Parsing, Representing and Transforming Units of Measure

Basel Shbita, Arunkumar Rajendran, Jay Pujara, and Craig A. Knoblock

Information Sciences Institute, University of Southern California

Abstract

Data-intensive models have become critical to understanding the world. In order to reuse or combine datasets to support modeling, scientists must select, understand, and align them manually, a laborious process that requires understanding different domains and formats. To assist the modeling process, we present an unsupervised approach that identifies units in source data and provides a corresponding semantic representation. Then, we provide a method that enables scientists to perform data transformations, such as unit conversions, which are commonly necessary in modeling world systems. Our preliminary results demonstrate that our method can be used to automatically capture and transform units over spreadsheets achieving an F1-score of 0.48 in unit detection and parsing, and an accuracy of 62% in the semantic representation and transformation.

1 Introduction

Data normalization is a difficult task which occupies as much as 80% of the total data analysis time [4]. One reason data normalization is challenging is because the data are drawn from diverse domains and expressed in a variety of potentially incompatible formats. In this paper we focus on an important task in data normalization, the identification of units that are associated with the data. Unit identification is challenging because it requires having some domain knowledge about the process that produced the data. For example, a scientist who wishes to integrate a hydrology model which associates groundwater-withdrawal units in gallons with an agricultural watering model that presumes input units in liters would need to manipulate the data to allow the composition of the data in the required units.

Today, scientists' approach to handling incompatibility in units is usually via a one-time transformation. If they associate data with the incorrect units, they might be unable to use the dataset and resort to ad-hoc strategies that harm the transparency and reproducibility of the results. The diversity in disciplines, domains and conventions in different regions around the world poses an additional challenge to this problem. Considering the vast amount of data presently used in modeling infrastructure this problem becomes intractable and tedious and is often susceptible to human error.

One potential solution to the difficulties of manual unit detection and conversion is the use of automated systems. Unfortunately, automating unit detection is a difficult task. Frequently, units appear in files within datasets in a textual representations that is not easily recognized. These text strings usually contain a formula-like form: unit abbreviations, exponents, and additional elements which represent an atomic or a compound unit. An atomic unit is a single unit symbol which may be modified by additional elements such as exponents or prefixes (i.e. `'GHz'`, `'m^2'`) whereas compound units are composed from two or more atomic units with some relationship between them (i.e. `'A/cm^2'`). Textual representation is not sufficient and does not carry any

semantic or dimensional meaning and may require additional investigation if one needs to perform transformations and data alignment. Additionally, lexical conventions such as capitalizations are very important when representing units to resolve semantic ambiguities. For example: the text string `'S'` would stand for `Siemens` (electric conductance unit) where `'s'` would represent a unit of type `second` (time unit). Supporting SI (International System of Units) prefixes complicates the problem even more, adding an additional layer of combinatorial complexity.

Previous research [9, 11, 13] has developed approaches to allow easier data representation, cleaning and transformation. However, previous research still depends on human interaction in early data processing stages. Published ontologies are a beneficial resource that can be used for a faster process of data cleaning. NASA has published an ontology called QUDT (Quantity, Unit, Dimension and Type) [2] which defines the base classes and attributes used for modeling physical quantities, units of measure, and their dimensions. The main drawback of existing unit ontologies is the amount of effort required to represent datasets, there is no readily an available tool to generally link data to the units used in the datasets. Thus it is intuitive to extend and integrate existing ontologies into a framework to enable an automatic and fast process of data understanding, normalization and transformation.

Significance In this paper, we present an approach to automatically detect, parse, normalize and represent compound and atomic units of measure in a data source. Using the semantic representation we are able to support scientists by providing the ability to perform unit conversions that are less prone to error. To demonstrate our idea, we implemented a prototype system, called CCUT (**C**anonicalization **C**ompound **U**nit Representation and **T**ransformation), which uses grammar tools to automatically parse the different components in a unit found in textual data in files and map them to elements of a standard ontology defined by NASA [2] to form a structured semantic output. The output depicts the different relationships, attributes and semantics of units and allows users to safely perform a transformation between units. Our method was tested on spreadsheets and can be easily deployed over a range of quantitative data resources and thus accelerate and improve the modeling process in any scientific domain.

2 Problem Definition

The problem we address is given arbitrary dataset files, in some common structured format (i.e. `csv`, `json`, `xls`), find textual explicit mentions of measurement units and map them to the QUDT units ontology [2]. Using the semantic representation we want to produce a structured standard ontologized output that can be easily interpreted by humans and machines. Then, we want to leverage the ontology to assist users in common modeling transformations such as complex compound units conversion.

As an example, consider the compound unit present in cell `B8`, `'A/cm^2'` (marked in a red box), in a spreadsheet file as shown in figure 1a. A conventional semantic representation for this string is shown in figure 1b where each base unit is an element of the list labeled with `qudt:_hasPart_` and its URI is introduced within the key labeled `qudt:quantity`. Besides the unit ontology, each part has additional attributes such as `UNK:prefix` and `UNK:exponent`. Additionally, `qudt:abbreviation` and `qudt:hasDimension` describe the properties of each element in the unit and are computed for the overall normalized compound unit, and the attribute labeled `UNK:prefix_conversion_multiplier` can be easily utilized for a unit conversion service.

	A	B	C
2	Year in Production	Units	2003
3	Technology Generation		
4	Physical Lgate (Low-Standby-Power)	nm	75
5	EOT (Equivalent Oxide Thickness)	A	22
6	Gate Poly Depletion & Inversion-Layer Thickness	A	8
7	Inversion Gate Dielectric Thickness Value	A	30
8	Maximum Gate Leakage Limit	A/cm^2	4.4E-03
9	Power Supply Voltage	V	1.2
10	Saturation Threshold Voltage	V	0.50

READ ME | PIDS LSTP Table | Worksheet LSTP | Plots | +

(a) a compound unit in `xls` spreadsheet

```
qudt:_hasPart_: [
    {
        qudt:hasDimension: "I",
        qudt:quantity: "http://data.nasa.gov/qudt/owl/unit#Ampere",
        qudt:symbol: "A"
    },
    {
        UNK:exponent: "-2",
        UNK:prefix: "http://data.nasa.gov/qudt/owl/unit#Centi",
        UNK:prefix_conversion_multiplier: 0.01,
        qudt:hasDimension: "L",
        qudt:quantity: "http://data.nasa.gov/qudt/owl/unit#Meter",
        qudt:symbol: "cm"
    }
],
qudt:abbreviation: "A cm-2",
qudt:hasDimension: "L-2 I"
```

(b) partial JSON representation for `'A/cm^2'`

Figure 1: An example of a detected compound unit and its representation

3 Our Approach

Our approach tackles three core problems, these are illustrated in figure 2 and can be summarized as follows:

1. Identify and parse the individual prefixes, atomic units, their exponents and multipliers which compose a string of a compound unit. As shown in transition 1 in figure 2 (section 3.1)
2. Map each atomic unit to its correct ontology in the schema. As shown in transition 2 in figure 2 (section 3.2)
3. Compute the dimension of the compound unit and construct a normalized representation of the compound unit with attributes that are required for transformation. As shown in transition 3 in figure 2 (section 3.3)

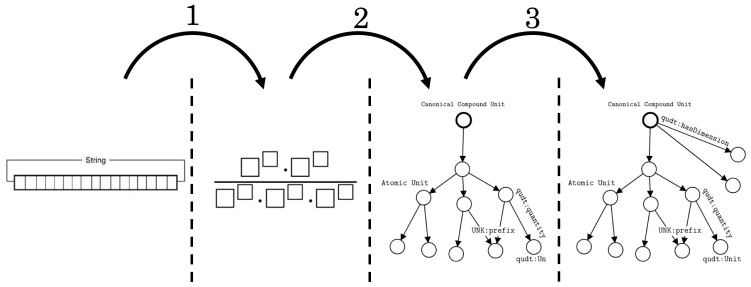

Figure 2: flowchart describing our approach

3.1 Parsing

Given a string that contains units, we want to generate a structured form which represents that unit and provide a set of relationships between its components. In this stage, there are a few problems we need to solve.

First, in the QUDT ontology there are some units that are missing a complete definition and have partial dimension information. In order to address this issue, we created a closed set of meaningful units and defined base dimension classes which were derived from the ontology.

Second, atomic units may be accompanied with unit prefixes (such as `'kilo'`, `'micro'` or `'mega'`), and may have different variants that are used to symbolize the same prefix. For example: a prefix of type micro (10^{-6}) may show up in text as `'micro'`, `'mu'` or `'μ'`. Additionally, there are some

cases in which our parser can run into ambiguous cases. For example: the unit string `'min'` can be interpreted as `minute` (time unit) or as a combination of the prefix `'m'` as in `'milli'` (10^{-3}) and the unit `inch` that is abbreviated as `'in'` (length unit). To handle these issues we manually encoded 20 prefix classes that characterize unit prefixes in addition to their well-known different variants, and implemented an iterative joint matching algorithm for {*prefix, unit*} pairs. The algorithm gives higher confidence to a mapping of a single atomic unit and iteratively keeps searching for a joint match in case of a failure in the first attempt until the two elements are mapped to a well-defined terms correspondingly.

Last, we must consider the different characters which represent the relations between the atomic units (i.e. fractions, exponents, etc) in a string. To tackle this issues, we implemented a well defined grammar, that is, a specification for how to read a language of compound and atomic units. We used Arpeggio [5] as a grammar parser. Arpeggio is a recursive descent parser with backtracking and memoization that is based on PEG (Parsing Expression Grammar) formalism. Table 1 depicts some grammar rules we have defined.

Table 1: Some grammar rules

Character	**Rule**
`/`	Split to numerator & denominator
`^`	Interpret as an exponent
`( and )`	Split to canonical form
`-`	Interpret as a negative sign
`.`	Interpret as a floating point

3.2 Structured Unit Representation

While the structured representation is more informative, it still does not capture a semantic meaning. In order to provide the semantic meaning we have to rely on ontologies that define universal conventions for units. The QUDT ontology defines a unit symbol (`qudt:symbol`) that is associated with a unit ontology instance (`qudt:Quantity`) that has a unique URI. This unit instance is associated with the symbol using a relation of type `qudt:QuantityKind`. This relation is an instance in itself and is associated to some dimension (`qudt:Dimension`). Given the above, we are able to map an atomic unit string to its semantic type and find its dimensions. Utilizing the additional grammar output terms (i.e. exponents) enables us to normalize the compound unit and present an interpretable representation.

As mentioned earlier in section 3.1 we implemented a tight integration between the prefix classes and the unit classes. This provides an efficient solution and insures a proper semantic representation. Since each element is linked to a Uniform Resource Identifier (URI) and is uniquely identified by it, our solution provides a cost-free representation.

3.3 Transforming Compound and Atomic Units

In this stage, our goal is to enable arbitrary transformations between units using our semantic structured representation which we have already generated and mapped. A dimensions-based approach encoded in the QUDT ontology relates each unit to a system of base units using numeric factors. For example, any measurement of length can be expressed as a number multiplied by the unit `meter` (the SI base for the length dimension). Given that, and the set of exponents, prefixes and multipliers derived from the grammar and applied over a set of fundamental dimensions, we are able to generate the required calculation to perform unit conversions of same dimension. We use the conversion multiplier and offset, which are multiplied and added to quantities to convert from the current unit to the corresponding SI unit. So, if m_1, o_1 and m_2, o_2 are the conversion multipliers

and offsets for U_1 and U_2 respectively, m_{p1}, o_{p1} and m_{p2}, o_{p2} are the conversion multipliers and offsets of their prefixes respectively, and α and β are their exponents respectively, then the proper conversion is according to Equation 1. When a conversion is desired between two SI units, their offsets are equal to zero by definition and therefore we get a simplified form as in Equation 2. Thus, we can offer a transformation service to users via an additional service endpoint in the system to enforce correct and safe conversions.

$$U_2 = \frac{(\frac{(U_1 \cdot m_{p1}^{\alpha} + o_{p1}) \cdot m_1^{\alpha} + o_1 - o_2}{m_2^{\beta}}) - o_{p2}}{m_{p2}^{\beta}} \tag{1}$$

$$U_2 = U_1 \frac{(m_1 \cdot m_{p1})^{\alpha}}{(m_2 \cdot m_{p2})^{\beta}} \tag{2}$$

4 Evaluation

We have employed the EUSES spreadsheet corpus [6] and implemented an `xls` file reader for the purpose of testing against our API endpoints of the CCUT system. The corpus contains 1345 files and an overall number of 5891 spreadsheets collected from different sources. We randomly sampled 30 files that included 112 spreadsheets. Explicit unit strings were present only in 31 spreadsheets. We used the sampled set as a testing set and for which we created a validation file to compare the dimensional analysis and the different elements to (including their URIs and normalized form). In the given set 267 compound units (and a total of 530 atomic units) were observed and manually annotated in the validation set. In addition, we defined an in-code python dictionary to keep track of the observed compound unit strings which have the same normalized dimension in order to test the accuracy of our transformation service between each pair of units in the same dimension.

CCUT detected and parsed a total of 882 atomic elements (328 true positives; 554 false positives) and misdetected 150 elements (false negatives), generating a total precision of 37.2%, a recall of 68.6% and an overall F1-score of 0.48. Out of the valid compound units (true positives) 62.12% were normalized correctly (overall dimension representation was precise). A total of 11 distinct (compound) dimension groups were identified, out of which 5 groups included more than a single distinct string representation, providing us a total of 42 transformation test cases of pairs which had a 100% accuracy in the transformation calculation. This is normally what we expected in the transformation tests since these compound units were accurately captured and represented.

We examined cases where we were unable to map the correct unit and discovered that in some cases the units were detected in irrelevant text. Some of the detected units were mistakenly extracted from abbreviations for other entities or organizations. These issues caused an overall low precision score. In other cases there is an ambiguity because the same abbreviation is used in multiple units (i.e. `'L'` stands both for `liter`, a volume unit, and `lambert`, a luminance unit). Our system does not currently support a principled mechanism for using context to make a more intelligent prediction. In some cases we simply did not have the correct unit in our knowledge base which affected our recall.

These early results are encouraging and are expected to be increased as described in section 7 where we address some of these issues.

5 Related Work

Chambers and Erwing [3] presented a reasoning system for inferring dimension information in spreadsheets. Their system aims to check the consistency of formulas and detect errors in spreadsheets. Abraham and Erwig [1] developed a VBA-based add-on for excel which enables the detection of errors and which is based on a set of rules for automatic header inference. Although these systems do not require any user intervention for their operation, they do not offer a semantic representation or any conversion services as we present in our work.

Existing frameworks such as VizieR [10], the yt Project [13] and Measurement-units-in-R [11] attempt to deal with the problem of unit representation and conversion by giving users the option to enforce a unit of measure for a given fixed set of data. This enables one to add, subtract, multiply, and divide using quantities and dimensional arrays. When used in expressions, some of these platforms automatically convert units, and simplify them when possible. Measurement-units-in-R gives the user the flexibility to expand beyond predefined units but it requires an initial user definition and understanding of data. Our work differs by providing users the ability to capture the semantics behind units given a string without any initial definitions since it uses a standard and structured representation defined by NASA.

The NASA QUDT ontology [2] is being adopted in many scientific research projects [8, 12]. Of the established and well-governed unit of measure ontology options, QUDT is well-aligned with our understanding of the relationships between measurements and units of measure.

6 Conclusion

With increasing volume and variety of data, we require better techniques and tools to enable end-users and non domain experts to easily understand, analyze and transform quantitative data. Existing techniques rely on human interaction and some expert domain knowledge in some cases. In this paper, we presented a baseline unsupervised approach to automatically captures a semantic representation and normalization for units of measure and offer unit conversions to end-users. The evaluation of the system has demonstrated early results and leaves room for improvement but can be beneficial for scientists to perform a fast process of data analysis.

7 Discussion and Future Work

Our approach has several limitations, including the inability to use context to disambiguate units and a naive approach to match text. In future work we plan to use context to increase the precision of the system. Examples of useful context include the co-occurrence of units within a domain and locations in datasets (e.g., column headers) that are more likely to contain units. Moreover, we want to use machine learning techniques to distinguish different domains to help disambiguate units.

Our system is limited by our knowledge base of types, which affects our recall. We will address this issue by expanding our knowledge base with more types of units and provide users with an interface in which they can extend the base ontology by adding new units and transformation attributes.

We also believe that a combination of unit detection with NLP approaches for contextual text could strengthen the reasoning of the system and allow detection of units which are not mentioned explicitly in text. Beyond unit detection, we plan to extend our approach to solve the broader

problem of table understanding, allowing us to detect specific variables contained in datasets along with temporal and geospatial scoping.

8 Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) with award W911NF-18-1-0027. We thank our colleague Dr. Daniel Garijo who provided insights and expertise that greatly assisted the research.

9 Resources

This work has been developed as part of the Model INTegration (MINT) project [7] which envisions the development of a framework for resolving semantic, spatio-temporal, and execution mismatches in model integration. This work is under ongoing development; Releases of the software and tools will appear on the MINT project website[1] in the future.

References

[1] Robin Abraham and Martin Erwig. Header and unit inference for spreadsheets through spatial analyses. In *IEEE Symposium on Visual Languages and Human Centric Computing*, 2004.

[2] Stuart Chalk, Ralph Hodgson, and Steve Ray. Qudt toolkit: Development of framework to allow management of digital scientific units. In *ABSTRACTS OF PAPERS OF THE AMERICAN CHEMICAL SOCIETY*, volume 253, 2017.

[3] Chris Chambers and Martin Erwig. Dimension inference in spreadsheets. In *IEEE Symposium on Visual Languages and Human-Centric Computing*.

[4] T Dasu and T Johnson. Exploratory data mining and data cleaning: An overview. *Exploratory data mining and data cleaning*.

[5] Igor Dejanović, Gordana Milosavljević, and Renata Vaderna. Arpeggio: A flexible peg parser for python. *Knowledge-Based Systems*, 95.

[6] Marc Fisher and Gregg Rothermel. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *ACM SIGSOFT Software Engineering Notes*, volume 30.

[7] Yolanda Gil, Kelly Cobourn, Ewa Deelman, Chris Duffy, Rafael Ferreira da Silva, Armen Kemanian, Craig Knoblock, Vipin Kumar, Scott Peckham, Lucas Carvalho, et al. Mint: model integration through knowledge-powered data and process composition. In *9th International Congress on Environmental Modelling and Software*, 2018.

[8] Mark Hennessy, Chris Oentojo, and Steven Ray. A framework and ontology for mobile sensor platforms in home health management. In *1st International Workshop on the Engineering of Mobile-Enabled Systems (MOBS)*, 2013.

[9] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. Activeclean: interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12).

[10] François Ochsenbein, Patricia Bauer, and James Marcout. The vizier database of astronomical catalogues. *Astronomy and Astrophysics Supplement Series*, 143(1).

[11] Edzer Pebesma, Thomas Mailund, and James Hiebert. Measurement units in r. *The R Journal*, 8(2).

[12] BA Simons, Jonathan Yu, and SJD Cox. Defininga water quality vocabulary using qudt and chebi. In *Proceedings of the 20th International Congress on Modelling and Simulation*.

[13] Matthew J Turk, Britton D Smith, Jeffrey S Oishi, Stephen Skory, Samuel W Skillman, Tom Abel,and Michael L Norman. yt: A multi-code analysis toolkit for astrophysical simulation data. *The Astrophysical Journal Supplement Series*, 192(1).

[1] `http://mint-project.info/`