Converting and Annotating Quantitative Data Tables

Mark van Assem[1], Hajo Rijgersberg[3], Mari Wigham[2,3] and Jan Top[1,2,3]

[1] VU University Amsterdam, The Netherlands
mark@cs.vu.nl
[2] Top Institute Food and Nutrition, The Netherlands
[3] Wageningen University and Research Centre, The Netherlands
first.last@wur.nl

Abstract. Companies, governmental agencies and scientists produce a large amount of quantitative (research) data, consisting of measurements ranging from e.g. the surface temperatures of an ocean to the viscosity of a sample of mayonnaise. Such measurements are stored in tables in e.g. spreadsheet files and research reports. To integrate and reuse such data, it is necessary to have a semantic description of the data. However, the notation used is often ambiguous, making automatic interpretation and conversion to RDF or other suitable format difficult. For example, the table header cell "f (Hz)" refers to frequency measured in Hertz, but the symbol "f" can also refer to the unit farad or the quantities force or luminous flux. Current annotation tools for this task either work on less ambiguous data or perform a more limited task. We introduce new disambiguation strategies based on an ontology, which allows to improve performance on "sloppy" datasets not yet targeted by existing systems.

1 Introduction

In this paper we study how to convert and annotate unstructured, "raw" quantitative data stored in tables into a semantic representation in RDF(S). Quantitative data are found in diverse sources, such as scientific papers, spreadsheets in company databases and governmental agencies' reports. The data consist of observations such as the heart rate of a patient measured in beats per minute, the viscosity of a sample of mayonnaise in pascal second, or the income of households in dollars in the US. Usually the tables consist of a header row that indicates which quantities and units are being measured and which objects; e.g. *Sample Nr. / Fat % / Visc. (Pa.s)*. Each content row then contains the values of one actual measurement.

Current reuse and integration of such data is not optimal, because a semantic description is not available. Researchers tend to write their data down in a "sloppy" way, because it is not anticipated that the data will ever be reused. This causes data to be "lost" and experiments to be needlessly repeated. To enable integration of data from different tables with each other, a complete description of all quantities and units in the table is necessary; annotation with

a few key concepts does not suffice. There are two main reasons why it is difficult to automatically convert the original data to a semantic description. Firstly, humans use different syntax for expressing quantities and units (e.g. separating the quantity from the unit with either brackets or a space). Secondly, the symbols and abbreviations used are highly ambiguous. For example, the symbol "g" can refer to at least ten different quantities and units.

This problem is not tackled by existing systems for conversion of tabular data to RDF, such as XLWrap [8]. These rely on a mapping specification constructed by a human analyst that is specific to the header of one table. Creating such a mapping is labour-intensive, especially if there are many differently structured tables involved. This is the case in government repositories such as Data.gov [4], and repositories of research departments of companies such as Unilever and DSM (from experience we know these contain thousands of different tables).

A solution is to include an automated annotation system into the conversion tool, as proposed by [9]. However, such an annotation system needs to tackle the ambiguity problem if it is to be succesfully used in the domain of quantities and units. We know of two existing annotation systems that target the domain of quantities and units [7, 1], and our research can be seen as a continuation of these efforts. The results of these systems are good (over 90% F-measure), but they target "clean" datasets such as patent specifications, or focus on part of the total problem, such as detecting units only. In our work we focus on datasets with a high degree of ambiguity and attempt to detect quantities and units (including compound units).

The main contribution of our work is to show how ontology-based disambiguation can be used succesfully in several ways. Firstly, ambiguous quantity and unit symbols can be disambiguated by checking which of the candidate units/quantities are explicitly related to each other in the ontology. Secondly, ambiguous unit symbols may refer to units in specific application areas (e.g. nautical mile) or generic ones (meter). Some concepts act as indicators for a particular area (e.g. the unit nautical mile for "shipping"). After the area is identified by the presence or absence of indicators, we can disambiguate unit symbols. Thirdly, ambiguous compound unit expressions such as g/l can refer to gram per liter or gauss per liter. Only the former makes sense, as the ontology allows to derive that it refers to the quantity density, while the latter matches no known quantity. We show the benefits of ontology-based disambiguation by measuring precision and recall on two datasets and comparing with the performance achieved without these techniques. The datasets concerned are: (1) tables from the Top Institute Food and Nutrition; and (2) diverse scientific/academic tables downloaded from the Web.

The structure of this paper is as follows. We first present a detailed description of the problem, followed by related work (Sections 2 and 3). In Section 4 the datasets and ontology used in our experiment are described. Our approach is given in Section 5, which we evaluate in Section 6. We conclude with a discussion in Section 7.

2 Problem Description

Correct annotation of documents is faced with similar problems across many domains, including homonymy (a cause of low precision) and synonymy (a cause of low recall if the synonym is not known to the system). Below we discuss in what way these problems play a role in this domain.

Homonymy is caused in several ways. Firstly, it is not known beforehand whether cells contain a quantity (e.g. frequency), a unit (e.g. hertz), or both (e.g. f (Hz)). Secondly, homonymous symbols such as f are used, which can refer to quantities (frequency, force), units (farad) and prefixes (femto). The cell ms-1 might stand for either reciprocal millisecond or for meter per second (in the latter case m and s-1 should have been separated by a multiplication sign or space). This problem is aggravated because people often do not use official casing (e.g. f for force instead of the official F).

There are several types of synonymy involved in this domain: partial names (current for electric current), abbreviations (e.g. freq, Deg. C), plural forms (meters) and contractions (ms-1 instead of the correct form m s-1 for meter per second). Another type of synonym occurs when a quantity is prefixed with a term that describes the situation in more detail ("finalDiameter", "start_time", "mouthTemperature"). People also use colloquial names for quantities which overlap with other quantity names (i.e. confuse them). Two examples are weight (kg) and speed (1/s). The former should be mass (weight is measured in newton), the latter should be frequency.

A problem that is specific to this domain is the correct detection of compound units. The system has to detect the right compound unit instead of returning the units of which the unit is composed. For example, it should detect that km/h means kilometer per hour, instead of returning the units kilometer and hour separately (these should be counted as wrong results). This problem is aggravated by the fact that the number of compound units is virtually unlimited. For example, the quantity speed can be expressed in km/h, mm/picosecond, mile/year, etcetera. It is impractical to list them explicitly in an ontology. The interpretation of compound expressions is also difficult because of homonymy: g/l might stand for gram per liter or gauss per liter. The annotation process must somehow detect that gram per liter is the right compound unit (gauss per liter is not used), without gram per liter being present in the ontology. Returning gram, gauss and liter means returning three wrong results.

For correct detection of compound expressions, syntactic variations have to be taken into account (multiplication signs, brackets, etcetera). Compound expressions are also sometimes combined with substances, e.g. Conc. (g sugar/l water). Taken together this means a flexible matching process is needed instead of a strict grammar parser.

Particular to this domain is also that people tend to write down a quantity that is too generic or specific for the situation. For example, velocity (m/s) is too specific if the table contains scalar values only. The quantity velocity is only appropriate when a vector or a direction is indicated (e.g. "180 km/h north"). The other way around, the cell viscosity (stokes) should not be annotated with

viscosity. The specific quantity kinematic viscosity (measured in stokes), is more precise. These "underspecifications" need to be corrected before successful data integration can take place.

3 Related Work

Annotation systems for quantitative data. As far as we know there are two existing systems that focus on automated annotation of tables with quantities and units. The system of [7] annotates table headers with both quantities and units, focusing on the biological domain (it contains generic physical quantities such as temperature and domain-specific ones such as colony count). The names and symbols are matched against their own ontology of 18 quantities with their associated unit symbols. Table headers and labels in the ontology are first lemmatized, turned into a vector space model, and compared using cosine similarity. Weights for terms are fixed beforehand: tokens that appear in the ontology get a weight of 1, stopwords and single letter tokens get weight zero. The advantage of this technique is that the order of tokens within terms is not important, so that "celsius temperature" matches "temperature celsius". This technique does not take abbreviations and spelling errors into account (e.g. "temp cels" will not match).

[1] present a system based on GATE/ANNIE for annotating measurements found in patent specifications (natural language documents). Symbols found in the documents are first tagged as possible unit matches using a flat list[4]. Domain-specific pattern matching rules then disambiguate the results, using the actual text plus detected types as input. For example, if a number is followed by letter(s) that match a unit symbol (e.g. 100 g), then the letter(s) are classified as a unit. It uses a similar rule to detect that 40-50mph refers to a range of numbers. Thirty of such rules were defined using the JAPE pattern language, but these cannot be inspected because the work is not open source. As far as we can tell no use is made of features of an ontology.

Both systems make simplifications. [1] only aim to identify units, not quantities. No techniques are provided to deal with homonymy and synonymy of unit symbols. The matching step is based on a list of units that does not contain homonymous symbols (e.g. uses "Gs" for gauss instead of the official "G"; fahrenheit has symbol "degF"). Matching using this list will miss correct matches (e.g. when "g" is used to refer to gauss).

Simplifications made by [7] include that they assume that quantities are only written with their full name, and units only written with their symbol. Both system's high performance (over 90% F-measure) are not likely to be reached on ambiguous data as found in repositories of research results. We conclude that existing systems do not sufficiently target the homonymy and synonymy problems. In the remainder of this section we discuss techniques used in other domains that may help solve these.

[4] Obtained from `http://www.gnu.org/software/units/`

Scoring functions. A usual technique for filtering out false positives and disambiguating between alternative candidates is to provide a *scoring function* and a threshold. The candidate with the highest score is accepted (if it scores above the threshold). We give two examples of scoring functions found in literature.

Firstly, the similarity of the whole document being annotated can be compared to already correctly annotated documents. Their vector representations are compared using cosine similarity. [5] uses this technique to disambiguate matches for the same text fragment, and to find matches missed earlier in the process (in the BioCreative effort where genes are detected in medical texts; a task similar to ours). Unfortunately, the "documents" in our domain usually contain little content (in natural language) to compare. Often there is no more information available beyond the text in the header row, which is already ambiguous itself. Secondly, an example of a scoring function specific to our domain is proposed by [7]. They observe that sometimes the data cells in a column contain units and can be used as evidence to disambiguate the column's quantity. Their function is composed of (1) cosine similarity of quantity to column header; and (2) average cosine similarity of units in that column to the quantity's units. Cosine similarity is computed on a vector representation of the terms; terms are first lemmatized. This function only works if the data cells in the column contains units, which is relatively rare in our datasets.

Ontology-based filtering and disambiguation. A useful *ontology-based scoring technique* is to use concepts related to the candidate concept. If these related concepts are detected in the text near to the candidate concept, this increases the likelihood that a candidate is correct. [5] implemented this technique so that the candidate genes for string "P54" are disambiguated by comparing the gene's species, chromosomal location and biological process against occurrences of species, location and process in the text surrounding "P54". We implement this technique for our domain through the relationship between units and their quantity listed in our ontology.

[7] use the value range of units stored in the ontology to filter out false positives. They look up the data values (numbers) in the column. If the values lie outside the unit's value range, the candidate is removed. This works on their data set and quantities, but this is not likely to work for large quantitative ontologies and varied datasets. For example, a temperature value of "-20" can only rule out the unit kelvin (its scale starts from 0), but leaves celsius and fahrenheit as possible interpretations. In case we are dealing with a relative temperature, then "-20" can even not strike kelvin from the list of candidates. Celsius and fahrenheit can only be disambiguated by a few actual values, which are unlikely to appear in actual measurements.

None of the techniques mentioned above addresses the problem of ambigous compound concepts (e.g. m/s might refer to meter per second or mile per siemens). We developed a solution that uses an ontology to determine whether the units together express a quantity that is defined in the ontology.

4 Materials

The data, annotator instructions, gold standard and ontology used in our evaluation are available online[5]. We start by giving a more detailed description of the problem.

4.1 Datasets

We use two datasets to develop and validate our approach. The first set is obtained from a data repository of researchers at the Dutch Top Institute Food and Nutrition.[6]

The second dataset was collected from the Web, especially from .edu, and .org sites and sites of scientific/academic organizations. The files were found through Google by querying for combinations of quantity names and unit symbols and filtering on Excel files, such as in "speed (m/s)" filetype:xls". Topics include: chemical properties of elements, throughput of rivers, break times and energy usage of motor cycles, length and weight of test persons.

Our datasets can be considered a "worst-case scenario". The dataset of [7] is simpler in that (1) quantities are always written in their full name and units with symbols only; (2) no abbreviations or misspellings occur; (3) no compound units appear; and (4) both data and ontology contain no ambiguous unit symbols. The dataset used by [1] may be simpler because the documents (patent specifications) are intended to be precise.

We make the assumption, like [7] and [1], that the header rows have already been identified and separated from the content rows. We have effectuated this assumption by deleting cells that do not belong to the table header from the Excel files used in our experiment.

4.2 Ontology

We use an ontology we developed, the *Ontology of Units of Measure and related concepts* (OUM) in the annotation process [11]. OUM's main classes are Quantity, Unit of Measure, Dimension and Application Area. (see Figure 1 for an overview). OUM currently consists of approximately 450 quantities and about 1,000 units. Concepts have English labels, an extension in Dutch is under development.

For each quantity the units in which it can be expressed are listed. For example, speed can be expressed in (amongst others) km/s and mm/s. Each unit belongs to one or more quantities. OUM groups similar quantities into classes. For example, Kinetic energy and Heat are subclasses of Energy.

Units can be split into singular units, multiples and submultiples, and compound units. *Singular units* (units with a special name) such as meter can be prefixed to create so-called multiples and submultiples (e.g. *kilo*meter, *milli*meter).

[5] See `http://www.cs.vu.nl/~mark/iswc2010/`. The food dataset was not included as it is commercially sensitive data.

[6] `http://www.tifn.nl`

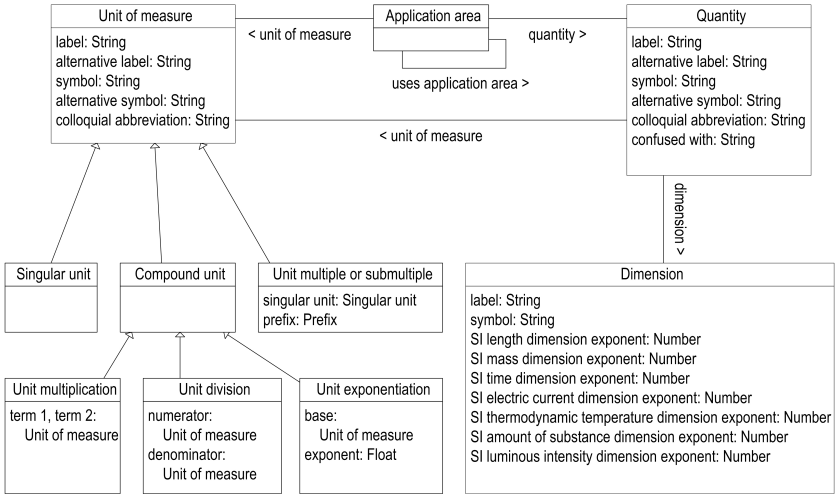

Fig. 1. UML diagram of main OUM classes and properties.

Compound units are constructed by multiplying, dividing, or exponentiating units (e.g. m/s^2). Unit multiplications are linked to their constituent units through the properties term1 and term2, unit divisions are linked to their constituents through numerator and denominator.

Because units can be prefixed and composed, the number of possible units is almost endless. For example, units for the quantity velocity may be a combination of any unit for length (e.g. kilometer, centimeter, nordic mile) and any unit for time (hour, picosecond, sidereal year, etcetera). For practical reasons OUM only lists the more common combinations, but the analysis of what is "common" has not been finalised yet. As a consequence, for specific application areas some compound units may be missing. Each quantity or unit has one full name and one or more symbols. Each full name is unique, but words in the name can overlap (e.g. "magnetic field intensity", "luminous intensity").

Humans regulary confuse some quantities with each other (e.g. weight and mass). Our ontology records the concepts and their definitions as they are prescribed in standards, but for automated annotation it is useful to know which terms people use to denote these concepts. This dichotomy is well-known in the vocabulary world, and reflected in the SKOS standard through the skos:hiddenLabel property[7]. It is used to record labels not meant for display but useful in searching. We introduce a property confused_with (subproperty of skos:hiddenLabel). By attaching the label "weight" to mass our annotation system will be able to generate mass as a candidate. In the same vein we introduce colloquial_abbreviation to denote often used abbreviations as "temp" and "freq" for temperature and

[7] http://www.w3.org/TR/2009/NOTE-skos-primer-20090818/#sechidden

frequency. Less than ten of such abbreviations and confusions are currently included.

Quantities and units are sometimes used primarily in a particular *application area*. OUM specifies generic application areas, such as *space and time* (contains units such as mile and second). OUM also contains specific areas like *shipping* (contains nautical mile) and *astronomy* (contains sidereal second).

Quantities and units have dimensions, which are abstractions of quantities ignoring magnitude, sign and direction aspects. Analysis of dimensions is common practice in science and engineering [2]. It allows for example to detect errors in equations and to construct mathematical models of e.g. aircraft. OUM lists all dimensions which occur in practice, which can be used in disambiguation of compound units (see Section 5.4). The dimension of a quantity or unit can be viewed as a vector in a space spanned by an independent set of base vectors (i.e. base dimensions). For example, the quantity speed has a dimension that can be decomposed into base dimension *length* and base dimension *time* (with certain magnitudes as we show below). In principle we could also have expressed time in terms of base dimensions distance and speed. Each *system of units* used defines such a set of base dimensions to span the dimensional space. Each other dimension can be expressed as a combination of these base dimensions, each with a certain magnitude.

For example, the SI system of units has selected as its base dimensions *length (L)*, *mass (M)*, *time (T)*, *electric current (I)*, *thermodynamic temperature (Θ)*, *amount of substance (N)* and *luminous intensity (J)*. Since all other dimensions can be computed by multiplication and division of one or more of these base dimensions, an arbitrary dimension can be expressed as multiplication $L^{\alpha}M^{\beta}T^{\gamma}I^{\delta}\Theta^{\epsilon}N^{\zeta}J^{\eta}$. If an exponent is 0, the respective basic quantity does not play a role. For example, the quantity velocity and unit cm/hr have SI-dimension $L^1M^0T^{-1}l^0\Theta^0N^0J^0$, which is equivalent to L^1T^{-1} or length per time. A quantity or unit with a dimension for which all powers are 0 is said to be dimensionless. It is typically obtained as a ratio between quantities of equal dimension, such as strain or Reynolds number, and expressed as for example fractions or percentages.

5 Approach

We have divided the annotation process into the following steps: (0) table extraction; (1) tokenization; (2) basic matching; (3) matching compounds listed in OUM; (4) matching unknown compounds using dimensional analysis; (5) disambiguation. We do not treat the extraction step here; its output is a list of cells and their contents. Our main assumption is that the identification of the header row(s) has already been done.

5.1 Tokenization

The string value of a cell is separated into tokens by first splitting on spaces, underscores ("start_time") and punctuation marks (brackets, dots, stars, etc.).

Number-letter combinations such as "100g" are separated, as are camel-cased tokens ("StartTime"). Basic classification of tokens into numbers, punctuation, and words is performed. Punctuation tokens that may represent multiplication (period, stars, dots), and division (slash) are also typed. Two other token types are detected: stopwords and a list of "modifiers" that are particular to this domain (e.g. mean, total, expected, estimated).

5.2 Basic matching: full names and symbols

Before matching takes place we generate several synonyms: plural forms of units (e.g. "meters"), contractions of compound unit symbols (e.g. "Pas" for pascal second), some alternative spellings (e.g. "C" for °C, s-1 vs. s^-1 vs. 1/s for reciprocal units, s^2 vs. s2 for exponentiated units). Because these can be generated systematically this is easier than adding them to the ontology.

Matching starts by comparing the input to full names of quantities and units, including confused_with and colloquial_abbreviations. The match with the highest score above a threshold is selected. We have used a string distance metric to overcome spelling mistakes, called Jaro-Winkler-TFIDF [3].

After full name matching is completed, a second matcher finds matches between input tokens and quantities/units based on their *symbols*, e.g. "f", "km", "s"). This is a simple exact match that ignores case. The outcome of this step will contain many ambiguous matches, especially for short unit and quantity symbols.

5.3 Matching: Compounds in OUM

The matches obtained in the basic matching in some cases represent compound units that are listed in OUM. For example, the previous step will return for the cell C.m the matches calorie, coulomb, meter, nautical mile. We detect that this is the compound coulomb meter by detecting that some of the unit matches are constituents of a compound listed in OUM. Comparison to a unit multiplication uses the properties term1 and term2, for comparing to unit division the properties numerator and denominator. In the latter case the additional constraint is that units have to appear in the input in the order prescribed (first numerator, then denominator). The punctuation used in the input determines whether we are dealing with a multiplication or a division. Notice that this step already helps to disambiguate matches; in this case calorie and nautical mile could be excluded.

A special case are compounds consisting of (sub)multiple units, e.g. μNm which stands for micronewton meter. Because OUM only lists newton meter, we have to first detect the prefix (in this case μ, other prefixes include m, M, k, T), remove it and then perform the compound check described above.

5.4 Matching: Compounds not in OUM

The previous step will miss compound units not listed in OUM. If the unit symbols in the compound are not ambiguous, we can assume that this interpretation

is correct. However, in many cases the symbols are ambiguous. For example, g/l can either denote gauss per liter or gram per liter. A way to disambiguate is to find out if the compound expresses a quantity that is listed in OUM. The quantity implied by the compound can be computed using the dimensional properties of the units (also listed in OUM).

The first step is to compute the overall dimension of the compound based on the individual units, the second step is to check whether a quantity with that dimension exists in OUM. Computing dimensions is a matter of subtracting the dimension exponents of the underlying elementary dimensions. Each unit is associated with an instance of Dimension, which in turn lists the dimension exponents through the properties SI length exponent, SI time exponent, etcetera. If, for example, we interpret g/l as gram per liter, we retrieve the units' dimensions (mass-dimension and volume-dimension, respectively). Then we divide the dimensional exponents of mass $L^0M^1T^0l^0\Theta^0N^0J^0$ by the dimensional exponents of volume $L^3M^0T^{-1}l^0\Theta^0N^0J^0$ which gives $L^{-3}M^1T^{-1}l^0\Theta^0N^0J^0$ These dimensional exponents match exactly with the dimensions of the quantity density. On the other hand, viewing g as gauss would yield $L^{-3}M^1T^{-2}l^{-1}\Theta^0N^0J^0$ for the dimension of the compound unit, which does not correspond to the dimension of any quantity in OUM.

This step is implemented by normalizing the input string, constructing a tree representation of the compound through a grammar parser, assigning the units to it, and sending it to a service that calculates the implied dimension components.

An interesting option in the future is to automatically enrich OUM with new compounds that pass the above test, and add them to OUM. This would be a valid way to continuously extend the set of compound units in OUM, not in an arbitrary manner, but learning from actual occurrences in practice. If we combine this with monitoring which compound units are never used in practice (but were added for theoretical reasons or just arbitrarily), a reliable mechanism for maintaining a relevant set of compound units in OUM would be created.

5.5 Disambiguation

The previous step will still contain ambiguous matches, e.g. for the cells f (Hz) and wght in g. We have developed a set of heuristics or "rules" to remove the remaining ambiguities.[8] First we list domain-specific pattern matching rules in the style of [1], then three disambiguation rules that make use of relations in the ontology (rules 7, 8 and 9).

Rule 1: Symbols in brackets usually refer to units. For example, "s" in delay (s) refers to second and not area or entropy.

Rule 2: Prefer singular units over (sub)multiples. Symbols for singular units (e.g. pascal (Pa)) overlap with symbols for (sub)multiples (e.g. picoampere (pA)). In these cases, select the singular unit because it is more likely.

[8] Formulated as "rules" for reading convenience, but both the rules and previous "steps" can be implemented differently.

Rule 3: A symbol that follows a number usually refers to a unit. For example, 100 g refers to gram. This disambiguation deletes six potential quantity matches for "g", and retains units gram and gauss. (Rule also used by [1].)

Rule 4: Take letter case into account for longer symbols. People are sloppy in the correct letter case of symbols. One-letter symbols such as "t" may stand for temperature (T) or tonne (t). Two-letter symbols as "Km" may stand for kilometer (km) or maximum spectral luminous efficacy (Km). Casing used in the text cannot be trusted to disambiguate; the context usually does make clear which is meant. However, casing used in writing down units of three or more letters is usually reliable. For example, (sub)multiples such as mPa and MPa (milli/megapascal) are usually written correctly. Humans pay more attention to submultiples because errors are hard to disambiguate for humans too. We thus perform disambiguation based on case if the symbol is three letters or longer.

Rule 5: Modifier words usually appear before quantities, not units. For example, mean t or avg t is an indication that "t" stands for the quantity Time instead of the unit tonne. The idea of using specific types of tokens to improve correct concept detection is due to [6] in the gene annotation domain.

Rule 6: Too many symbol matches implies it is not a quantity or unit. If previous steps were not able to disambiguate a symbol that has many candidate matches (e.g. "g" can match ten quantities and units), then the symbol probably does not refer to a quantity or unit at all (it might be a variable or e.g. part of the code of product). For such an ambiguous symbol, humans usually provide disambiguating information, such as the quantity. We therefore delete such matches. This rule can hurt recall, but has a greater potential to improve precision which will pay off in the F-measure. This rule should be executed after all other rules.

Rule 7: Symbols that refer to related quantities and units are more likely than unrelated quantities and units. For example, T (C) is more likely to refer to temperature and celsius than to time and coulomb. The former pair is connected in OUM through property unit_of_measure (domain/range Quantity/Unit), while the latter pair is not. We filter out the second pair of matches. We first apply this rule on quantities and units in the same cell. This rule also allows to select the quantity mass for cell weight (g) instead of the erroneous weight. Mass was found in basic matching through its confused_with label. We repeat application of the rule on the whole table after application on single cells. A quantity mentioned in one cell (e.g. mass) can thus be used to disambiguate cells where the quantity was omitted (e.g. containing only "g"). During application of this rule we prefer matches on preferred symbols over matches on non-preferred ("alternative") symbols. For example, cell Length (m) matches length-meter (meter has symbol "m") which we prefer over length-nautical mile (mile has alternative_symbol "m").

Rule 8: Choose the most specific quantity that matches the evidence. Generic quantities such as Viscosity and Temperature have specific instances such as kinematic viscosity and celsius temperature. The user may have meant the specific quantity. If a unit is given, this can be disambiguated. For example, viscosity expressed in stokes means that kinematic viscosity was meant. When poise is used, dynamic viscosity was meant. In other cases, the units of the specific quantities overlap, so that the proper quantity cannot be determined (e.g. diameter and radius are forms of Length measured in units such as meter.

Rule 9: Choose the interpretation based on the most likely application area. Symbols such as "m" can refer to units from a generic application area or a specific application area (e.g. nautical mile in *shipping* or meter in *space and time*). If there is evidence that the table contains measurements in a specific area then all ambiguous units can be interpreted as a unit used in that area, instead of those in more generic areas. If there is no such evidence, the unit from the generic area is more likely. As evidence that the observations concern a specific area we currently accept that the table contains at least one unambiguous unit that is particular to that area (i.e. written in its full name). Other types of evidence can be taken into account in the future (e.g. column name "distance to star").

5.6 Implementation

We developed a prototype implementation of our annotation approach in Java. It provides a simple framework to implement matchers and disambiguation rules. Our matchers and disambiguation rules can probably also be implemented as JAPE rules on top of GATE; this is future work.

The Excel extractor uses the Apache POI library[9]. The prototype can emit the parsed and annotated tables as RDF files or as CSV files. For representing and manipulating the OUM ontology and the output as objects in Java we used the Elmo framework[10] with Sesame as RDF backend. For string metrics we use the SecondString[11] library developed by Cohen et al. The parser for compound units was built using YACC.

6 Evaluation and Analysis

6.1 Evaluation type and data selection

We evaluate our approach by measuring recall and precision against a gold standard for two datasets. We could not measure the performance of our system on the data of [1] because it is not publicly available. Comparison against the data of [7] is not useful as they identify only a few (unambiguous) quantities and units.

[9] http://poi.apache.org/
[10] http://www.openrdf.org/doc/elmo/1.5/
[11] http://secondstring.sourceforge.net/

The tables were selected as follows. We randomly selected files from the food dataset and removed those that were unsuitable for our experiment because they were (1) written in Dutch; or (2) contained no physical quantities/units; or (3) had the same header as an already selected file (this occurs because measuring machines are used that produce the same table header each time). We kept selecting until we obtained 39 files. Selection of 48 Web tables was also random; no tables had to be removed.

The number of correctly and wrongly assigned URIs is counted on a per-document basis, by comparing the set of URIs returned by the system with the set of URIs of the human, ignoring the cell in which they were found. Based on the total number of correct/wrong/retrieved URIs, the macro-averaged precision and recall is calculated (each correct/wrong URI contributes evenly to the total score).[12]

6.2 Gold standard creation

The files were divided over three annotators (the authors). They used an Excel add-in [11] developed in earlier work that allows selection of concepts from OUM. Each cell could be annotated with zero or one quantity, and zero or one unit. The annotators were incouraged to use all knowledge they could deduce from the table in creating annotations. If the exact quantity was not available in OUM, a more generic quantity was selected. For example, the cell half-life (denoting the quantity for substance decay) was annotated with Time. After that, each file was checked on consistency by one of the authors.

Compound units that do not appear in OUM can not be annotated by assigning a URI to them (simply because they have no URI in OUM). They were put in a separate result file and were compared by hand.

6.3 Results

We have tested different configurations (Table 1). Firstly, a *baseline* system that only detects exact matches, including our strategies to enhance recall such as contraction of symbols and generation of plural forms (comparable to [7]'s system). Secondly, with flexible string matching turned on. Thirdly, with pattern disambiguation rules turned on (rules 1-6); this may be comparable to the GATE-based system [1]. We cannot be certain because their system is not open source. This indicated what can be achieved with pattern matching only. Fourthly, with also compound detection and ontology-based rules turned on (rules 7-9) .

The following points are of interest. Firstly, the baseline scores show that the extent of the ambiguity problem is different for quantities and units. Performance

[12] A comparison per cell would introduce a bias towards frequently occurring quantities/units, which either rewards or punishes the system for getting those frequent cases right. Micro-averaging calculates precision/recall for each document and takes the mean over all documents. A single annotation may contribute more or less to the total precision or recall, depending on whether it appears in a document with little or a lot of annotations.

for quantities is not high (F-measure ranging from 0.09 to 0.20), while F-measure for units is already reasonable (around 0.40). It turns out that the datasets in our experiment relatively often use non-ambigous unit symbols, including "N" for newton and "sec" for second. Secondly, flexible string matching does not help to increase recall (threshold 0.90 was used but no clear increase was seen at 0.85 either). The results of the remaining two configurations are obtained with flexible matching turned off. Thirdly, pattern matching rules help considerable, improving F-measure with 0.15-0.60. Fourthly, ontology-based disambiguation increases the F-measure further for units: 0.16-0.25. The results for quantities are mixed: 0.07 increase in the Food dataset, no difference in the Web dataset. Fifthly, in the Web dataset unit scores are higher than quantity scores, and the other way around in the Food dataset.

	Food						Web					
	Quantities			Units			Quantities			Units		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
baseline	0.11	0.84	0.20	0.30	0.61	0.40	0.05	0.70	0.09	0.29	0.61	0.40
flex. match	0.11	0.84	0.20	0.29	0.61	0.39	0.05	0.72	0.09	0.28	0.61	0.39
pat. rules	0.78	0.82	0.80	0.50	0.57	0.53	0.63	0.64	**0.63**	0.50	0.57	0.53
full	0.83	0.93	**0.87**	0.72	0.83	**0.78**	0.59	0.67	**0.63**	0.63	0.76	**0.69**

Table 1. Results of evaluation. Separate precision (P), recall (R) and F-measure (F) are given for both datasets, based on macro-averaging. Best F-measures are in bold.

6.4 Qualitative analysis

We analyzed the causes for false positives and false negatives in the results. The following should be highlighted. Firstly, the performance of the pattern rules is not improved upon as much as we had expected in the case of quantities. One explanation is that many of the symbols in the input did not represent a quantity, and the pattern rules successfully filter these false positives out through rule 6. In the future we will try our method on more varied datasets to determine if this effect is consistent or not.

Secondly, some quantities are simply missing in OUM, such as half life and resonance energy. The annotators used the more generic quantity (time and molar energy) to annotate the cells where they appear. The generic quantities are not not found because there is no lexical overlap. This can be solved by adding them or importing them from another ontology. Thirdly, a number of quantities is not found because they are not mentioned explicitly, but implied. For example, letters X and Y are used to indicate a coordinate system, and thus imply length. Failing to detect the quantity also causes loss of precision in unit detection: the quantity would help to disambiguate the units through rule 7. These issues points to the importance of a high-coverage ontology.

Fourthly, another cause for missed quantities is that the object being measured is stated, which together with the unit implies the quantity. For example, the cell Stock (g), refers to quantity mass as the word "stock" implies a food product (stock is a basis for making soup). This can be solved by using more ontologies in the matching step, and link concepts from those ontologies to OUM. For example, a class Food product could be linked to quantities that are usually measured on food products such as mass. Because field strength is not one of those quantities, the erroneous match gauss could be removed.

Fifthly, some of the problems are difficult to solve as very case-specific background knowledge would be required. For example, cells Lung (L) and Lung (R) produce false positive matches such as röntgen and liter and can only be solved with knowledge on human fysiology.

Lastly, analysis of the detection of compounds not in OUM shows that this step performed well at recognizing unit divisions (kilojoule per mole, newton per square millimeter). However, its performance is degraded considerably by false positives such as dP for decapoise and V_c for volt coulomb.

7 Discussion

In this paper we have studied annotation of quantitative research data stored in tables. This is relevant to today's world because scientists, companies and governments are accumulating large amounts of data, but these datasets are not semantically annotated. We presented several ways in which an ontology can help solve the ambiguity problems: (1) detection of compound units present in the ontology; (2) dimensional analysis to correctly interpret compound units not explicitly listed in the ontology; (3) identification of application areas to disambiguate units; and (4) identification of quantity-unit pairs to disambiguate them both. Especially the performance for unit detection is good. This is positive, as correct unit detection is more important than correct quantity detection: the quantity can be derived from the unit using the ontology. For example, time can be derived from millisecond. Even when the right specific quantity is not known (e.g. half-life), the more generic quantity that could be derived is a suitable starting point for data integration. For example, to integrate two datasets about the half-life of elements it is sufficient to know that columns are being merged that deal with time (if the units are not the same they can be automatically converted into each other).

However, performance is still far from perfect. We have suggested several ways in which performance may be improved, of which linking ontologies about the objects being measured is an attractive one. One promising line of future work is the application of machine learning (ML) techniques to the disambiguation problem. However, this is not straightforward since our domain lacks the typical features that ML approaches rely on, e.g. those based on the surrounding natural language text. We do see possibilities to use the properties of the candidate concepts as features and thus combine our rule-based approach with

a machine learning approach – as e.g. proposed by [10]. This would require a larger annotated dataset to serve as training and test set.

An implication of this work for the Web of Data is that conversion tools need to be tuned to the domain at hand. Current tools target sources that are already structured to a large extent, but if the Web of Data is to grow, more unstructured sources should be targeted. The work of [9] already suggests to include an annotation system into a conversion tool, but the annotation system is generic. As shown a generic system will fail to capture the semantics of this domain. A system that can be configured for the domain is required.

Acknowledgements

This work was carried out within the Food Informatics subprogram of the Virtual Laboratory for e-Science, a BSIK project of the Dutch government. We thank Jeen Broekstra for implementation advice, Remko van Brakel for the Excel export tool, and Laura Hollink and Tuukka Ruotsalo for their comments.

References

1. Agatonovic, M., Aswani, N., Bontcheva, K., Cunningham, H., Heitz, T., Li, Y., Roberts, I., Tablan, V.: Large-scale, parallel automatic patent annotation. Conference on Information and Knowledge Management (2008)
2. Bridgman, P.: Dimensional Analysis. Yale University Press (1922)
3. Cohen, W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proc. of IJCAI-03 Workshop on Inf. Integration. pp. 73–78 (2003)
4. Ding, L., DiFranzo, D., Magidson, S., McGuinness, D.L., Hendler, J.: The Data-gov Wiki: A Semantic Web Portal for Linked Government Data. In: Proc. of the 8th Int'l Semantic Web Conference. LNCS, vol. 5823. Springer (2009)
5. Hakenberg, J., Royer, L., Plake, C., Strobelt, H., Schroeder, M.: Me and my friends: gene mention normalization with background knowledge. In: Proc 2nd BioCreative Challenge Evaluation Workshop. pp. 1–4 (2007)
6. Hanisch, D., Fundel, K., Mevissen, H., Zimmer, R., Fluck, J.: ProMiner: rule-based protein and gene entity recognition. BMC bioinformatics 6 Suppl 1, S14 (2005)
7. Hignette, G., Buche, P., Dibie-Barthélemy, J., Haemmerlé, O.: Fuzzy Annotation of Web Data Tables Driven by a Domain Ontology. In: Proc. of the 6th European Semantic Web Conference. p. 653. Springer (2009)
8. Langegger, A., Woss, W.: Xlwrap - querying and integrating arbitrary spreadsheets with sparql. In: Proc. of the 8th Int'l Semantic Web Conference. LNCS, vol. 5823, pp. 359–374. Springer (2009)
9. Lynn, S., Embley, D.W.: Semantically Conceptualizing and Annotating Tables. In: Proc. of the 3rd Asian Semantic Web Conference. pp. 345–359. Springer (2008)
10. Medelyan, O., Witten, I.: Thesaurus-based index term extraction for agricultural documents. In: Proc. of the 6th Agricultural Ontology Service (AOS) workshop at EFITA/WCCA (2005)
11. Rijgersberg, H., Wigham, M., Top, J.L.: How semantics can improve engineering processes - a case of units of measure and quantities (2010), accepted for publication in Advanced Engineering Informatics