RESEARCH

Open Access

Standardizing free-text data exemplified by two fields from the Immune Epitope Database

Sebastian Duesing^{1*}, Jason Bennett¹, James A. Overton², Randi Vita¹ and Bjoern Peters^{1,3*}

Abstract

Background While unstructured data, such as free text, constitutes a large amount of publicly available biomedical data, it is underutilized in automated analyses due to the difficulty of extracting meaning from it. Normalizing free-text data, *i.e.*, removing inessential variance, enables the use of structured vocabularies like ontologies to represent the data and allow for harmonized queries over it. This paper presents an adaptable tool for free-text normalization and an evaluation of the application of this tool to two different fields curated from the literature in the Immune Epitope Database (IEDB): "age" and "data-location" (the part of a paper in which data was found).

Results Free text entries for the database fields for subject age (4095 distinct values) and publication data-location (251,810 distinct values) in the IEDB were analyzed. Normalization was performed in three steps, namely character normalization, word normalization, and phrase normalization, using generalizable rules developed and applied with the tool presented in this manuscript. For the age dataset, in the character stage, the application of 21 rules resulted in 99.97% output validity; in the word stage, the application of 94 rules resulted in 98.06% output validity; and in the phrase stage, the application of 16 rules resulted in 83.81% output validity. For the data-location dataset, in the character stage, the application of 39 rules resulted in 99.99% output validity; in the word stage, the application of 187 rules resulted in 98.46% output validity; and in the phrase stage, the application of 12 rules resulted in 97.95% output validity.

Conclusions We developed a generalizable approach for normalization of free text as found in database fields with content on a specific topic. Creating and testing the rules took a one-time effort for a given field that can now be applied to data as it is being curated. The standardization achieved in two datasets tested produces significantly reduced variance in the content which enhances the findability and usability of that data, chiefly by improving search functionality and enabling linkages with formal ontologies.

Keywords Unstructured data, Free-text data, Data normalization, Data standardization, Immune epitope database, Ontology

*Correspondence: Sebastian Duesing sduesing@lji.org Bjoern Peters bpeters@lji.org Full list of author information is available at the end of the article

Background

A lot of data within and outside the biomedical field is unstructured, a category that includes data in the form of text, images, audio, and video, with estimates ranging as high as 95% [1]. Unstructured data is commonly underutilized due to the difficulty of automatically extracting meaningful information from these forms of data. We work on the Immune Epitope Database (IEDB) [2],

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

a publicly available database of immune epitopes that prominently features epitopes recognized in the context of infectious and immune-mediated diseases. The IEDB links its data to ontology terms so its data is interoperable with other ontologized databases and connected to the large network of ontology metadata [3]. In our work on the IEDB, we have found that unstructured data also lags behind structured data in adherence to FAIR data standards, and we identified data from certain free-text fields as a target for improvement [3]. Creating more linkages between free-text data in the IEDB and structured vocabularies like ontologies is one way to improve the FAIRness of that data [3]. Normalizing free-text data, *i.e.*, removing variance that does not affect meaning from text, can be used to align data to structured vocabularies. The IEDB therefore aims to use text normalization to enable linkages between that data and ontology terms. This paper presents a novel repository of Python scripts for free-text data normalization and an evaluation of the application of these scripts to two different sets of biomedical data from the IEDB, an age dataset and a data-location dataset.

Variance, a term that this paper uses to refer to differences in representations of information that do not change meaning, is a key problem of free-text normalization. Free-text data can contain several different kinds of variance. Character variance (such as differences in diacritic usage, whitespace, or encoding) differentiates data items like "6-8 weeks" and "6-8 weeks". Word-level variance, which includes misspellings, abbreviations, synonyms, and colloquialisms, differentiates data items like "6-8 weeks" and "6-8 wks". Phrase-level variance includes the ways that one idea can be expressed with different permutations of words, and it differentiates data items like "6-8 weeks" and "6 to 8 weeks". The data items "6-8 weeks", "6-8 weeks", "6-8 wks", and "6 to 8 weeks" all mean the same thing, but in their unstandardized free-text forms, they are all parsed as distinct. The aim of free-text normalization is to ensure that data items that mean the same thing look the same way. The extent to which each of those three types of variance might exist in a particular dataset is highly dependent on the nature of the data. Text normalization has been applied to a variety of domains, including use on text from social media [4]. Domainindependent normalization strategies must be able to parse diverse forms of textual variance, as data from different domains presents unique challenges, e.g., the variance in a biomedical dataset is likely to be different than that of a dataset of text from social media. Normalization of text from social media, for instance, faces distinctive challenges resulting from use of shortform abbreviations (e.g., "nite" for "night" or "gr8" for "great"), intentional misspellings for effect (e.g., Page 2 of 18

"soooooo great"), and emoticons (e.g., "<3"), which have hindered the performance of automated tasks like machine translation on text from social media [4]. These text features are unlikely to be found in biomedical datasets, which contain their own distinctive forms of variance, including uncommon technical terminology, symbols, and abbreviations, like occurrences of the character μ , which is otherwise uncommon in English text, in the unit " μ g" ("micrograms), and the use of "ug" and "mcg" as alternatives. To be broadly applicable to free-text datasets of all sorts, a free-text normalization tool must be able to address all three types of variance in a way that is flexible enough to account for different datasets' unique normalization needs.

There is a robust history of development of automated tools for addressing some types of variance, such as spellcheck technologies, but there are comparatively few holistic tools designed to normalize dataset variance at the character, word, and phrase levels, though the proliferation of LLMs has led to the use of a variety of LLM-based tools for text normalization. However, there remains a use-case for non-LLM-based tools for datasets with features such as uncommon data elements, which tend to decrease the performance of LLM-based-tools (see the Evaluating the Utility of ADP and Other Tools subsection in the Discussion section below). To that end, we created the non-LLM-based free-text normalization tool ADP, which stands for Adaptable, user-Dependent, and Precise. In this paper, we examine the application of ADP to data from two free-text fields from the IEDB: the "age" field, which contains subject ages, and the "data location" field, which contains information about data provenance, both of which were accessed using SQL queries.

The age dataset records the ages of subjects in investigations archived in the IEDB. It contains 7,151 total unique organism-age pairs (e.g., age: "6–8 weeks old", organism name: "Mus musculus C57BL/6"), meaning some age values are duplicated in that dataset because they occur with multiple organisms; there are 4,095 unique age value strings. Strings in the age dataset typically contained one piece of information per string, and where list-like strings were present, they were legitimate lists ostensibly linked to studies that investigated subjects at multiple specific ages, e.g., the data item "21, 27 and 36 weeks".

The data-location dataset the provenance of data from an IEDB-curated manuscript. It contains 251,810 unique data-location strings, such as "Cited reference [PMID: 16472860]". In contrast with the age dataset, many strings in this dataset contained several individually valid data locations in a single line, such as "Data set S1 and S11 and Figs. 1, 2, 3, and 4". See Table 1 below for further examples from each dataset.

 Table 1
 Example age & data-location data items

Age Dataset	Data-Location Dataset
6 to 8 weeks	Figures 2, 3, 4, 5, 6, S4, S5, S7, Tables 2, 3, 4 and 5
Adults (pregnant)	PDB: 5EC1, 5EC2, 5EBW, 5EBL, 5EBM
Mean age of 32.2 years with a range from 18 to 49 years	Richardson et al. Virol 1986;155:508– 523 [PMID: 3788062]
18–22 months or 4–6 months	pg. 1410 and J. Virol. 61:1358–1367

^a The data-location dataset contained a large number of Protein Data Bank (PDB) identifiers that parsed as distinct words. These IDs, which follow a standard four-character alphanumeric format, were selected using a regular expression and then mass-allowed. There are 186 non-PDB-ID words in the data-location reference file

The data in both of these datasets were input into the free-text "age" and "data location" fields by the human curators who enter publication data into the IEDB. Since the start of the IEDB's data collection process in 2004, these fields have been collected as free-text data because the diversity of values and formats present among the data made it impossible to develop a single standardized input format that would work for all data items from all publications. For the age field, subject ages reported in publications frequently including complex information in several formats (e.g., multiple age ranges, statistical information like means and medians, life stages, etc.), and for the data-location field, it may be necessary to include publication-specific section titles or individual publication names, etc., that are not found in an existing structured vocabulary. Even after standardization, the values and formats of these fields remain diverse. As both fields are input as free-text, we regard the individual data items that result from this process as unstructured data because they are not in a singular standardized format.

Methods

ADP is a non-fully-automated normalization tool that enables a user to create standardization rules and apply them to datasets, which is available on GitHub [5]. The ADP normalization scripts are written in Python version 3.10. The core normalization scripts import the libraries os, re, and sys from the Python Standard Library and the non-native library editdistance (imported as ed). ADP is open-source software licensed under GNU GPL-3.0.

ADP's three core normalization scripts (char_normalizer.py, word_normalizer.py, and phrase_normalizer. py) address the three types of variance outlined in the introduction: character-, word-, and phrase-level variance. At the character and word stages, ADP also logs a Levenshtein distance score for each data item to indicate the extent of the changes made in that stage. ADP uses a script (calculate_metrics.py) to pull relevant metrics from the normalized output files and generate figures using the Python libraries ast, math, matplotlib.pyplot (imported as plt), pandas (imported as pd), seaborn (imported as sns), and warnings.

ADP text normalization workflow

Action decision-based normalization of characters and words While standardizing character variance can be as simple as selecting acceptable special characters and determining case-sensitivity of the data, standardizing word-level variance involves identifying and correcting misspellings in free-text data, a process which is well-known to be "cumbersome" [4]. Normalization tools must also be able to handle "non-standard words," including numbers, acronyms, and other abbreviations [6]. Some existing word normalization tools overcorrect and have higher rates of "unresolved errors," or incorrectly-spelled words that the tool swaps with a context-incorrect word; others tend to undercorrect, e.g., by failing to recognize "cant" as a misspelling of "can't" [4]. ADP uses an iterative character and word normalization process designed to prioritize accuracy of outputs.

The character and word normalization scripts share a similar rule-building workflow. When one of these two scripts is run on a dataset for the first time, it identifies distinct text units (characters or words, which for ADP's purposes is a sequence of characters delineated by one of several common separators, like hyphens, spaces, or punctuation, or the start or end of a string) and creates a review file to be used for normalization rule-setting.

The review file is a TSV containing one row for each distinct character—except lowercase letters, digits, and a small number of basic punctuation characters, which are treated as valid for character normalization—or word found in the file. It has columns for the character or word, its context (i.e., the data item strings in which that character or word was found), and a count of its occurrences. The review file also has four action columns with the headings "replace_with", "remove", "invalidate", and "allow". Entering text in one of the action columns (which we refer to as "making an action decision") sets a rule for the behavior of the script concerning the character or word in that row during future runs of the script. Table 2 describes how entering text in one of the action columns modifies the behavior of the script.

Every time the script is rerun, it moves any review file rows in which an action decision has been made to a reference file, which serves as a bank of rules for the behavior of the script.

Tables 3 and 4 contain examples of the rules applied to these datasets at the character and word stages. These tables are intended to summarize the implementations of the example rules and are abridged from the reference

Table 2Action decisions

Action Column	Function
replace_with	This character or word is replaced with the text that is entered in this column
remove	This character or word is removed from the data items in which it occurs
invalidate	This character or word remains as-is, and data items containing this char- acter or word will fail validation
allow	This character or word remains as-is, and this character or word is considered an accepted text unit for validation

files. The context column, which contains a sample (up to 300 characters in length) of data items containing a given character or word, has been omitted here to improve the readability of this table in the manuscript, and the individual action decision columns are condensed into the singular "Rule" column in the tables below. Please refer to the reference files in the ADP repository to see the full versions of the tables and all rules applied at the character and word stages [5].¹

Following the transfer of rows with new action decisions from the review file to the reference file, the script runs its normalization functions, applying the rules based on the user's action decisions to the dataset, and it checks for any new text units that do not have a line in either the review file or the reference. See Fig. 1 for a visual representation of how this process works during the character normalization stage.

In the character normalization stage, data items pass validation if in the second reference check (as shown in the diagram), only allowed characters are found in the string; otherwise, validation fails. Only data items that pass character-level validation are normalized in the word normalization stage. Data items pass word-level validation if in the second reference check, only allowed words are found in the string; otherwise, validation fails.

Pattern-based normalization of phrases

ADP phrase normalization uses a process of matching phrase structures to user-defined patterns. This process begins in the word normalization stage. In the word review and reference TSV, there is an additional "category" column. Adding text to this column in the row of a particular word asserts the category to which that word belongs, *e.g.*, in rows for the words "week", "month", and "year", the category has been set to "unit" in the word reference TSV for the age dataset.

When the phrase normalization script is called, it divides the data item into individual words as was done for the word normalization phase. The script tracks the word's place in the string and any delimiters (including punctuation, whitespace, and the start or end of a string) on either side of the word. Then, it searches the word

 Table 3
 Sample character normalization rules & applications to data items

Dataset	Char	Occurrences	Example string	Rule	Post-normalization string
age	=	65	"mean age = 30 years"	Allow	"mean age = 30 years"
age	-	31	"20–67 years"	Replace with: -	"20-67 years"
data-loc	&	53	"Abstract & p. 664"	Replace with: and	"abstract and p. 664"
data-loc	€	10	"Fig. 1 and Fig. 1â€"figure supple- ment 1 and PDB 6HD8"	Invalidate	Invalid, not normalized

Table 4 Sampl	le word normal	ization rules &	applications t	o data items
---------------	----------------	-----------------	----------------	--------------

Dataset	Word	Occurrences	Example string	Rule	Post-normalization string
age	old	710	"6–10 week old"	Remove	"6–10 week"
age	wk	57	"8–10 wk"	Replace with: week	"8-10 week"
data-loc	fig	285	"Figs. 1 and 2"	Replace with: figure	"Fig. 1 and 2"
data-loc	file	148	"additional file 1"	Allow	"additional file 1"

¹ The reference files can be found at the following paths in the repository:

age/output_files/char_reference.tsv.

age/output_files/word_reference.tsv.

data_loc/output_files/char_reference.tsv.

data_loc/output_files/word_reference.tsv.



Dataset	Pattern name	Pattern	Valid?	Standard form	Example matched phrases	Example normalized phrases
age	range	[number(0)] [range_indica- tor(1)] [number(2)] [unit(3)]	Y	[0]-[2] [3]	"6 to 8-week", "44.9 to 74.1 year", "36 to 68.2 year"	"6–8 week", "44.9–74.1 year", "36–68.2 year"
age	statistical	[statistical(0)] [number(1)] [unit(2)]	Y	[0]: [1] [2]	"mean 29.8 year", "mean: 30 year", "median: 7.5 year"	"mean: 29.8 year", "mean: 30 year", "median: 7.5 year"
age	unitless range	[number(0)] [range_indica- tor(1)] [number(2)]	Ν		"8–10", "31–80", "12–20"	N/A
data-loc	pdb id	[pdb(0)][pdb_id(1)]	Y	[0] [1]	"pdb 1mfd","pdb 1rzj","pdb 1rzk"	"pdb 1mfd", "pdb 1rzj", "pdb 1rzk"
data-loc	loc number	[location(0)] [number(1)]	Y	[0] [1]	"page 11,782", "information 9", "data 1"	"page 11,782", "information 9", "data 1"
data-loc	number	[number(0)]	Ν	_	"3","1","151"	N/A

 Table 5
 Sample phrase normalization rules

reference file to see if a category has been assigned to the word; if not, it categorizes the word as "unknown". The script produces a string that uses a simple grammar to indicate the categories of each word and their position in the string, e.g., the age datum "6 week mean" is parsed as "[number(0)][unit(1)][statistical(2)]". The phrase categorization string is stored in a dedicated column in the phrase normalization output file to enable the user to determine which phrase structures occur the most frequently in a dataset and develop normalization rules accordingly.

Like the character and word normalization phases, the phrase normalization phase depends on the user to create rules for distinct phrase structures. A dataset's phrase-type ruleset (found in age_phrase_types.tsv and data_loc_phrase_types.tsv) establishes a name for a pattern, indicates whether or not it is a valid pattern (e.g., in the age dataset, a data item consisting of a number and a unit is valid, but a number by itself is not, as being unitless makes its meaning uncertain), and sets a rule for how phrases that match that pattern should be formatted. See Table 5 for examples.

The categorization string, e.g., [number(0)][unit(1)][statistical(2)] (extracted from "6 week mean"), is matched to a pattern—in this case, the pattern called "statistical" which matches to the structures of data items that provide a mean or median age value. In the "standard_form" column in the phrase-type ruleset, the user can specify how data items matching a pattern should be formatted. In the case of "6 week mean", the standard form is represented as "[2]: [0] [1]", in which the numbers in brackets refer to the indices from the categorization string, and how they should be arranged within the standard form string.

The phrase normalization script generates a blank phrase-type ruleset file if none exists, but if one exists, it checks each data item's categorization string against any patterns in the file and applies the pattern in the "standard_form" column if applicable by inserting words where their indices are placed in the standard form string. Through this process, "6 week mean" is rearranged to match the standard form string "[2]: [0] [1]", so the output for that data item is "mean: 6 week". This work-flow ensures that data items with diverse structures, like "6 week mean" and "mean=6 week", take on a single standard phrase structure, like "mean: 6 week". The specific structure we chose for data items of this type is arbitrary; the crucial part is the ability to quickly modify diversely expressed data items into one standard style.

Table 5 contains sample rows from both datasets' phrase type tables as examples of the rules applied to these datasets. To see the full phrase type tables and all normalization rules applied at the phrase stage, please refer to the relevant files in the ADP repository [5].²

Patterns designated as invalid, like the "unitless range" pattern in the age dataset or the "number" pattern from the age dataset, are used to catch and invalidate unusable data: unitless ranges, for instance, are invalidated at the phrase stage because the variety of units used in other age data items renders the meaning a data item like "8-10" ambiguous. Lone numbers in the data-location dataset suffer from the same problem, as "3" could refer to a page number, a line number, a section, etc. For a more thorough explanation of why invalidating these data items is a desirable outcome of the normalization process, please see the Validity Rate by Dataset and Stage section below. If a data item matches to an invalid pattern, it fails validation at the phrase stage and is not normalized. Data items that match to valid patterns, like "range" or "pdb id", have their component words rearranged to match the format specified in the standard form column to bring them into alignment.

Only data items passing validation at the character and word stages are normalized at the phrase stage. At the phrase stage, data items pass validation only if they match to a pattern designated as valid. The phrase normalization and validation processes are visualized in Fig. 2.

² The phrase type files can be found at the following paths in the repository: age/input_files/age_phrase_types.tsv. data_loc/input_files/data_loc_phrase_types.tsv.



Fig. 2 Flowchart of ADP phrase normalization processes

Measuring string change during normalization

The ADP normalization code imports the package editdistance to measure the Levenshtein distance between the inputs and outputs in the character and word normalization stages. The normalized output files contain dedicated columns for distance scores comparing the character-normalized string against the original and the word-normalized string against the character-normalized string. Levenshtein distance ceases to be a sensible measure of continuity between input and output at the phrase normalization stage, as desirable and innocuous changes in word order can produce high Levenshtein distance scores. For instance, the hypothetical age data items "18 years average" and "average 18 years" have a Levenshtein distance of 14 despite being semantically identical. While identifying string meaning is not within ADP's scope, it may prove useful in the future to implement existing Python tools to calculate semantic or cosine similarity as a metric of change at the phrase stage.

Modular normalization & accessory stages

The ADP normalization process is designed to be modular; because it is split into discrete processes for character, word, and phrase normalization, it is possible to plug in accessory stages to address dataset-specific normalization needs that are not easily handled within the predefined stages. The data-location dataset, for instance, implements an accessory stage to split list-like data items into individual strings for data location.

Data-location splitting

Because the data-location dataset contained list-like data items in which several distinct data locations were included in a single data item (e.g., the real data item "Fig. 2A,B,C, Fig. 6."), phrase normalization would be much more difficult without splitting list-like inputs into multiple items that could then be normalized independently. The script functions as a pre-phrase-normalization stage for the data-location dataset; that script creates multiple rows from list-like data items, transforming the single data item "Fig. 2A,B,C, Fig. 6." into a set of segments including "Fig. 2a," "Fig. 2b", "Fig. 2c", and "Fig. 6". Each segment is separated into

Table 6 Sample data items at each stage

a distinct row, which is assigned a post-splitting index and an original index to be able to both track segments individually and trace them back to the list-like data items from which they were originally split.

When phrase normalization is applied to the datalocation dataset, because the segments have been split into their own rows, they are treated as distinct phrases, allowing all of the "figure x" example segments above to match to a single pattern, rather than needing dedicated patterns to match to each list-like permutation.

Sample normalized data items

Table 6 contains sample data items from the age and datalocation datasets. The columns represent the progression of these data items through the normalization process, with changes made by the character, word, and phrase normalization parts of the code represented in those respective columns. Note that for the data-location dataset, the list-like phrase-normalized strings are split into individual TSV rows for each data item in the list, e.g., the single input data item "Fig. 2A,B,C, Fig. 6." becomes four output data items: "Fig. 2a," "Fig. 2b," "Fig. 2c," and "Fig. 6."

Results

Using ADP's normalization scripts on the IEDB age and data-location datasets demonstrates that it is possible to use ADP to effect significant improvements to the overall standardization of a dataset.

User action efficiency

ADP is a tool for the development and implementation of standardization rules. Accordingly, the thoroughness with which a user makes action decisions (in the character and word stages) and builds phrase type patterns (in the phrase stage) determines the overall success of ADP at standardizing a dataset. The data presented in this manuscript is the result of a non-exhaustive approach to both datasets in which rule-setting for particularly common characters, words, and phrases was prioritized, to represent a practical and realistic normalization outcome.

Table 7 provides an overview of the extent of the normalization rule-setting done for each dataset. The "items

Dataset	Before Normalization	Character Normalized	Word Normalized	Phrase Normalized
Age	Six week old	six week old	6 week	6 week
Age	6–8 week	6 to 8-week old	6 to 8-week	6–8 week
Age	Median age 6.3 years	median age 6.3 years	median 6.3 year	median: 6.3 year
Data-Location	Additional File 4, Tables 1 and 2	additional file 4, Tables 1 and 2	additional file 4, Tables 1 and 2	['additional file 4', 'Table 1', 'Table 2']
Data-Location	Figures 2A,B,C, Fig. 6	Figures 2a,b,c, Fig. 6	Figures 2a,b,c, Fig. 6	['Fig. 2a', 'Fig. 2b', 'Fig. 2c', 'Fig. 6']
Data-Location	Figure 2A,B, Suppl Fig. 2	Figure 2a,b, suppl Fig. 2	Figure 2a,b, supplemental Fig. 2	['Fig. 2a', 'Fig. 2b', 'supplemental Fig. 2']

Table 7 Number of action decisions by dataset

	Age Dataset	Data-Location Dataset
Characters in review	1	7
Words in review	84	1160
Characters in reference	21	39
Words in reference	94	5780 counting mass-allowed Protein Data Bank (PDB) identifiers, otherwise 186ª
Phrase-type patterns	16	12
Valid phrase-type patterns	9	11

in review" counts reflect the number of characters or words for which action decisions were not made at the time of manuscript submission. The "items in reference" counts reflect the number of characters or words for which action decisions were made. The "phrase-type patterns" counts reflect the number of user-generated patterns against which phrases are matched to determine their validity, and "valid phrase-type patterns" reflect how many of the defined patterns are specified as valid phrases.

The results presented in this manuscript are accordingly the results of a non-exhaustive rule-setting effort intended to prioritize the creation of rules targeting highoccurrence characters, words, and phrase patterns. More comprehensive normalization and higher validity rates at each stage could be achieved by targeting increasingly lower-frequency characters, words, and phrases. Ultimately, reasonable stopping points will vary for each dataset; making action decisions and creating phrase patterns for increasingly infrequent characters, words, and phrases offers diminishing returns in overall dataset standardization.

The specific amount of time and effort that goes into creating the normalization rules varies substantially based on their complexity and the experience of the person creating them. In our hands, we have found that one cycle of rule implementation and testing takes 5–20 min, with 5 being typical. Some rules make sense to implement as a group, e.g., uppercase-to-lowercase conversions at the character stage or plural-to-singular conversions at the word stage. For these, we typically make the desired changes in all relevant lines in the review file, rerun the script, and then inspect the results in the output file. We find that adding several rules in one cycle of implementation and testing does not substantially increase the time taken by that cycle. A high estimate of the time it takes to normalize these datasets can be calculated by multiplying the number of rules by the typical time of 5 min per rule, which translates to approximately 11 h for the age dataset (with 131 rules) and approximately 20 h for the datalocation dataset (with 237 rules). However, in practice, the time it took us is significantly lower than this because of the efficiency added by implementing multiple similar rules together at one time.

Validity rates by dataset and stage

ADP validates data items at each stage. In the character stage, data items pass validation if they contain only characters that have been marked as allowed. Data items pass validation at the word stage if they contain only words that have been marked as allowed. In the phrase stage, data items pass validation if they match to a pattern designated as valid. The word and phrase stages only attempt to normalize data items that have passed validation in the previous stage(s). Figure 3 shows the rates of validity achieved with the non-exhaustive rule-setting approach.

In the character stage, validity rates for both datasets are above 99%. These character validation results were achieved following 21 action decisions for the age dataset and 39 action decisions for the data-location dataset in the character normalization stage (see Table 7).

In the word stage, validity rates for both data sets are above 98%. These word validation results were achieved following 94 action decisions for the age dataset and 187 action decisions³ for the data-location dataset in the character normalization stage (see Table 7).

The age dataset's validity rate at the phrase stage is significantly lower than that of the data-location dataset: 83.8% of data items pass phrase validation in the age dataset, while 97.9% of data items pass phrase validation in the data-location dataset. This is the result of a relatively large number of data items that match invalid patterns. As is recorded in the phrase-normalized age dataset file, of the 1019 data items that failed phrase validation, only 105 (1.47% of all data items) failed because they did not match any pattern; all the rest failed because they matched a pattern designated as invalid, a way to intentionally eliminate non-useful data items, like numerical values without units in the age dataset.

³ 186 row-by-row action decisions plus one mass-allow of Protein Data Bank (PDB) identifiers, as described in footnote 1, performed via regular expression selection of the rows containing PDB identifiers.



Numerical exact ages without units (e.g., "7") and age ranges without units (e.g., "8-10") are designated as invalid phrase types because the age dataset contains ages expressed in hours, days, weeks, months, and years, so the units of these data items are unclear, which makes the meaning of the data item ambiguous. From a practical perspective, queries on the age dataset must take units into account. If a study with its subject age listed as "8–10" appeared in the results of a query for studies with a subject age of 8-10 years, it would be troublesome if the "8-10" study was actually performed on mice with an age of 8-10 weeks old. Numerical ages without units are thus not useful for querying. Accordingly, we intentionally invalidate numerical data items without units, like "8–10", so the fact that a significant number of data items failed phrase validation because they matched an invalid pattern is not a poor outcome of normalization, as it represents the elimination of non-useful data.

These phrase validation results were achieved by matching against 16 phrase-type patterns for the age dataset and 12 patterns for the data-location dataset (see Table 7).

It is evident that a relatively low number of user action decisions is sufficient to produce very high rates of validity in at least these two free-text datasets. Notably, in both the character and word stages, reaching similar results (>99% validity in the character stage and >98% validity in the word stage) in the two datasets required only about twice as many action decisions in the data-location dataset as in the age dataset, despite that the former dataset is more than 35 times longer than the latter.

Accuracy of normalized outputs

While robust accuracy-checking procedures are still in development, a preliminary manual review of a randomized subset of data items from each dataset has been conducted in accordance with peer review feedback. These preliminary accuracy tests were performed using 200-line tables in which each line contains an input data item and the normalized output of that data item. The reviewer was instructed to mark all lines where the input did not mean the same thing as the output. In each test, a small, randomized number of intentional procedurally generated errors were included. Table 8 shows the results

Test	Intentional Errors	Unaltered Lines	Intentional Errors Identified	Genuine Errors Identified	Identified Error Rate
Age	9	191	7	0	0%
Data-Location 1	7	193	7	9	3.63%
Data-Location 2	3	197	2	5	2.54%

Table 8 Preliminary manual review results

of this preliminary manual review; "unaltered lines" refer to the number of data items in the sample minus the number of lines with intentional errors, and identified error rate is calculated as the percentage of data items with genuine errors among the unaltered lines.

The manual review of the data-location dataset revealed error-introducing issues with the code that splits data-location strings like "Fig. 1, 2, and 4" into its component parts (e.g., "Fig. 1", "Fig. 2", "Fig. 4") to be parsed as strings individually. The second manual review of the data-location dataset was performed following fixes to issues discovered through the first manual review, and this second round revealed other issues; as of the time of writing, fixes for these issues are in progress and are a high priority. Importantly, every issue found in the data-location dataset in both reviews was a product of errors in the splitter code, an accessory module which is not part of the core normalization toolset; the reviewer found no errors that could be attributed to the core normalization functions.

This preliminary review is small and non-comprehensive. Further evaluation of the accuracy of ADP normalization using more extensive manual review and other methods is an imminent next step. As part of this process, we also intend to test accuracy comparatively between ADP and similar tools.

Extent of change to data items

In the character and word stages, the values in the Levenshtein distance score columns (see Measuring String Change During Normalization above) serve as indicators of the extent to which strings are modified during the normalization process. Figures 4 and 5 show the frequency distributions of Levenshtein distance scores by dataset and stage. Note that the word stage figures for both datasets use a logarithmic scale for clarity.

For the age dataset, Levenshtein distance score frequency graphs show that most data items receive little modification during the character and word stages. The notable spike at a score of 1 in the word stage results from the abundance of age data items with plural units that were normalized to singular; the score of 1 frequently represents the removal of an "s" from "years", "months", or "weeks."

In the data-location dataset, the uniform nature of much of the dataset (namely the > 200,000 lines of HLA Ligand Atlas URLs) produces other spikes in the character stage Levenshtein distance frequency chart. The spike at 9 is one such case. Of the 57,517 data-location data items with a Levenshtein distance score of 9 at the character stage, 91% (52,522) are HLA Ligand Atlas URLs that have paths that a string of 9 uppercase letters (e.g., "https://hla-ligand-atlas.org/peptide/AAAAAQSVY"). The URLs resolve in the same way with lowercase and uppercase letters in that path; the former URL is functionally equivalent to "https://hla-ligand-atlas.org/pepti de/aaaaaqsvy", so normalizing to lowercase does not result in any lost meaning. Levenshtein distance scores at the word stage cluster strongly around 0 for the datalocation dataset, a reflection of the fact that a large portion of the dataset, namely the URLs, received no word normalization.

Levenshtein distance ceases to be a useful metric at the phrase stage, at which it is often desirable to make significant changes to the overall structure of the data item. Straightforward and benign changes like alterations in word order produce high Levenshtein distances. Accordingly, Levenshtein distance scores are not tracked at the phrase stage.

Data-location phrase splitting and phrase-part validity.

Because the data-location dataset included a high number of list-like inputs made up of several individual data locations, the data items in that dataset were put through a splitter script that divided list-like data items so that each output datum referenced exactly one data location (see Data-Location Splitting above).

For this dataset, we calculate additional relevant metrics. Split phrase count (listed in the split_phrase_count column) refers to the total number of outputs split out of an original input data item; e.g., the input item "Table 8 and Fig. 1", which is split into the data items "Table 8" and "Fig. 1", has a split phrase count of 2. Validity rate is the number of valid output data items divided by the split phrase count. A validity rate of 1 means that every output data item that derives from a particular input data item is valid, while a validity rate of 0 means that none of those output data items are valid. Split phrase count and validity



Fig. 4 Levenshtein distance scores by stage, age dataset



Fig. 5 Levenshtein distance scores by stage, data-location dataset

rate (along with all other analytics, like Levenshtein distance scores) are recorded in the phrase-normalized output file exactly once for each input data item so that means and frequency distributions of those metrics are not skewed by the row-count increase that occurs during phrase splitting.

As is evident in Fig. 6, the large number of HLA Ligand Atlas URLs in the dataset concentrate both the split phrase count and validity score around 1, as the URLs are all unsplit and valid. Including URLs, the mean split phrase count is 1.24 (standard deviation 0.92), and the mean phrase validity rate is 1.00 (standard deviation 0.06).

It is noteworthy that the data items with high split phrase counts tend towards high validity rates. It appears that those data items tend to be simple and orderly lists, such as the data item "Figs. 1, 2, 3, 4, Supplementary Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13", which has a split phrase count of 16 and a validity rate of 1.0. Such data items are simpler to split, and their split outputs are individually simpler and more readily matchable to basic phrase patterns than the less uniform lists that occur towards the middle of the split phrase count range, such as "Table 3 and Figs. 1 and 2 and Supporting Information S2 Figure" (split phrase count 4, validity rate 0.75).

When examining only non-URL data items, strong clustering around a validity rate of 1 remains, but with

a more obvious spread of split phrase count values, as is evident in Fig. 7. Excluding URLs, the mean split phrase count is 3.19 (standard deviation 1.91), and the mean phrase validity rate is 0.96 (standard deviation 0.17).

The implementation of data-location phrase splitting achieves very high rates of validity even among the complicated minority made up of non-URL data items.

Discussion

Measuring normalization empirically

The ADP toolset provides several metrics by which a user can measure the extent to which ADP normalization modifies the data, including Levenshtein distance scoring and validation pass/fail rates. These metrics are intended to approximate the degree to which the ADP normalization code improved the overall normality of the data without losing the original string's meaning. However, empirically evaluating the success of the normalization process as a whole remains difficult due to the lack of a clear universal metric for dataset normalization. A useful future direction would be to establish an empirical way to measure degrees of standardization in unstructured datasets; ideally, such a metric would allow comparisons between free-text datasets' spelling accuracy, adherence to grammar, and stylistic consistency.



Fig. 6 Scatter plot of split phrase count and phrase validity rate



Fig. 7 Scatter plot of split phrase count and phrase validity rate, excluding URLs

Evaluating the utility of ADP and other tools

While the age and data-location datasets are distinct enough in size, content, and style to make a case for the flexibility of the ADP rule-setting framework for normalization, its use on these two datasets is not sufficient to demonstrate that ADP is a useful tool for a truly wide range of free-text datasets. Further experimentation with other free-text datasets will be necessary to ensure that ADP normalization is adaptable enough to be used with a diverse range of free-text datasets. Promising future candidates to test ADP's generalizability are the IEDB's other free-text data fields, which would benefit from standardization for the same reasons as the age and data-location fields. Of particular interest is the "dose" dataset, which documents dosages of substances administered as part of an experiment and which contains several thousand distinct lines of free-text data. Dose data items tend to be long and diverse, such as "6 dose(s) of 0.1 mg in saline" and "Four doses of epitope covalently linked to BSA in CFA administered at intervals of 1 week". It appears that this dataset would benefit from several standardization rules applied to the age and data-location datasets, like the conversion of spelled-out numbers to numerals, e.g., converting "Four" to "4" in the second data item above. The dose dataset's complex phrase structures could pose a challenge at the phrase normalization stage, though it may benefit from splitting in a manner like the data-location dataset.

Developing frameworks for testing the accuracy of ADP's outputs compared to other normalization methods is an active priority. ADP's user-dependence is a design feature that was implemented specifically because we hypothesize that it will result in higher precision of normalization results compared to predictive normalization tools, which can struggle with certain contextspecific normalization decisions, like handling instances of "cant" occurring as a synonym of "slang" rather than a misspelling of "can't", that humans can make quickly and accurately [4]. Future testing will likely include evaluating how effectively ADP normalization preserves the meaning of data items throughout the normalization process compared to analogous normalization tools. Performing further accuracy testing, both through more extensive random-sample review as described in the Accuracy of Normalized Outputs section above, as well as through measures like semantic similarity, is a critical next step for this process. Comparison between ADP and large language model-based tools is of particular interest.

Many recent tools for free-text standardization make use of large language models (LLMs) to perform

standardization tasks. One such tool is CleanAgent, which uses an LLM agent to identify the types of data (e.g., phone number, email address, date) in each column of a CSV, write and run Python code to standardize each column's data based on its type, and interact with the user throughout the standardization process [7]. At the time of the writing of this manuscript, the authors were unable to run CleanAgent on either the age or data-location datasets; we have reached out to the developers of CleanAgent about a recurring error message. We aim to do a direct comparison between the free-text normalization outcomes from CleanAgent and ADP in the future. Based on its demonstration publication, CleanAgent appears to be an efficient solution to the problem of standardizing simple data items (e.g., dates or phone numbers), though in that publication, CleanAgent did not identify a datatype for the columns named "AGE" and "weight_" [7], suggesting that it may be less well-suited for standardizing columns of data that lack an obvious standard form. Its use-case is different from ADP's, which requires more of the user's time and effort but is designed to handle data items of that sort.

ChatGPT also been employed for conversions between structured and unstructured data formats in experiments on biomedical data, such as those described in the 2024 analysis by Yoon et al., which evaluated multiple types of transformations between structured data (like ICD codes or tables of laboratory results) and unstructured text [8]. Yoon et al.'s results indicate that the GPT-3.5 model performs better on conversion tasks involving more common data items: in a task involving conversion of ICD-9-CM codes to and from natural language, the model displayed a linear reduction in accuracy among less frequently-used codes, exemplified by an accuracy of 73.3-91.8% for ICD-9-CM codes within the top 1000 by frequency, which decreased to 54.8-91% among ICD-9-CM codes outside of the top 3000 by frequency [8]. Similarly, the model performed better in a task that required identifying prescription drugs listed in unstructured text from discharge summaries when it could match drugs by active ingredient, e.g., matching "paracetamol" with "acetaminophen", rather than strictly adhering to the original terminology [8]. These results suggest that GPT-3.5-based tools are best suited for text standardization on datasets featuring common data, like the particularly frequent ICD-9-CM codes, and datasets for which synonym substitution is acceptable.

The newer GPT-4 model has been shown to outperform GPT-3.5 across multiple domains, indicating that it and other more advanced LLMs may be able to provide better results than GPT-3.5 [8]. Due to the ongoing rapid advancement in the performance of newer LLMs like GPT-40 and DeepSeek, we expect that these contemporary LLMs would outperform older models like GPT-3.5 on data standardization tasks, and future LLMs will likely outperform today's models. As output accuracy is a vitally important feature of standardization processes for biomedical datasets, an error rate of 0 is often necessary. If LLM advancements do indeed result in higher output accuracy, these advancements will further increase the utility of LLMs for biomedical data standardization. Nevertheless, there remains a use-case for standardization tools designed for data with features that tend to decrease LLM performance, like data that does not feature common elements or for which synonym substitution is unacceptable. ADP is designed for this niche, as its non-fully-automated design enables complete and precise control over its outputs regardless of the features of the data itself. The amount of fine-tuning possible within ADP's normalization process is intended to enable high output accuracy. Our preliminary accuracy testing, which resulted in an identified error rate of 0-3.63%, as described in the Accuracy of Normalized Outputs section, suggests that ADP can achieve higher accuracy than the LLM-based techniques for conversion of natural language to structured data that are described in [8], though more work is necessary to fully assess this. The authors intend to conduct further empirical evaluation of the accuracy of ADP and comparable LLM-based normalization tools in the future.

Productive value of results

These datasets contain all distinct age and data-location values recorded in the IEDB, but many of the individual values in these datasets represent thousands of instances of that value. As a result, the effect of normalizing these datasets is multiplied. The IEDB records more than 18 million total age values and more than 21 million total data-location values [5], so applying normalization to these data values in the IEDB will accordingly produce significant improvements to findability and usability of millions of lines of data.

Improving data findability

Using the ADP normalization toolkit, we normalized the age and data-location free-text datasets from the IEDB, two datasets with very different content and normalization needs, in such a way that renders the data in these datasets searchable, findable, and ontologizable in a way that they simply were not before. Standardizing the text in these datasets will enable the forthcoming implementation of dedicated search tools for these datasets in the IEDB. For instance, IEDB users could query for data from experiments on mice less than 28 days old and receive results within that range including ages originally expressed with varying formats and units (e.g., '10–20 days old', '2 days', '24 h', '1–3 wks'). Similarly, by ontologizing categorical age data items like 'juvenile', 'calf', 'foal', 'piglet', or 'child', we can enable searches for pre-adult life stages across species.

The FAIR data principles identify searchability (principle F4) as a critical aspect of data findability, so improving the IEDB's search functionalities is core to the IEDB's effort to improve its overall data FAIRness [9]. The data-location dataset in particular was identified as a promising candidate for work to improve the IEDB's FAIRness in a 2018 analysis of the IEDB's adherence to the FAIR standards [3]. Accordingly, the normalization performed on the data-location dataset using ADP completes that long-standing goal and demonstrates the IEDB's ongoing commitment to improving data FAIRness in immunology.

Enabling ontologization of free-text data

By standardizing the characters, words, and phrase structures in free-text datasets, ADP makes it easier to ontologize those datasets. Several prior publications have illustrated the benefits of linkages between IEDB data and formal ontologies [2, 3, 10, 11]. Already, many IEDB data fields are mapped to terms from a wide range of ontologies, such as the "Organism" field being mapped to NCBI Taxonomy [12] terms and the "Evidence Code" field being mapped to Evidence Ontology [13] terms [14]. By standardizing the terms in use in the age and datalocation datasets, ADP normalization is an effective step towards ontologizing the data in these fields. In particular, promising next steps include the ontologization of units in the age dataset via the Unit Ontology [15] and document parts via the Information Artifact Ontology and Ontology for Biomedical Investigations [16]. Should ADP prove effective on other free-text datasets within and beyond the IEDB, it will make it possible to reap the benefits of ontologization from large amounts of previously underutilized biomedical data.

Conclusions

While further testing is necessary to validate ADP normalization on other datasets, preliminary evaluations of its application to the age and data-location datasets suggest that ADP normalization can produce high rates of output validity in diverse free-text datasets following a relatively low number of user action decisions.

The Immune Epitope Database (IEDB) has made significant efforts over the past several years to improve its adherence to FAIR data standards through improvements to findability and interoperability of its data. Creating linkages with formal ontologies is a pillar of the IEDB's efforts to improve interoperability, but these efforts have been concentrated on standardized datasets. The ability to standardize free-text datasets would enable further FAIRness and more effective utilization of the vast quantities of free-text data in the IEDB. Our preliminary results are promising indications that ADP normalization can standardize free-text datasets efficiently and accurately.

Acknowledgements

We wish to acknowledge the entire IEDB and CEDAR development and curation team.

Authors' contribution

All authors contributed to the conception of this project. S.D. and J.B. designed and developed the ADP software, collected resulting data, and drafted this manuscript. B.P. and J.A.O. advised on the software design and data collection process. R.V. advised on datasets to target for normalization, assisted in collection of input data, and provided substantial feedback on the software design. All authors read and approved the final manuscript.

Funding

Research reported in this publication was supported by the National Institutes of Health contract 75N93019C00001 and grant U24CA248138.

Data availability

All code and data discussed in this manuscript is available in the following GitHub repository: https://github.com/sebastianduesing/adp.

Declarations

Ethics approval and consent to participate Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare no competing interests.

Author details

¹Center for Vaccine Innovation, La Jolla Institute for Immunology, La Jolla, CA 92037, USA. ²Knocean Inc., Toronto, ON M2P 2T3, Canada. ³Department of Medicine, University of California San Diego, La Jolla, CA 92093, USA.

Received: 30 October 2024 Accepted: 25 February 2025 Published online: 22 March 2025

References

- Gandomi A, Haider M. Beyond the hype: big data concepts, methods, and analytics. Int J Inf Manag. 2015;35(2):137–44.
- Vita R, Mahajan S, Overton JA, Dhanda SK, Martini S, Cantrell JR, et al. The Immune Epitope Database (IEDB): 2018 update. Nucleic Acids Res. 2019;47(Database issue):D339-43.
- Vita R, Overton JA, Mungall CJ, Sette A, Peters B. FAIR principles and the IEDB: short-term improvements and a long-term vision of OBOfoundry mediated machine-actionable interoperability. Database. 2018;1(2018):bax105.
- Clark E, Araki K. Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. Procedia -Soc Behav Sci. 2011;1(27):2–11.
- 5. Duesing S. sebastianduesing/adp. 2024. Available from: https://github. com/sebastianduesing/adp. Cited 2024 Jul 1.
- Sproat R, Black AW, Chen S, Kumar S, Ostendorf M, Richards CD. Normalization of non-standard words. Comput Speech Lang. 2001;15(3):287–333.
- Qi D, Wang J. CleanAgent: Automating Data Standardization with LLMbased Agents. arXiv; 2024. Available from: http://arxiv.org/abs/2403. 08291. Cited 2024 Sep 30.

- Yoon D, Han C, Kim DW, Kim S, Bae S, Ryu JA, et al. Redefining health care data interoperability: empirical exploration of large language models in information exchange. J Med Internet Res. 2024;26(1):e56614.
- GO FAIR. F4: (Meta)data are registered or indexed in a searchable resource. Available from: https://www.go-fair.org/fair-principles/f4-metad ata-registered-indexed-searchable-resource/. Cited 2024 Jun 7.
- Vita R, Overton JA, Greenbaum JA, Ponomarenko J, Clark JD, Cantrell JR, et al. The immune epitope database (IEDB) 3.0. Nucleic Acids Res. 2015;43(Database issue):D405–12.
- Vita R, Overton JA, Sette A, Peters B. Better living through ontologies at the Immune Epitope Database. Database J Biol Databases Curation. 2017;18(2017):bax014.
- NCBI Resource Coordinators. Database Resources of the National Center for Biotechnology Information. Nucleic Acids Res. 2017;45(Database issue):D12–7. https://academic.oup.com/nar/article/45/D1/D12/2605705.
- Chibucos MC, Mungall CJ, Balakrishnan R, Christie KR, Huntley RP, White O, et al. Standardized description of scientific evidence using the Evidence Ontology (ECO). Database J Biol Databases Curation. 2014;2014:bau075.
- 14. Vita R, Overton JA, Peters B. Identification of errors in the IEDB using ontologies. Database J Biol Databases Curation. 2018;2018:bay005.
- Gkoutos GV, Schofield PN, Hoehndorf R. The Units Ontology: a tool for integrating units of measurement in science. Database J Biol Databases Curation. 2012;5(2012):bas033.
- Bandrowski A, Brinkman R, Brochhausen M, Brush MH, Bug B, Chibucos MC, et al. The ontology for biomedical investigations. PLoS ONE. 2016;11(4):e0154556.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.