	`$0.5$` eV/Å`${}^{3}$`	[TTGV98]
5	100 km s^{-1} Mpc^{-1}	`$100$` km s`${}^{-1}$` Mpc`${}^{-1}$`	[Iwa98]
6	3.72 $\times$ 10 ^{10}cm^{-2}	3.72 `$\times$` 10`${}^{10}$`cm`${}^{-2}$`	[YLS+98]
7	$F_{-7} \times 10^{-7}$ ergsec^{-1}cm^{-2}	`$F_{-7} \times 10^{-7}` `{\rm ergsec}^{-1}{\rm cm}^{-2}$`	[PS92]
8	$(0.4$ GeV$)^2$	`$(0.4 \;\rm GeV)^2$`	[AF92]
9	10^{-34}gm/cm^3	`$10^{-34}{\rm gm/cm^{3}}$`	[HK92]

Table 5: Examples of the Category of Complex Quantity Expressions.

Nr.	Rendered Result	LaTeX-source	Reference
1	30°	`$30^\circ$`	[RSM+98]
2	$00^h49^m37^s.71$	`$00^{h}49^{m}37^{s}.71$`	[ZHM+98]
3	$-29^\circ50'58''.7$	`$-29^{\circ}50^{\prime}` `58^{\prime\prime}.7$`	[ZHM+98]
4	$0.025^\circ C$	`$0.025^{\circ}C$`	[TMM98]
5	2 ° C	`$2\;^{\circ}$C`	[FGMV92]
6	$10''$	`$10^{\prime\prime}$`	[HK92]

Table 6: Examples of the Category of Quantity Expressions with Units in Superscript.

Nr.	Rendered Result	LaTeX-source	Reference
1	five seconds	five seconds	
2	1.27 square degrees	1.27 square degrees	[ZHM+98]
3	one GHz	one GHz	[Ros92]

Table 7: Examples of the Category of Textual Quantity Expressions.

Nr.	Rendered Result	LaTeX-source	Reference
1	$0.01 - 0.1\mu m^2$	`$0.01-0.1\mu m^{2}$`	[Ser98]
2	0.53 ± 0.01 eV	`$0.53 \pm 0.01$` eV	[TTGV98]
3	3.7–19 GeV	3.7–19 GeV	[MS98]
4	20 to 10,000 kilometers	20 to 10,000 kilometers	[PHL92]

Table 8: Examples of the Category of Range Expressions.

Nr.	Rendered Result	LaTeX-source	Reference
1	$23\ \mu\text{m} \times 23\ \mu\text{m}$	`$23\;\mu\text{m}\times 23\;\mu\text{m}$`	[RSM+98]

Table 9: Examples of the Category of Quantity Products.

tive and divisive quantity expressions to the category for *complex quantity expressions* (Table 5) which contains expressions from the former categories, but allows additional superscripts for units (i.e. "5 m^2" or "5 m/s^2"). This category subsumes the first two.

Additionally, we have a category for quantity expressions where unit symbols are part of the superscript – as in "30°". Although written in their own manner, these expressions are, from a semantic point of view, closely related to the simple multiplicative units.

Furthermore, there are extra classes for *range expressions* (Table 8) and *unit products* (Table 9) because they include the previous expression, but contain additional semantic information. For instance "23 μm × 23 μm" describes not only an area of size "529 μm^2", but also contains the information that the area is quadratic and has an edge length of "23 μm". Additional information is necessary to handle quantity expressions involving constants (Table 10). For instance, we need to know that

> "Ω_a is the ratio of the axion energy density to the critical density in the Universe" [Iwa98]

and that "h" is the Hubble constant to understand the meaning of Example 2 of this table. The examples in Table 11 do not describe quantity expressions, but depict that certain formulae are written in these units. Hence these terms also form their own class. Figure 1 summarizes the relations between the categories in a taxonomy.

Nr.	Rendered Result	LaTeX-source	Reference
1	$3h^{-1}$ Gpc	`$3 h^{-1}$ Gpc`	[Ros92]
2	$10^{-12} M_\odot \Omega_a h^2$	`$10^{-12}M_{\odot}\Omega_{a}h^{2}$`	[Iwa98]

Table 10: Examples of the Category of Quantity Expressions involving Constants.

Nr.	Rendered Result	LaTeX-source	Reference
1	[...] $(\mathrm{MeV/fm^3})$	`(\rm MeV/fm^{3})`	[LK98]
2	in GeV ^{2}	in GeV `$^2$`	[AF92]

Table 11: Examples of Single Units.

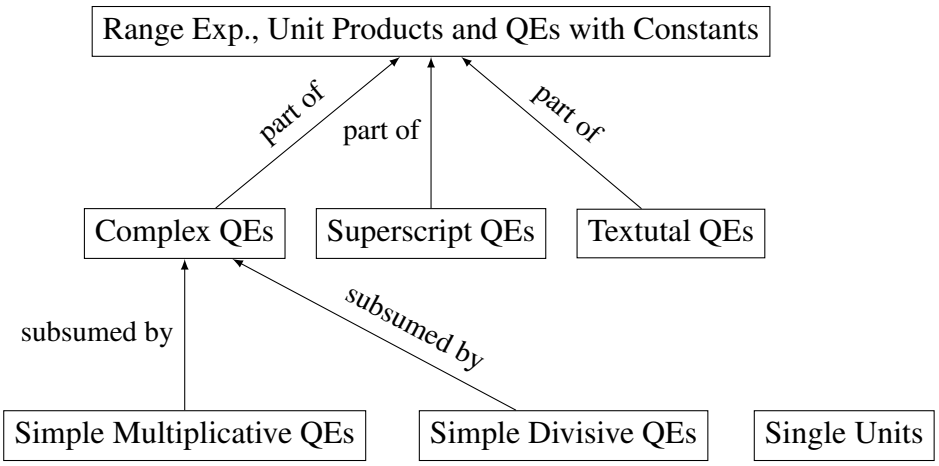


Figure 1: The Taxonomy of the Categorization of Quantity Expressions (QE).

2.2 Discussion

In this section we discuss the perception of the examples from the previous section. At first, we look at the presentational part and then at the corresponding LaTeX source.

Marcus Foster pointed out that unit strings can be ambiguous (compare related work in Section 1.2) and we agree with his observation. For instance "GHz" is very likely to stand for "gigahertz", but could also denote "gauß[1] · hertz". Similarly, "Pa" has two possible meanings – "petayear" and "pascal[2]". Additionally, it is not always clear to which part of the expression exponents refer to. In Example 4 from Table 5, "0.5 eV/Å^3", the exponent might refer only to "Ångström"[3], but also to the whole expression. Hence we have the two possible meanings "0.5 eV/(Å^3)" and "0.5 (eV/Å)^3". Given the context of the paper, we see that the former was meant in this case.

In the rendered result, the changes between text and math mode do not complicate the understanding of the expressions for humans. But when looking at the LaTeX source, we observe that this adds a lot of noise to the data. Authors of scientific papers tend to misuse text mode to ensure an upright font for units, instead of using the "`\rm`" command for this purpose. This leads to somewhat surprising encodings with frequent

[1] Gauß is a unit of magnetic induction, commonly abbreviated as "G".
[2] Pascal is a unit of pressure, commonly abbreviated as "Pa".
[3] Ångström is a unit of length (1 Å $= 10^{-10}$ m).

changes of math and text mode like

"100 km s^{-1} Mpc ${}^{-1}$" (from [Iwa98]),

"10^{-4}M$_\odot$" (from [BBGHK92])

and especially

"3.72 $\times$ 10 ^{10} cm ${}^{-2}$" (from [BBGHK92]).

Example 1 from Table 5 is also a remarkable case, because here, we see a difference between the semantics of the LaTeX source and the rendered result. From its source, we can assume the meaning "$(\mathrm{W}/\mathrm{cm})^2$", while the presentational part seems more consistent with "$\mathrm{W}/(\mathrm{cm}^2)$". The latter is correct here, because the expression describes the intensity of a laser beam.

2.3 Restrictions for this Thesis

The detection of all kinds of quantity expressions and units in STEM documents is a task which exceeds the scope of this thesis and the author thus has to restrict his attention to only a part of this problem. We further investigate the detection of quantity expressions from the categories in the Tables 3 to 6. We omit textual quantity expressions and single units, because they hardly occur in the documents and because they need to be handled differently. We also omit unit products and range expressions, due the additional complexity of handling their semantics, as well expressions involving constants which would require to also detect or infer the definition of the constants.

3 Basics and Tool Environment

In this section, we introduce the technologies necessary for the thesis. They include the markup language MathML (Section 3.1), LaTeXML – the converter from LaTeX to XML – and the arXMLiv corpus (Section 3.2), the annotation tool KAT (Section 3.3) and the unit converter from the community python package Astropy (Section 3.4). We assume basic knowledge of LaTeX, HTML and XML.

3.1 MathML

MathML [MCI14] is a mathematical markup language which can be used to write formulae in HTML. There are two different strains of it – presentation MathML and content MathML. Presentation MathML describes the – possibly ambiguous – layout of a formulae. For instance, $g(x + y)$ might either encode the application of a function g

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <semantics>
    <mrow> ⟨⟨ Presentation MathML ⟩⟩ </mrow>
    <annotation−xml> ⟨⟨ Content MathML ⟩⟩ </annotation−xml>
    <annotation encoding="application/x−tex">
      ⟨⟨ LaTeX code ⟩⟩
    </annotation>
  </semantics>
</math>
```

Figure 2: A General Frame for a MathML Expression.

to the argument $x + y$ or a multiplication of g and $x + y$. Content MathML resolves this ambiguities by explicitly denoting the semantics of an expression. Figure 2 shows the general frame for an expression in MathML which always starts with a math tag and the corresponding namespace. The semantics child is a container whose first child is usually an expression in presentation MathML, followed by annotations for this element. An annotation can be in XML, for instance content MathML, or it can be any sequence of characters, for example LaTeX code. Annotations can also be omitted. In this case, we can also omit the semantics child and write presentation MathML directly as a child of the math node. We use angle brackets to denote meta variables in XML like ⟨⟨ Presentation MathML ⟩⟩ , ⟨⟨ Content MathML ⟩⟩ and ⟨⟨ LaTeX ⟩⟩ in Figure 2.

We briefly introduce presentation and content MathML and point to [MCI14] for details.

3.1.1 Presentation MathML

Figure 3 shows the presentation MathML encoding of the formula "$3.0 \cdot 10^{-17}\mu\mathrm{m}$" as an example. There are two kinds of tokens for presentation MathML – element tokens and layout schemata. Element tokens include numbers (mn), operators (mo), identifiers (mi) and text (mtext), while layout schemata include horizontal grouping (mrow), superscript (msup) and subscript (msub). Tags can be modified by attributes, such as the `mathvariant` attribute. Its value `normal` ensures an upright font for the correct presentation of a unit in MathML – for details compare the W3C note about units in MathML [DH03].

3.1.2 Content MathML

The corresponding content MathML expression is displayed in Figure 4. We immediately observe that content MathML is a prefix-style notation, compared to the infix-style of presentation MathML. There is no analogous tag for mo, instead content MathML supports a large amount of special operator tags such as times, power and minus. We can

```
<mrow>
  <mn>3.0</mn>
  <mo>⋅</mo>
  <msup>
    <mn>10</mn>
    <mrow>
      <mo>−</mo>
      <mn>17</mn>
    </mrow>
  </msup>
  <mi mathvariant="normal">μ</mi>
  <mi mathvariant="normal">m</mi>
</mrow>
```

Figure 3: A Presentation MathML Expression for "$3.0 \cdot 10^{-17}$μm".

express function application using the apply tag. cn corresponds to the presentational tag mn, but content MathML distinguishes between csymbol and ci for identifiers. csymbol is used for concrete mathematical terms and the `cd` attribute can be used to point to the definition of this term. Local identifiers are written using ci.

3.2 LaTeXML and the arXMLiv Corpus

LaTeXML [Mil16] is a LaTeX to XML converter that imitates TeX, but generates an XML document instead of a PDF document. It also contains a postprocessor which can further convert the XML documents to HTML. LaTeXML has several options to handle math elements from LaTeX. They include translating formulae to presentation MathML and even to content MathML. The original LaTeX code can also be preserved as an annotation. Using all three methods results in MathML elements as displayed in Figure 2. However, the LaTeXML manual states that the generated content MathML is "currently rather experimental" – we can thus not assume that it reflects the intended semantics of the original formula.

LaTeXML was used to convert the online e-print archive, arXiv, from LaTeX to HTML5. The resulting corpus is called arXMLiv [SKG+10]. The arXiv[4] contains articles from areas such as physics, mathematics, computer science, quantitative biology, quantitative finance and statistics. The articles are mostly uploaded as LaTeX files and published as PDF and Postscript documents. PDF and Postscript files are obviously impractical for any further processing, but also the LaTeX source is not intended for any other purpose than typesetting. Instead a content-oriented format, such as XML or HTML, simplifies further reasoning and manipulation of the document. We use the

[4] http://www.arxiv.org

```
<apply>
  <times/>
  <apply>
    <times/>
    <cn>3.0</cn>
    <apply>
      <power/>
      <cn>10</cn>
      <apply>
        <minus/>
        <cn>17</cn>
      </apply>
    </apply>
  </apply>
  <csymbol cd="micro">μ</csymbol>
  <csymbol cd="meter">m</csymbol>
</apply>
```

Figure 4: A Content MathML Expression for "$3.0 \cdot 10^{-17}\mu\mathrm{m}$".

arXMLiv as the underlying corpus of this thesis.

3.3 KAT

The KWARC Annotation Tool (KAT, [DGK+14] and [SW16]) is an annotation tool with support for mathematics which is implement in JavaScript and executed by a browser. It allows the definition of annotation formats in the form of "KAT Annotation Specifications (KAnnSpecs)". Besides an annotation mode, the tool also contains a review mode, in which a user can rate existing annotations – for instance for the evaluation of an automated annotation system. Annotations can be stored as Resource Description Framework (RDF, [SR14]) documents and imported in the same way. Figure 5 shows a screenshot from KAT.

3.4 Astropy

Astropy [Ast13] is an open source python library for astronomy and contains a very useful sub-package for units and quantities. The package supports units from different systems of measurement including SI units, imperial units and astronomy-specific units, such as solar masses. For a given unit, one can query for equivalent units – units of the same dimension – and convert the units respectively. One can also directly convert a unit to a system of choice. Astropy can easily be extended with new units and new equivalences for them.

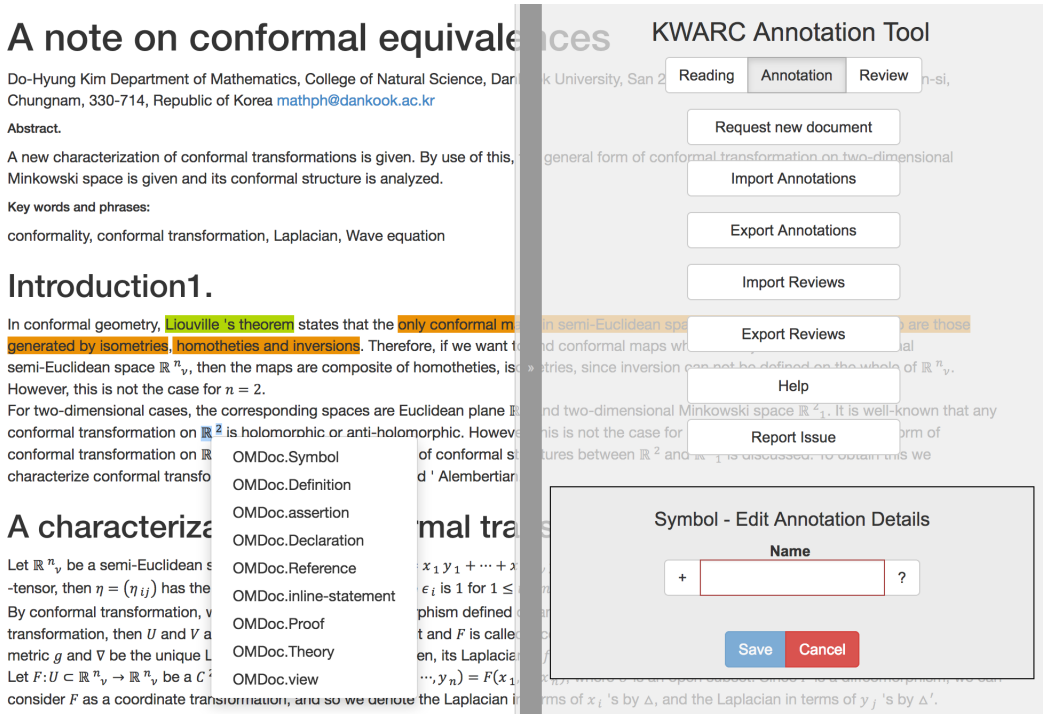


Figure 5: Creating an Annotation in KAT (Screenshot from [SW16]).

4 Research Problem

In this thesis, we want to investigate the extraction of semantics from the arXMLiv corpus and demonstrate its utility with three prototypical semantic applications including semantic search, in-document conversion and the improvement of current accessibility programs.

The main focus is the detection of quantity expressions in HTML5 documents with MathML content. For that, we need to examine suitable detection methods and an adaptive and extensible architecture for the implementation. Compared to that, the semantic services, the input for Tom Wiesing's search engine (compare Section 1.2) and the automatic unit conversion can be implemented rather straightforwardly. For the enhancement of the screen readers, we first need to figure out how to feed additional information to them.

We now derive requirements which the implementation and its architecture need to fulfill. In Section 2.2, we have already observed that unit strings can be ambiguous and we thus need a way to take care of these uncertainties. One possible way to do so, would be to implement heuristics which try to guess the correct meaning during the parsing of the expression. This has the major disadvantage that we would loose information very early in the process. Instead, the desired behavior is to allow multiple meanings for an

expression. So rather than disambiguating expressions directly, we want to have a set of possible meanings for an expression. As an additional and independent task, we can then rate the likelihood of the different meanings, say, that "gigahertz" is more likely than "gauß · hertz" for the unit string "GHz". An application based on the results of the detection of quantity expressions, can then decide how to deal with ambiguities. A search engine could, for instance, include all ambiguities. In contrast to that a tool for the automatic conversion of quantity expressions could ask its user for help with the disambiguation or simply use the most likely meaning.

We can derive an additional requirement from the phenomenology in Section 2. We have seen that many quantity expressions are written in LaTeX as a mixture of math and text mode. While this is no problem for humans, it adds additional complexity to the encoding in HTML documents. Hence the detection needs to work on math and text mode as well as on the mixture of both in order to be feasible for our problem.

From an architectural point of view, we want to support different detection routines as well as different methods to rate the likelihood of the detected expressions.

5 Implementation

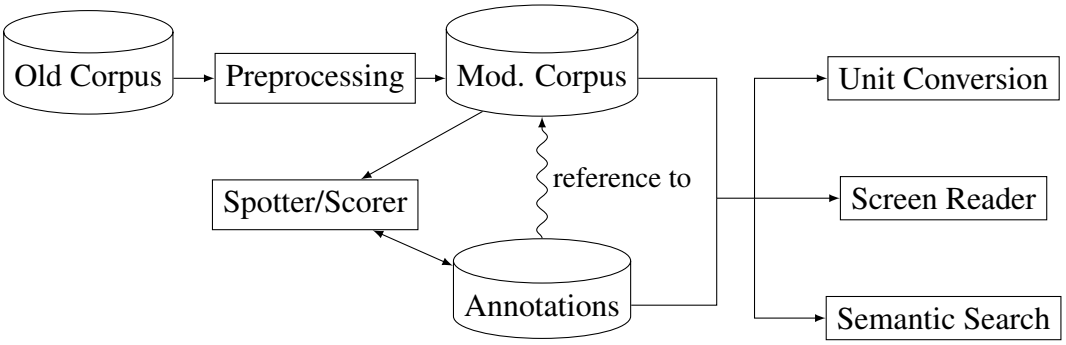


Figure 6: The Architecture of the Implementation.

We present the architecture of our implementation in Figure 6. It consists of three disjoint parts, where the first one is a small but important preprocessing step.

Quantity expressions are spotted as a second step. The results are stored as separate annotations which contain references to the documents. Rating the likelihood of the annotations is a subtask of this step. Although expressions often only have one meaning, the author also observed cases with up to six meanings for an expression. For these cases, it is especially relevant to assign preferences to the meanings.

We refer to a program as *spotter*, if it creates annotations from the document. In addition to that, we call a program *scorer*, when it is working on existing annotations and manipulates their value of likelihood. As an example, say, a spotter finds "5 GHz" and

infers “5 gigahertz” and “5 gauß · hertz” as possible meanings. Adding a preference to these two meanings is then a task for the scorer which adds a higher value of likelihood to the first meaning. The architecture permits multiple spotting and scoring routines and can easily be extended by additional ones.

Semantic services are based on the documents and their annotations and form the last part.

The implementation uses the LLaMaPUn library [GS] which was created by Deyan Ginev and Jan Frederik Schäfer especially for processing arXMLiv documents. It is written in Rust. Due to compatibility, most of the author's implementation is also written in Rust. The code of the implementation is open source and available under GPL license[5].

The structure of this section follows the structure of the architecture. We thus first describe the preprocessing step (Section 5.1) and then the spotting and scoring of quantity expressions including the output format (Section 5.2). An evaluation of the presented application is presented in Section 5.3.

5.1 Text Tokenization

In the original LaTeXML documents, text passages are positioned in text nodes without any further partitioning. In this setup, KAT can only annotate complete text passages, since it requires references – in the form of node identifiers – to the first and the last node that should be annotated, as well as to their lowest common parent. Text nodes do not have identifiers, so the only possible reference points to the parent of the text node, hence the whole parent is annotated instead of just a part of its text. This problem can be solved by wrapping every word of a text passage into an own node with an unique identifier. We call this process *text tokenization* and perform it as a first step on the given documents. For the following processing, we can thus assume that texts are tokenized.

Words are detected by splitting text on whitespaces and on special non-alphabetic characters like e.g. ’=’; for example in “Var=5”. Punctuation characters, other non-alphabetic characters and numbers are separated from words but wrapped in exactly the same way. This tokenization could for instance be further improved by adding part-of-speech tags to every node.

5.2 Spotting Quantity Expressions

In Section 4, we derived the requirement for an approach which tolerates the mixture of math and text mode. There is only one format for text, but there are three different formats for math in MathML, namely presentation MathML, content MathML and

[5] `https://gl.kwarc.info/urabenstein/Semanticextraction/tree/master/spotter`

the original LaTeX source. The author found that using the representation in content MathML turned out to be the most general approach, even though the LaTeXML conversion to this format is still experimental.

In this section, we first show the XML representation of some of the examples from Section 2 to familiarize the reader with it (Section 5.2.1). After that, we discuss two procedures for the spotting, one for simple multiplicative and divisive quantity expressions, as well as complex quantity expressions and a different one for quantity expressions with units in superscripts. The first one is split in two parts, the extraction of potential expressions (Section 5.2.2) and the parsing of them (Section 5.2.3). The latter is explained in Section 5.2.4.

After that, we describe the output format for the extracted expressions (Section 5.2.5). In Section 5.2.6 we discuss observations of the spotting results and derive rules for a scoring system from them.

5.2.1 XML Representation of Quantity Expressions

Showing the XML representation of all examples from Section 2, would result in a large and quite redundant enumeration. Instead we only show a small subset of them and display only the content MathML part, since only this part is used for the implementation. We also omit attributes to increase readability.

The examples in Figure 7 belong to the category of simple multiplicative units, while the examples in Figure 8 additionally include superscripts. In both tables, the examples on the left side are completely in math mode and the ones on the right side are partly in math and partly in text mode. In the latter table, we see that the way superscripts are encoded is totally different, depending on whether the expression is purely in math mode or not. Additionally Figure 9 shows the encoding of the example "$30°$" where the unit itself is in superscript. It is interesting to observe that LaTexML falsely encodes the degree symbol using the compose element which is indented to represent function composition. We suggest a correct representation on the right side of Figure 9. Instead of the compose operator, it uses the csymbol element with the degree symbol "°" as content and a reference to the meaning in the `cd` attribute. Besides representing the correct meaning, it has the advantage that it fits into the same schema as the content MathML encoding of other quantity expressions; namely that it is based on a multiplication of a number and a unit symbol. However, even though the markup generated by LaTeXML does not represent the semantic of the original expression, we can still extract the correct meaning from it.

We will refer to these examples in the next section where we describe the detection routine for quantity expressions.

```
<math>
  <apply>
    <times/>
    <cn>6</cn>
    <ci>μ</ci>
    <mtext>m</mtext>
  </apply>
</math>
```

```
<span>200</span>
<span> </span>
<math>
  <ci>μ</ci>
</math>
<span>m</span>
```

Figure 7: The XML Representation of Example 4 (“6 μm”, left side) and 6 (“200 μm”, right side) from Table 3.

```
<math>
  <apply>
    <times/>
    <cn>1.0</cn>
    <apply>
      <csymbol>
        superscript
      </csymbol>
      <mtext>Wcm</mtext>
      <apply>
        <minus/>
        <cn>2</cn>
      </apply>
    </apply>
    <ci>μ</ci>
    <apply>
      <csymbol>
        superscript
      </csymbol>
      <mtext>m</mtext>
      <cn>2</cn>
    </apply>
  </apply>
</math>
```

```
<math>
  <cn>100</cn>
</math>
<span> </span>
<span>km </span>
<span>s</span>
<math>
  <apply>
    <apply>
      <minus/>
      <cn>1</cn>
    </apply>
  </apply>
</math>
<span> </span>
<span>Mpc</span>
<math>
  <apply>
    <apply>
      <minus>
      <cn>1</cn>
    </apply>
  </apply>
</math>
```

Figure 8: The XML Representation of Example 2 (“$1.0\ \mathrm{Wcm^{-2}}$μm^2”, left side) and 5 (“100 km s^{-1} Mpc^{-1}”, right side) from Table 5.

```
<math>
  <apply>
    <csymbol>superscript</csymbol>
    <cn>30</cn>
    <compose/>
  </apply>
</math>
```

```
<math>
  <apply>
    <times/>
    <cn>30</cn>
    <csymbol cd="degree">°</csymbol>
  </apply>
</math>
```

Figure 9: The XML Representation of Example 1 (“30°”) from Table 6. On the left, we see the semantically incorrect encoding generated by LaTeXML. We present a correct encoding on the right.

5.2.2 Format of Potential Quantity Expressions

In order to spot quantity expressions, the rough pattern for their detection consists of a numeric expression followed by prefix and unit symbols. There are two strains of this pattern, one for expressions that are completely in MathML and one which involves changes between text and math. We discuss the former first.

For the detection of quantity expressions in MathML, we first define the two terms *numeric expression* and *unit symbol expression*. Numeric expressions are inductively defined as follows.

- cn nodes are numeric
- apply nodes are numeric if the first child is a
  - csymbol node with content “superscript”,
  - ci node with content “⁢”,
  - minus node or a
  - times node

  and all other children are numeric.

This definition allows the detection of terms like “$5.0 \cdot 10^{-20}$”. It is derived from observations of the LaTeXML documents rather than from the specification of content MathML.

We use the term *unit symbol expression* to denote the encoding of unit symbols, possibly with their corresponding exponents, in MathML. There are three different cases for unit symbol expressions. The first case consists of ci and mtext nodes which contain unit symbols, e.g. “eV” and “W”. In case of additional exponents, we get the following format which represents the second case.

```
<apply>
  <times/> or <divide/>
   ⟨⟨ Numeric expression ⟩⟩
   ⟨⟨ Unit symbol expressions ⟩⟩
</apply>
```

Figure 10: Basic Pattern for the Detection of Quantity Expressions in Content MathML.

```
<apply>
  <divide/>
  <apply>
    <times/>
     ⟨⟨ Numeric expression ⟩⟩
     ⟨⟨ Unit symbol expressions ⟩⟩
  </apply>
   ⟨⟨ Unit symbol expressions ⟩⟩
</apply>
```

Figure 11: Divide Pattern for the Detection of Quantity Expressions in Content MathML.

```
<apply>
  <csymbol>superscript</csymbol>
  <ci> or <mtext> node
   ⟨⟨ Numeric expression ⟩⟩
</apply>
```

Note that ci and mtext nodes are further processed in exactly the same way, since both are often used similarly, as we observed in Section 2.2. apply nodes indicating a subscript are also unit symbol expressions, but depict a special case. Nodes with subscripts are ignored by default and only exceptions are further processed. At the moment, these are limited to astronomic mass and radius units, like "$M_\odot$" for the mass of the sun and "$R_\oplus$" for the radius of the earth, compare Examples 8, 9 and 10 from Table 3.

Using the definition of numeric and unit symbol expressions, we can now distinguish different formats of quantity expressions in MathML. There are three cases to consider, one that is based on multiplication, one on division and one that contains both. This can be illustrated by the following three examples: "$5\ \mathrm{s}$", "$5/\mathrm{s}$" and "$5\ \mathrm{m}/\mathrm{s}$". The general pattern of the encoding of the first two is displayed in Figure 10 with times or divide respectively. The latter example is encoded with the operator priority "$(5\ \mathrm{m})/\mathrm{s}$" – the corresponding content MathML pattern in shown in Figure 11.

Two samples matching the presented patterns are displayed on the left side of Figure 7 and 8. For the former, we have the numerical expressions <cn>6</cn> as well as the unit symbol expressions <ci>μ</ci> and <mtext>m</mtext>. For the further analysis, we simplify

the unit symbol expressions to a list of strings with the corresponding exponents being attached. Since the expressions from Figure 7 do not have additional exponents, we convert them to [(μ, 1), (m,1)]. Similarly, we get [(Wcm, -2), (μ, 1), (m, 2)] for the second matching sample in Figure 8.

We have discussed the pattern for the detection of quantity expressions in content MathML. Quantity expressions which are written as a mixture of math and text mode need to be detected differently. In this case, we search for either a span node that contains a number or for a math node that is a numeric expression or ends with an numeric expressions, as for instance in the encoding of "`$X = 5$ GeV`". After that, we consider math nodes consisting of unit symbol expressions as well as spans which contain a prefix, a unit or a combination of both. Superscript expressions are attached to the unit that occurred before it. Potential quantity expressions detected this way, are then simplified in the same way as the ones being detected completely in content MathML, i.e. [(μ, 1), (m, 1)] for the example on the right hand side of Figure 7. For the one on the right hand side of Figure 8, we find the symbols "km", "s" and "Mpc" in span nodes. For the latter two, we detect corresponding exponents and thus simplify it to [(km, 1), (s, -1), (Mpc, -1)].

5.2.3 Parsing Potential Quantity Expressions

Potential quantity expressions are parsed and as a result either rejected or decomposed to the units they contain. We have already discussed that quantity expressions are ambiguous (compare Section 2.2), we hence allow multiple decompositions for the same expression. Instead of using a grammar framework, the author implemented a rather ad hoc parsing approach which has proven to be both flexible and effective. This approach implements the following rules and parses a list of characters beginning at the end:

R1 When a string matches the name of a unit symbol (i.e. "W" for "watt"), the parsing continues to search for a corresponding prefix as well as for another unit.

R2 When a unit is found (with or without prefix), its exponent is set to the attached exponent of the string. In case this is the second unit found in a string, we find two meanings of it – one with the exponent that is attached to the string and one with exponent "1".

R3 The candidate for a quantity expression is rejected if the list of strings cannot be decomposed into units completely.

R4 Characters indicating a division of units (i.e. "/") are recognized and processed accordingly.

For instance, for the input [(GHz, 1)], there are two possible meanings found; one is "gigahertz" and the other one is "gauß · hertz". This is due to Rule R1, which says that after a unit symbol ("Hz") was found, the parsing continues to search for both a prefix ("giga") and a unit symbol ("gauß"). In this case, both attempts are successful. Of course, "gigahertz" is much more likely to be meant here, but without any further knowledge, the other option cannot be ruled out completely.

There is also an ambiguity for the MathML representation of "`$\text{m/s}^2$`" which is translated to [(m/s, 2)]. Because of Rule R2, the exponent of "s" is always set to "2", but there are two cases for the exponent of "m", namely "1" and "2". Hence the two possibilities "$(\text{meter}/\text{second})^2$" and "$\text{meter}/(\text{second}^2)$" are detected. The first one makes more sense, when looking at the LaTeX and MathML code of the statement, while the second one is closer to its textual representation ("m/s^2"). According to Rule R4, the slash is recognized and it is hence detected that the term consists of a multiplicative unit ("meter") and a divisive unit ("second"), both having their corresponding exponent - based on the ambiguity.

5.2.4 Spotting Units in Superscript

The approach just described does not capture quantity expressions with units in superscript and can also not be easily extended accordingly. Thus, a separate procedure is necessary for this case.

Figure 9 shows an example of the MathML encoding of a simple degree expression. We have already discussed that the encoding on the left of the figure does not represent the meaning of the underlying term (compare Section 5.2.1), but we can still detect these expressions by hardcoding their format. For that, we allow any numeric expressions as the second child of the apply node on the left side of Figure 9. When the pattern is followed by a "C" or "F", then the unit is classified as degree Celsius or degree Fahrenheit; otherwise it is classified as degree of arc. Quantity expressions involving arc minutes ("′") and seconds ("″") are encoded similarly with the only difference that the symbols are wrapped in ci nodes. They are also detected via a fixed format. Geographical coordinates, like "$-29°50′58″$" (Example 3 from Table 6), are detected as a composition of these patterns. Coordinates as in Example 2 from Table 6 ("$00^h49^m37^s.71$") are not yet detected, due to a lack of representative examples in the documents.

5.2.5 Format of the Annotations

The spotted quantity expressions are stored in an RDF-based format [SR14] suitable for the annotation tool KAT (compare Section 3.3). Figure 12 displays an example of highlighted quantity expressions in KAT. The expressions were spotted by the presented application and are thus limited by its restrictions (compare Section 2.3). Hence the expression "$10^{-12}\ M_\odot\Omega_a h^2$" is not detected as it involves constants. These visualizations

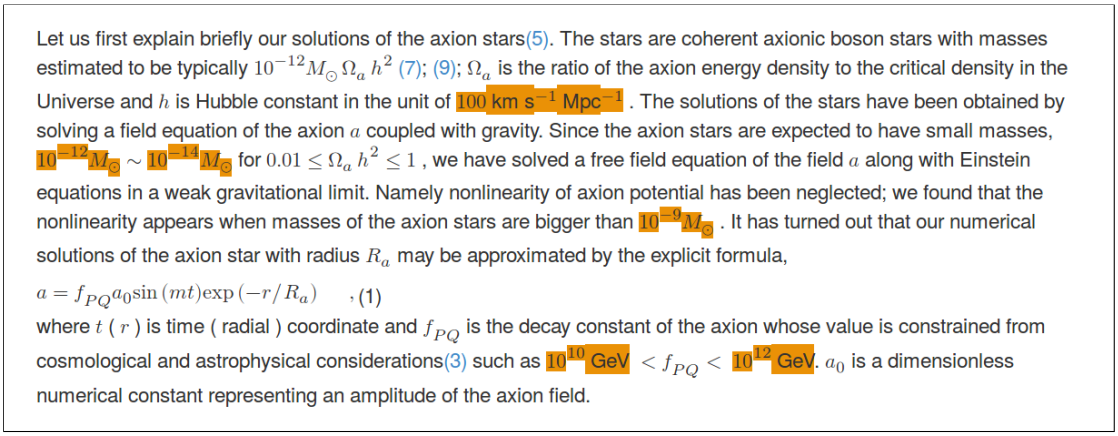


Figure 12: Highlighted Annotations in KAT for the Quantity Expressions Spotted in [Iwa98].

can easily be used for instance for quality control. The format is described in detail in [DGK+14] and [SW16]. This section gives only a very brief introduction to the format and points to the references for details.

We further describe a truncated example of the annotation of "10^{10} GeV" which is shown in Figure 13. The entry has a unique id and annotates the document "hep-ph9807232.html". The path to the document contains three XPath expressions; the first one pointing to the deepest node that fully contains the annotated expression, the other two reference the first and the last element of the expression. XPath is query language for XML documents and the expressions reference nodes via their unique identifier. Additionally, the semantics of the discovered quantity expressions is attached to the entry in a content MathML format. Two terms are attached here, due to the ambiguity of "GeV". Some technical parts of a KAT entry are omitted in this example, because they do not contain any semantic information.

The full skeleton of our content MathML encoding for quantity expressions is displayed in Figure 14 and consists of three parts: A numeric expression, a set of multiplicative units and a set of divisive units. Each set can be empty or have just one element, thus the MathML expressions can be simplified accordingly, as in the examples in Figure 13. We have defined the term "numeric expression" in Section 5.2.2 but use a slight modification for the output format. For that, we replace <csymbol>superscript</csymbol> nodes by <power/> and <ci>·</ci> nodes by <times/>. This results in a representation which is closer to the semantic of the underlying expression and removes artifacts created by the conversion application LaTeXML.

Figure 15 displays the skeleton for units. The prefix can be omitted for units without prefix. The exponent can be also be omitted which indicates the default value "1". Prefix and unit symbols coincide with the symbols found in the document. We use the

```
<rdf:Description rdf:nodeID="KAT_123">
  <kat:annotates rdf:resource=
    "http://localhost:3000/content/hep−ph9807232.html#cse(
    //∗[@id='S2.p5.1'],
    //∗[@id='S2.p5.1.w36'],
    //∗[@id='S2.p5.1.w38'])">
  </kat:annotates>
  <kat:contentmathml rdf:parseType="Literal">
    <apply>
      <times/>
      <apply>
        <power/>
        <cn>10</cn>
        <cn>10</cn>
      </apply>
      <apply>
        <csymbol cd="Prefix">Prefix</csymbol>
        <csymbol cd="Giga">G</csymbol>
        <csymbol cd="electron volt">eV</csymbol>
      </apply>
    </apply>
  </kat:contentmathml>
  <kat:contentmathml rdf:parseType="Literal">
    <apply>
      <times/>
      <apply>
        <power>
        <cn>10</cn>
        <cn>10</cn>
      </apply>
      <apply>
        <times/>
        <csymbol cd="Gauss">G</csymbol>
        <csymbol cd="electron volt">eV</csymbol>
      </apply>
    </apply>
  </kat:contentmathml>
</rdf:Description>
```

Figure 13: An Example of a KAT Entry for the Quantity Expression "10^{10} GeV" in [Iwa98]. The URL was decoded for this example.

```
<apply>
  <power/>
  <apply>
    <csymbol cd="Prefix">Prefix</csymbol>
    <csymbol cd= ⟨⟨ PrefixName ⟩⟩ > ⟨⟨ PrefixSymbol ⟩⟩ </csymbol>
    <csymbol cd= ⟨⟨ UnitName ⟩⟩ > ⟨⟨ UnitSymbol ⟩⟩ </csymbol>
  </apply>
   ⟨⟨ Numeric Term ⟩⟩
</apply>
```

Figure 15: The General Content MathML Output Format for a Spotted Unit. The Prefix and the Exponent can be omitted.

`cd` attribute to encode the meaning of the symbols, for instance “micro” for the symbol “μ”. The values of the `cd` attribute are chosen in such a way, that they can be passed to Astropy – the tool we use for the conversion of units (compare Section 3.4).

5.2.6 Scoring of Spotted Quantity Expressions

```
<apply>
  <times/>
   ⟨⟨ mod. Numeric Term ⟩⟩
  <apply>
    <divide/>
    <apply>
      <times/>
       ⟨⟨ Units ⟩⟩
    </apply>
    <apply>
      <times/>
       ⟨⟨ Units ⟩⟩
    </apply>
  </apply>
</apply>
```

Figure 14: The General Content MathML Output Format of a Spotted Quantity Expression.

The presented spotting procedure is very general and its results not only include multiple meanings for the same expression but also terms which were mistakenly detected as quantity expressions. We thus want to use a scoring system for two reasons, the first is to express preferences between different meanings and the second is to eliminate errors. We do not want to remove data from the annotation files, but instead use negative scores to mark certain expressions as wrong detections. Scores are attached to the annotations by adding the `score` attribute to the corresponding kat:contentmathml node (compare Figure 13). We first describe methods to evaluate the likelihood and then procedures to detect errors in the spotting results.

Multiple meanings are found for unit string such as “GHz” and “mm”, namely “gigahertz” and “gauß · hertz” for the former and “meter · meter” and “millimeter” for the latter. From these examples, we derive the rule that meanings involving less units and thus more prefixes are more likely and hence attach a higher score to them.

Superscripts are another source of ambiguity. In this case, we prefer meanings in which the exponent only applies to the last unit, since this is closer to the visual representation and thus to the likely intended meaning. Like this, we prefer, for instance, "$\mathrm{W} \cdot (\mathrm{cm}^2)$" over "$(\mathrm{W} \cdot \mathrm{cm})^2$" for the first part of Example 2 from Table 5, although its formal representation in LaTeX is closer to the latter meaning.

The word "as" is often falsely detected as the unit "attosecond" and thus causes errors. This happens for instance in phrases like

> "[...] $\langle \phi \rangle = 0$ as the temperature drops below [...]" (from [GG92])

and in

> "[...] 450 MeV as suggested by [...]" (from [GNR92]),

for which we find the quantity expressions "$0\ \mathrm{as}$" and "$450\ \mathrm{MeV} \cdot \mathrm{as}$". We eliminate both errors by giving them negative scores, but add the additional meaning "$450\ \mathrm{MeV}$" for the latter case. The score of the additional meaning is based on the methods describe above.

Jan Frederik Schäfer developed a declaration spotter [Sch16] and we can exploit his results to find errors in the detection of quantity expressions. The following declaration is an example from the document [CPP92].

> "[...] for each value of λ, δ, g and N one has to [...]"

The document then contains expressions like "$10\ N$". Our spotter wrongly classifies the latter as the quantity expression "10 Newton". Since Frederik's spotter detects that "N" is a declared variable, we can rule out the quantity expressions involving it – again by attaching a negative score to them.

5.3 Evaluation

In this section, we analyze the results of the application. At first, we describe the performance and the quality of the outcomes (Section 5.3.1), followed by observations of the author about common sources for false classifications (Section 5.3.2).

5.3.1 Quantitative Evaluation

The implementation was tested on a set of about 35000 documents and could successfully process nearly all documents. Fatal errors occurred only in about 150 documents.

	Quantity Expressions (QE)	Other Expressions	Total
Detected as QE	598	195	793
Not Detected as QE	48	-	48
Total	646	195	

Precision	$P = 598 \,/\, (598 + 195)$	$\approx 75\%$
Recall	$R = 598 \,/\, (598 + 48)$	$\approx 93\%$
F-Score	$2 \cdot P \cdot R/(P + R)$	$\approx 83\%$

Table 12: Results of the Manual Evaluation.

The processed documents as well as their annotations are publicly available[6]. The overall runtime was about 80 hours on nine 2.00 GHz cores of a Intel Xeon E5-2650 processor. This is equivalent to a runtime of about 70 seconds per document on a single core. It includes preprocessing as well as spotting and scoring of quantity expressions. Note that, the scoring task involves the execution of Schäfer's declaration spotter which adds a significant amount of runtime.

The author evaluated the quality of the implementation by manually validating 50 randomly selected documents. They include 646 quantity expressions of which 598 were successfully recognized (true positives). We regard the detection of a quantity expression as correct when its highest scored possible meaning reflects the correct meaning of the expression. 48 quantity expressions were not detected (false negatives) and 195 times expressions were marked as quantity expressions although they are not (false positives). In this setup, true negatives are equivalent to non-quantity expressions which were successfully not detected. However, there is no meaningful quantification of these expressions in this case and we thus omit them. The evaluation results in a precision of $598 \,/\, (598 + 195) \approx 75\%$ and in a recall of $598 \,/\, (598 + 48) \approx 93\%$. We can thus derive a F-score of about 83%. The results are summarized in Table 12.

While the recall of the implementation is already very promising, we observe that the precision offers potential for improvements. From that, we can derive that the implemented pattern is rather too wide than too narrow, in the sense that it marks too many terms as quantity expressions.

The detailed evaluation results, including the selected documents and the number of

[6] `https://kwarc.info/datahost/semanticextraction/`
There is one important remark to make about the annotations of these documents: In Section 5.2.5, we described a slight modification of numeric expressions. This modification does not apply to these documents, since it was introduced when the evaluation was already completed. Hence the numeric expressions in these annotations, still contain nodes such as <csymbol>superscript</csymbol> instead of <power/>. This does not influence the results of the evaluation.

detected quantity expressions per document, are available in the author's repository[7].

5.3.2 Qualitative Evaluation

In this section, we discuss some common sources for misclassifications observed by the author of this thesis.

Humans can easily disambiguate that an expression, such as "$10^{-7} - 10^{-6}$ yr^{-1}" from [KDKR00], denotes a range of values and that the "$-$" in the expression does not stand for subtraction. However in the generation of content MathML, LaTeXML converts this symbol to a minus operation and such expressions are thus detected as plain quantity expressions instead of as range expressions.

Abbreviations and variables are also a source of errors. For example, the following snippet defines the meaning of "s" and "S" for the paper [YS00].

> "Ground-state magnetization curves of ferrimagnetic Heisenberg chains of alternating spins S and s are numerically investigated."

After that, the authors often mention the "$2s$-plateau magnetization curve" and the "$2S$ chain period". The spotter recognizes the former as "two seconds" and the latter as "two siemens"[8] which causes more than forty misclassifications in this document. Also common phrases like "in the 70s" (from [vA00]) result in errors where "70 seconds" is detected while the author referred to an event that took place in the time span from 1970 to 1980. Astronomic names cause errors in a similar way. The authors of [MS98], for instance, reference the supernovae "1979C, 1992H and 1993W", resulting in the false detection of "1979 coulomb"[9], "1992 henry"[10] and "1993 watt".

Many of these misclassifications consist of single-letter unit string, such as "C", "H" or "W", since they are – without additional context – hard to distinguish from plain identifiers.

6 Semantic Services

The author implemented three prototypical applications to demonstrate the benefit of the gathered semantic information. The first is a service for the conversion of units in documents (Section 6.1). We then describe an enhancement for screen reading applications with semantic information (Section 6.2). In Section 6.3, we explain how to exploit our results for a semantic search engine. The semantic services are also displayed in Figure 6 which describes the architecture of the implementation.

[7] `https://gl.kwarc.info/urabenstein/Semanticextraction/blob/master/evaluation/evaluation.csv`

[8] Siemens is a unit of electric conductance.

[9] Coulomb is a unit of electric charge.

[10] Henry is a unit of inductance.

6.1 Conversion inside Documents

This section describes unit conversion inside documents as a semantic service. It is based on the results of the detection of quantity expressions. We first discuss use cases and then present the implemented service. We describe its implementation at the end of this section. This part of the author's work has been already been published as a part of a report about OpenDreamKit [WK17] and we follow the presentation of this report.

6.1.1 Use Case

In order to convert a document from one system of measurement to another, a user currently has to open a conversion service and enter the data. This is not only a distraction from the document, but also a source of errors due to the manual entering of the data. Automatic unit conversion in documents can solve these problems. Imagine, for instance, a recipe written in imperial units such as "fluid ounces" and "quarts" that someone who is used to metric units finds on the Internet. Converting that recipe manually is a unnecessary burden and it is instead desirable to offer the automatic conversion to units of his choice. This service can also simplify the understanding of scientific documents which contain unusual units like "solar masses". This thesis offers a service for the latter task.

6.1.2 Service

The implemented service allows us to first highlight all quantity expressions that were spotted in a given document. Spotted quantity expressions which are not modified are highlighted in orange. Figure 16 shows the result of this operation and we further refer to this snippet as a running example.

We can then either convert units locally, only in one expression, or use a global conversion, which affects all units in a document. Say, we first want to convert "watt" to "horsepower" in the given example. Figure 17 shows the units in which we can convert "watt", as well as the result of the operation. Modified quantity expressions are highlighted in green, such that we immediately observe the difference to unmodified expressions. Additionally, we can also convert all units in the document to irreducible SI base units (also Figure 17). This global conversion is a first prototype and could be further extended to convert units between different systems of measurement – for instance from metric to imperial units and the other way around.

6.1.3 Implementation

The unit conversion service is implemented using Jobad [Wie17] – a javascript framework for interactive web pages –, Flask [Ron17] – a web development framework for

$$\tan\theta' = \frac{\sqrt{1+\alpha I\lambda^2}-1}{\sqrt{\alpha I\lambda^2}}\tan\theta\ . \quad (12)$$

Equation (12) looses validity as soon as target deformations start to become significant. The validity also depends on the accuracy of the mean longitudinal momentum given as a function of intensity. For $I\lambda^2 = 1.0\cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain an ejection angle of $\theta' = 14^\circ$ and for $I\lambda^2 = 2.0\cdot 10^{18}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain $\theta' = 17^\circ$ from the simulations. This yields $\alpha^{-1} \approx 8.0\cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$.

Figure 16: Highlighting Spotted Quantity Expressions in [RSM+98]. Figure from [WK17].

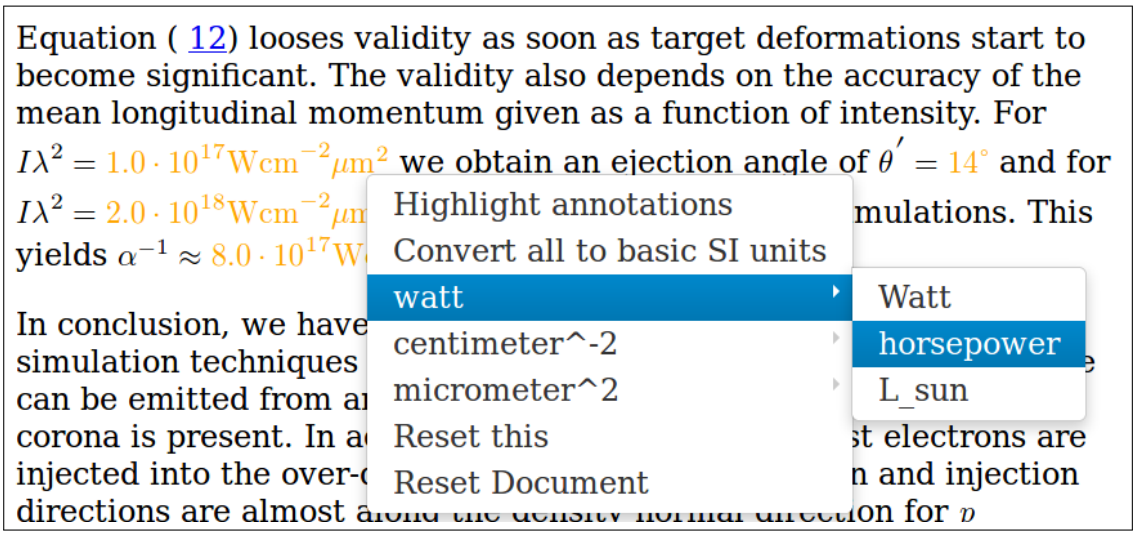
python and Astropy [Ast13] – a community python library for astronomy which contains a useful and extendible module for quantities and units (compare Section 3.4).

The implementation of the conversion service consists of two parts, the server and the Jobad module. The server contains the documents and their annotations and calls Astropy. Jobad requires a special header in every document to be activated. The header is automatically added during the processing of the documents. The Jobad module parses the annotations. In case of ambiguity it selects the meaning with the highest score. Annotations with a negative score are dismissed. Additionally it unifies the appearance of quantity expressions. This is especially useful to get rid of expressions written in a mixture of text and math mode and it simplifies further processing. The new format is completely in presentation MathML and created from the content MathML of the annotation. It follows the guideline for displaying units in MathML (see [DH03]), by encoding units in <mi mathvariant="normal"> nodes. The attribute ensures that units are written in an upright font. For highlighting, we use the `mathcolor` attribute. The query for equivalent units, as well as the conversions are delegated to Astropy. The result of the conversion is then also displayed using presentation MathML with a precision based on the number of significant figures of the original number.

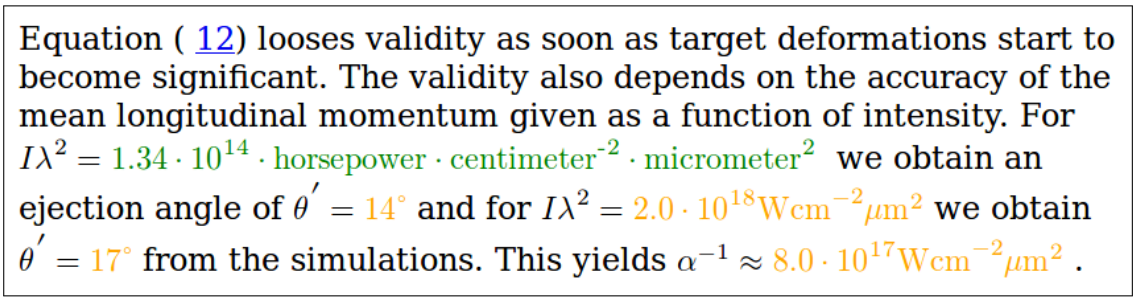
There is an online version of the service available[11] where a user can select a document and then use the functionalities as described in Section 6.1.2. It requires to have a browser which supports MathML. The author tested the service in Firefox. The code of the implementation is under GPL license can be found in the author's repository[12].

[11] `http://ash.eecs.jacobs-university.de/`

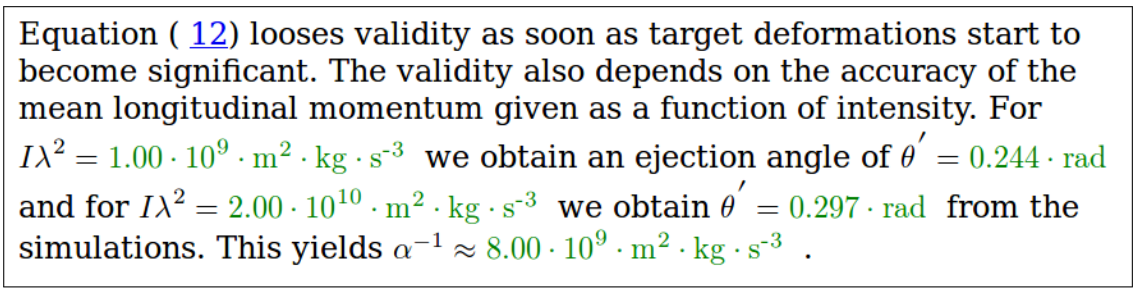
[12] `https://gl.kwarc.info/urabenstein/Semanticextraction/tree/master/server`



(a) Choosing A Target Unit



(b) The Result of converting one QE



(c) Converting a Document To SI

Figure 17: In-Situ Unit Conversion. Figure from [WK17].

6.2 Semantics for Screen Readers

The author collaborated with Daniel Hajas, the founder of Grapheel – an initiative for better access to STEM documents for visually impaired people, to investigate methods to improve the output of screen reading programs with semantic information.

6.2.1 Motivation

In a joint report [HKR17], Daniel Hajas explains that "access to Science, Technology, Engineering, Mathematics (STEM) information can be particularly challenging for visually impaired people. The routes of the challenges are

- the extensive use of graphical information e.g. graphs and diagrams,
- the highly symbolic language used for expressing mathematical relations,
- often in an out-of-line mode for example sub-, and superscripts, or fractions.

People with insufficient sight rely on screen reader software solutions either complementary or completely substituting screen magnification solutions. Screen readers can process computerised input and return output in either refreshable braille, or synthetic speech, known as Text-To-Speech (TTS). The principal way screen readers work is a line by line rendering of text. This is why WYSIWYG equation editors producing a visual '2D' output (e.g. fractions, binomial coefficients) are often inaccessible. Markup languages such as, LaTeX, or MathML offer in-line text processing and therefore are considered to be screen reader compatible." However, although screen readers are able to work with MathML, it is still desirable to improve their output in order to enhance user friendliness. We investigated methods to add the semantics of quantity expressions to the documents to allow screen readers to read the meaning instead of the representation. This not only includes simple cases, like "1500 V" for which a screen reader should read "1500 volt", but also more complex examples. For instance, the NVDA screen reader reads "$3\ \mathrm{M}_\odot$" as "three M sub circled dot operator" while it would be nice to hear "three solar masses". We use the former example as a running example.

6.2.2 Experiments

We have examined three different methods to add semantic labels to arXMLiv documents. All of them include the use of the `aria-label` attribute with which one can add descriptions to nodes in HTML5 documents for accessibility purposes [CC14].

As a first approach, we added aria labels to presentation MathML nodes as in Figure 18a. We refer to this approach as *deep labeling*, as it adds the labels directly to the corresponding nodes. In addition to that, we have tried to annotate whole expression by wrapping them in labeled div nodes with the value of the `role`-attribute being set to

```
<math>
  <mn>1500</mn>
  <mi aria-label="Volt">V</mi>
</math>
```

(a) An Example of a Presentation MathML Expression with Deep Labels.

```
<div role="math" aria-label="1500 Volt">
  <math>
    <mn>1500</mn>
    <mi>V</mi>
  </math>
</div>
```

(b) An Example of Presentation MathML wrapped in a div Node.

```
<div role="math" aria-label="1500 Volt">
  <span>1500</span>
  <span> </span>
  <span>V</span>
</div>
```

(c) An Example of span Nodes wrapped in a div Node.

Figure 18: Different Possibilities of Adding Aria Labels to Nodes in HTML5.

`math`. For that, we have considered two different versions: One where we wrap presentation MathML and one where we also wrap span nodes (compare Figure 18b and 18c).

We have tested the configurations with the screen readers *NVDA 2016*[13] and *JAWS 17.0*[14] on Windows with Firefox 50 and Internet Explorer 11. For both of the screen readers, there is the option to enable the *MathPlayer*[15] plugin to support MathML. On Mac, we have tried *VoiceOver*[16] with Safari.

All of these configuration completely ignored deep labels in presentation MathML (Figure 18a).

We found exactly one working configuration for the case where presentation MathML is wrapped in a div node (Figure 18b), namely NVDA with Internet Explorer and without MathPlayer. Like this, the `aria-label` is read instead of the content of the div node.

However, this configuration does not work when span nodes are wrapped in div nodes (Figure 18c). In this case, only JAWS with Firefox reads the `aria-label`. We have tried to change the value of the `role` attribute for the latter case, but we could not find a setup that works both with presentation MathML and span nodes in div nodes.

As a workaround, we convert spotted quantity expressions to presentation MathML, wrap them in div nodes and attach meaningful aria labels for the expression. Only standalone quantity expressions can be process this way. Like this, we cannot handle quantity expressions that are part of formulae since this would imply to destroy the structure

[13] `https://www.nvaccess.org/`
[14] `http://www.freedomscientific.com/Products/Blindness/JAWS`
[15] `https://www.dessci.com/en/products/mathplayer/tech/accessibility.htm`
[16] `http://www.apple.com/accessibility/mac/vision/`

```
<?xml version="1.0" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22−rdf−syntax−ns"
  xmlns:mathml="http://www.w3.org/1998/Math/MathML">
  <rdf:Description rdf:about= ⟨⟨ Identifier ⟩⟩ >
    <rdf:XMLLiteral>
      ⟨⟨ Content MathML ⟩⟩
    </rdf:XMLLiteral>
  </rdf:Description>
</rdf:RDF>
```

Figure 19: The General Frame of Harvest Files for Tom Wiesing's Search Engine.

of the corresponding MathML term which is not desirable. With NVDA and Internet Explorer it is now possible to read, say, "three solar masses" and switching to JAWS allows to hear the encoding of the expression; here "three M sub circled dot operator".

6.3 Semantic Search

Tom Wiesing created a semantic search engine for quantity expressions [Wie15] which allows to also retrieve results in equivalent formats. For instance, we find results including "$90\ \mathrm{km/h}$" when searching for "$25\ \mathrm{m/s}$".

However, Wiesing's work is dependent on an external source of data. We thus post-process our results of the spotting of quantity expressions and use them to provide a corpus for Wiesing's search engine. This is only a simple transformation from the output format of the spotter (compare Figure 13) to the format displayed in Figure 19. It contains one rdf:Description entry per spotted quantity expressions. Here " ⟨⟨ Identifier ⟩⟩ " denotes a unique path to the document and the quantity expression therein. We use the value of the resource attribute from the old format for this case, since it exactly references the quantity expression in the original document. In the same way, " ⟨⟨ Content MathML ⟩⟩ " coincides with the corresponding expression from the old format. During the conversion, we omit quantity expressions with a negative score.

7 Future Work

This section lists suggestions for the further development of the presented system. The first two items focus on the use of additional natural language processing tools and on the detection of more quantity expressions. New technologies that can enhance this system are suggested in item 3 and 4. The last recommendation refers the runtime.

1. Part-of-speech tagging can be used during text tokenization (compare Section 5.1).

This can, for instance, be exploited to correctly detect, that “as” is a part of the text and not an abbreviation for “attoseconds”. This misclassification is currently ruled out quite naively by a scorer (compare Section 5.2.6) and could be improved by part-of-speech tagging.

2. This thesis restricts its attention to only a subset of all quantity expression. A possible extension would be to include more expressions, for instance, by detecting also quantity expressions with textual numbers, such as “five seconds”. For that, one could use a natural language processing tool to detect textual numbers. Range expressions, like “20 to 100 kilometers” and “$1 - 5\mathrm{m}^2$”, are also an important extension which not only involves the adaption of the detection schema but also of the annotation format.

3. The evaluation of machine learning technologies for semantics extraction might also prove useful. A good starting point for that might be the implementation of a scoring system based on machine learning. One can either evaluate it using the current results of the rule based approach or by allowing manual disambiguation for the users of the unit conversion service and use this data for training and testing.

4. For his declaration spotter [Sch16] Jan Frederik Schäfer developed a XML-based pattern language. Its use for the detection of quantity expressions can be evaluated which can either lead to the implementation of an additional spotter or to a reimplementation of the current spotter using the pattern language. An advantage of the pattern language is that the detection patterns are separated from the source code of the program and can easier be extended. However the pattern language does not yet support content MathML.

5. The current spotter and its scoring system are currently prototypical implementations. One can improve their speed in order to allow them to scale potentially on all of the arXMLiv documents.

8 Conclusion

In this thesis, we have described the extraction of meaning from STEM documents with a special focus on quantity expressions and units. We have presented a rule-based and modular implementation thereof on the arXMLiv corpus which delivers promising results. The architecture is easily extensible by additional detection methods and proved to work well with other spotters, such as Frederik Schäfer's declaration spotter. We have exploited the detection results to offer useful semantic services like the automatic conversion of units in scientific papers which supports users to stay more focused while

reading and helps them to prevent calculation errors by allowing them to convert quantity expressions by right clicking on them. With the semantic enhancement of screen reading programs, we have also contributed to the field of accessibility for STEM documents and hereby lowered the burden for visually impaired people to participate in the scientific discourse. Additionally, we converted the spotting results in such a way that they can be exploited by Tom Wiesing's semantic search engine for quantity expressions. It searches not only for the entered expression, but takes also equivalent forms into account, say it also finds 212 degree Fahrenheit when searching for 100 degree Celsius.

These applications demonstrate the additional benefit of semantic information compared to common purely syntactic data.

9 Acknowledgements

The author wants to thank his supervisor Prof. Dr. Michael Kohlhase for offering and supervising an interesting topic, for many helpful and inspiring discussions and especially for his open-door policy which lead to an inviting working atmosphere. Thanks goes also to Tom Wiesing for his support with various technical questions and to Dennis Müller for widening the author's musical horizon.

References

[AF92] Mauro Anselmino and Stefano Forte. Small angle polarization in high energy p–p scattering through nonperturbative chiral symmetry breaking. 1992, arXiv:hep-ph/9211221.

[Ast13] Astropy Community et al. Astropy: A community Python package for astronomy. *Astronomy & Astrophysics*, Volume 558:A33, October 2013, 1307.6212. http://www.astropy.org.

[BBGHK92] Paul Baillon, Alain Bouquet, Yannick Giraud-Héraud, and Jean Kaplan. Detection of brown dwarfs by the micro-lensing of unresolved stars. 1992, arXiv:astro-ph/9211002.

[Boa99] Mishap Investigation Board. Mars Climate Orbiter – Phase I Report, November 1999. ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf (Visited on 27.2.2017).

[CC14] James Craig and Michael Cooper. Accessible rich internet applications (WAI-aria) 1.0. W3C recommendation, W3C, March 2014. http://www.w3.org/TR/2014/REC-wai-aria-20140320/.

[CGL11] Mihai Cîrlănaru, Deyan Ginev, and Christoph Lange. Authoring and publishing units and quantities in semantic documents. In *Extended Semantic Web Conference*, pages 202–216, Heraklion, Greece, 2011. Springer.

[CPP92] S. Caracciolo, G. Parisi, and A. Pelissetto. Random walks with long-range self-repulsion on proper time. 1992, arXiv:hep-lat/9211013.

[DGK+14] Mircea Alex Dumitru, Deyan Ginev, Michael Kohlhase, Vlad Merticariu, Stefan Mirea, and Tom Wiesing. System Description: KAT an Annotation Tool for STEM Documents. In *Conference on Intelligent Computer Mathematics (CICM)*, Coimbra, Portugal, 2014.

[DH03] Stan Devitt and Douglas Harder. Units in MathML. W3C note, W3C, November 2003. http://www.w3.org/TR/2003/NOTE-mathml-units-20031110/.

[FGMV92] M. De Francia, G. Goya, R. C. Mercader, and H. Vucetich. Mößbauer null redshift experiment ii, 1992, arXiv:gr-qc/9211005.

[Fos09] Marcus Foster. Disambiguating the SI notation would guarantee its correct parsing. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 465, pages 1227–1229. The Royal Society, 2009.

[fWM06] International Committee for Weights and Measures. The International System of Units (SI). 8th edition, 2006.

[GG92] Graciela Gelmini and Marcelo Gleiser. Kinetics of sub-critical bubbles and the electroweak transition. 1992, arXiv:hep-ph/9211303.

[GNR92] Michael Gronau, Alex Nippe, and Jonathan L. Rosner. Method for flavor tagging in neutral b meson decays. 1992, arXiv:hep-ph/9211311.

[GS] Deyan Ginev and Jan Frederik Schäfer. LLaMaPUn - common language and mathematics processing algorithms, in Rust. URL: https://github.com/KWARC/LLaMaPUn.

[Hau01] Heinz-Dieter Haustein. *Weltchronik des Messens*. de Gruyter, 2001.

[HK92] David Hochberg and Thomas W. Kephart. Wormhole cosmology and the horizon problem. 1992, arXiv:gr-qc/9211006.

[HKR17] Daniel Hajas, Michael Kohlhase, and Ulrich Rabenstein. MathML Accessibility via Aria Labels: a Practice Test. Published as Note for the MathML Association. Soon available at http://mathml-association.org/_-posts/2017-05-qe-for-screen-readers.html, Mai 2017.

[Iwa98] Aiichi Iwazaki. Axionic boson stars in magnetized conducting media and monochromatic radiations, 1998, arXiv:hep-ph/9807232.

[KDKR00] Ulrich Kolb, Melvyn Davies, Andrew King, and Hans Ritter. The violent past of cygnus x-2. 2000, arXiv:astro-ph/0003441.

[Koh08] Michael Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[LK98] Bao-An Li and C. M. Ko. Probing the softest region of nuclear equation of state. *Phys.Rev.C58:1382-1384*, 1998, arXiv:nucl-th/9807088.

[MCI14] Robert R Miner, David Carlisle, and Patrick D F Ion. Mathematical markup language (MathML) version 3.0 2nd edition. W3C recommendation, W3C, April 2014. http://www.w3.org/TR/2014/REC-MathML3-20140410/.

[Mil16] Bruce R. Miller. LaTeXML – the manual – a LaTeX to XML/HTML/-MathML converter; version 0.8.2, December 2016.

[MS98] I. V. Moskalenko and A. W. Strong. Deciphering diffuse galactic continuum gamma rays. *Proc. 16th Europ. Cosmic Ray Symp. (Alcala de Henares), ed. J.Medina, p. 347-350 (1998)*, 1998, arXiv:astro-ph/9807288.

[PHL92] J. Pantaleone, A. Halprin, and C. N. Leung. Neutrino mixing due to a violation of the equivalence principle. 1992, arXiv:hep-ph/9211214.

[PS92] Tsvi Piran and Amotz Shemi. Fireballs in the galactic halo and γ-ray bursts. 1992, arXiv:astro-ph/9211009.

[RK13] Florian Rabe and Michael Kohlhase. A scalable module system. *Information & Computation*, 0(230):1–54, 2013.

[Ron17] Armin Ronacher. Flask – a web development framework for python. http://flask.pocoo.org/, visitied 9.2.2017. Version 0.12.

[Ros92] Leszek Roszkowski. On the cobe discovery - for pedestrians. 1992, arXiv:hep-ph/9211201.

[RSM+98] H. Ruhl, Y. Sentoku, K. Mima, K. A. Tanaka, and R. Kodama. Collimated electron jets by intense laser beam-plasma surface interaction under oblique incidence. 1998, arXiv:physics/9807021.

[Sch16] Jan Frederik Schäfer. *Declaration Spotting in Mathematical Documents*. Bachelor Thesis in Computer Science, Jacobs University Bremen, 2016.

[SD08] Jonathan Stratford and James Davenport. Unit knowledge management. In *International Conference on Intelligent Computer Mathematics*, pages 382–397. Springer, 2008.

[Ser98] Sergei A. Sergeenkov. Magnetic field induced charging effects in josephson junction arrays. 1998, arXiv:cond-mat/9807111.

[She15] Stiv Sherko. *Extracting quantity and unit semantics from technical documents*. Bachelor Thesis in Computer Science, Jacobs University Bremen, 2015.

[SKG+10] Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce Miller. Transforming large collections of scientific publications to XML. *Mathematics in Computer Science*, 3(3):299–307, 2010.

[SOD+04] S.Buswell, O.Caprotti, D.P.Carlisle, M.C.Dewar, M.Gaëtano, and M.Kohlhase. The openmath standard. Technical report, The OpenMath Society, 2004.

[SR14] Guus Schreiber and Yves Raimond. RDF 1.1 primer. W3C working group note. http://www.w3.org/TR/rdf-primer, 2014.

[SW16] Felix Schmoll and Tom Wiesing. KAT: Enabling the Semantification of STEM Documents. Conference on Intelligent Computer Mathematics (CICM), 2016.

[TMM98] Wayne A. Tokaruk, T. C. A. Molteno, and Stephen W. Morris. Benard-marangoni convection in two layered liquids, 1998, arXiv:patt-sol/9807001.

[TTGV98] Yuhai Tu, J. Tersoff, G. Grinstein, and David Vanderbilt. Properties of a continuous-random-network model for amorphous systems. 1998, arXiv:cond-mat/9807211.

[vA00] U. Fritze v. Alvensleben. Modelling tools: Population and evolutionary synthesis, 2000, arXiv:astro-ph/0009290.

[VBR98] D. Vandembroucq, A. C. Boccara, and S. Roux. Breakdown patterns in branly's coheror. *Journal de Physique III,7: (2) 303-310, (1997)*, 1998, arXiv:cond-mat/9807012.

[WH] Joseph Wright and Marcel Heldoorn. siunits – International System of Units. LaTeX package. URL: https://www.ctan.org/pkg/siunits.

[Wie15] Tom Wiesing. *Semantic search for quantity expressions*. Bachelor Thesis in Computer Science, Jacobs University Bremen, 2015.

[Wie17] Tom Wiesing. Jobad – JavaScript API for OMDoc-based Active Documents. http://kwarc.github.io/jobad/, visited 9.2.2017. Version 3.2.2.

[WK17] Tom Wiesing and Michael Kohlhase. Report on OpenDreamKit deliverable D4.9 – in-place computation in active documents (context/computation). Technical report, FAU Erlangen-Nürnberg, 2017. https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP4/D4.9/report-final.pdf.

[YLS+98] Jongsoo Yoon, C. C. Li, D. Shahar, D. C. Tsui, and M. Shayegan. Wigner crystallization and metal-insulator transition of two-dimensional holes in gaas/algaas at b=0. 1998, arXiv:cond-mat/9807235.

[YS00] Shoji Yamamoto and Toru Sakai. Multi-plateau magnetization curves of one-dimensional heisenberg ferrimagnets. 2000, arXiv:cond-mat/0005248.

[ZHM+98] S. Zaggia, I. Hook, R. Mendez, L. da Costa, L. F. Olsen, M. Nonino, A. Wicenec, C. Benoist, E. Bertin, E. Deul, T. Erben, M. D. Guarnieri, R. Hook, I. Prandoni, M. Scodeggio, R. Slijkhuis, and R. Wichmann. ESO Imaging Survey IV. Exploring the EIS Multicolor Data, 1998, arXiv:astro-ph/9807152.

Curriculum Vitæ

Personal Details

Ulrich Rabenstein
Reinschartenweg 1A
91056 Erlangen
ulrich.rabenstein@fau.de

Education

08/2015 – 12/2015	Study Aboard at **Indian Institute of Technology Madras (IITM)**
From 04/2015	**Master of Science** in **Computer Science** **Friedrich-Alexander-University (FAU)**, Erlangen (Germany) Minor Subject: Mathematics
10/2011 – 03/2015	**Bachelor of Science** in **Computer Science** **Friedrich-Alexander-University (FAU)**, Erlangen (Germany) Minor Subject: Mathematics Thesis: "Ontology-supported classification of ad hoc disclosures"
09/2003 - 07/2011	**Higher Education Entrance Qualification** **Albert Schweitzer-Gymnasium** Erlangen (Germany)

Extra Curricular Activities

04/2012 – 08/2015	**Teaching Assistant for Computer Science Courses**
04/2016 – 03/2017	Preparation, marking of weekly homework and teaching of the following courses: Theory of Programming, Foundations of Logic in Computer Science and Artificial Intelligence
10/2009 – 03/2010	**Early-study project of the FAU** Permission to attend regular university lectures as a high-school student

Scholarships and Awards

since 10/2013	Member of the **Max Weber-Program** Part of the Elite Network of Bavaria which aims at highly-gifted students
since 09/2012	Member of the **Leonardo-Kolleg** of the FAU The Leonardo-Kolleg is an interdisciplinary institution for the best students of the FAU