ACS OMEGA

http://pubs.acs.org/journal/acsodf

Article

Using Machine Learning to Parse Chemical Mixture Descriptions

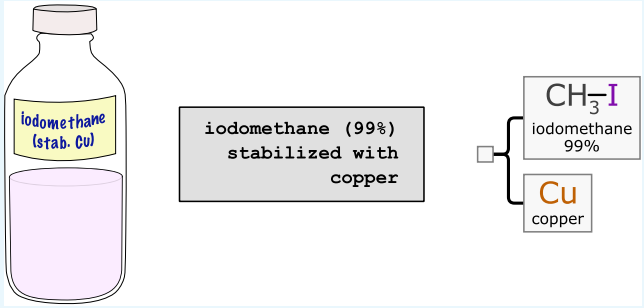
Alex M. Clark,* Peter Gedeck, Philip P. Cheung, and Barry A. Bunin

Cite This: *ACS Omega* 2021, 6, 22400−22409

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Chemical mixtures have recently come to the attention of open standards and data structures for capturing machine-readable descriptions for informatics uses. At the present time, essentially all transmission of information about mixtures is done using short text descriptions that are readable only by trained scientists, and there are no accessible repositories of marked-up mixture data. We have designed a machine learning tool that can interpret mixture descriptions and upgrade them to the high-level *Mixfile* format, which can in turn be used to generate *Mixtures InChI* notation. The interpretation achieves a high success rate and can be used at scale to markup large catalogs and inventories, with some expert checking to catch edge cases. The training data that was accumulated during the project is made openly available, along with previously released mixture editing tools and utilities.



INTRODUCTION

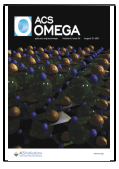
Mixtures is one of the domains of chemistry that is being actively inducted into the realm of informatics. While discrete-molecule cheminformatics has been a staple part of the drug discovery community for decades, using data structures to describe collections of chemicals in a standardized and machine-readable way is a recent development and one that is overdue given that essentially all real-world encounters with a chemical involve mixtures of some kind. We have previously described an open format called *Mixfile*, which is the mixture analog of the de facto industry standard *Molfile*, as well as a suite of open-source tools and generation of the *Mixtures InChI* (MInChI) standard notation, which is layered atop the well-known *InChI* identifier.[1] As part of the preliminary research, we described a proof-of-concept technique for extracting marked-up mixture descriptions from text descriptions, and in this work, we describe a much more sophisticated and robust algorithm for automated text-to-mixture transformation at scale.

The objective of this work is to design an algorithm that can take a single-line text description of a chemical mixture and derive the implied hierarchical composition. The outcome is considered fully correct if each component within the hierarchy correctly identifies the name, parses out any given quantity information,[2] and maps the name to a structure when applicable.

The use case environments are intended to be low-intervention scenarios where expert appraisal and correction is an option but human time is considered to be expensive, whereas computer time is cheap. This includes the interactive single-use scenario, where a user may be pasting a mixture description into an electronic lab notebook and would benefit from a tool that can unpack the description into a machine-readable form, presenting the opportunity to fix the prediction if necessary. A more ambitious use case is the importing of an existing corpus of mixtures that are identified by text description, which is often available from inventory management systems or fine chemical vendor catalogs. The goal in this scenario is to be able to import the overwhelming majority of these mixture descriptions correctly while leaving just a small number of difficult edge cases that require significant intervention.

As a general rule, collections of mixtures used by R&D labs and catalogs tend to follow the trend of being mostly simple with a long tail of increasingly abstruse cases. Most chemical catalogs are full of line items with descriptions that consist of a chemical name and purity (e.g., 1,3,7-trimethylxanthine, 99%) and possibly some circumstantial information that is not part of the primary description. The complexity increases as more components are involved, and various uses of scientific English facilitate many different ways to indicate how much of each component is present and under what circumstances. Other metadata is often present, such as branding information and material form, which we consider to be of secondary importance and do not seek to parse using the current incarnation of our algorithm. In any given collection of mixture descriptions, there are usually some number of unusually

Received: June 24, 2021
Accepted: August 9, 2021
Published: August 18, 2021



https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

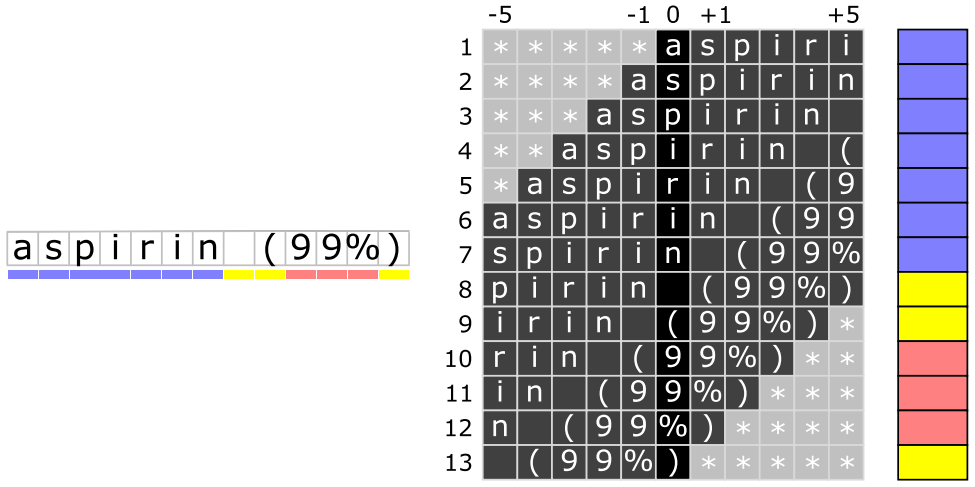


Figure 1. Feature matrix for a short mixture description. Each character is represented as its own row and contributes one datapoint to the model. Cells marked with an asterisk are out of bounds.

written descriptions that are essentially unique and do not have analogous phrasings within any available training set, and so extracting them is challenging. Furthermore, many collections include some number of materials that do not describe the contents: when a mixture is described just using a brand name or a reference number, it is obviously not possible to ascertain the content using a machine learning algorithm. Many collections also include mixtures for which some parts are poorly defined, which is common when biological components are mixed in. There are a variety of strategies for classifying such material components, most of which are dependent on lookup tables.

In this work, we begin with a simple hypothesis:

- for a chemical mixture description, we can tag each character belonging to one of three different classifications—material, quantity, or junk
- this predictive tagging provides enough information to guide a post-processing algorithm for successful partitioning of mixture components

For the remainder of this work, we will describe the construction of the algorithm, its validation, and viable real-world use cases.

METHODS

Machine Learning. The goal of the machine learning model is to predict each character in a mixture description as being one of three types: material, quantity, or junk (which are color-coded as blue, red, and yellow, respectively). Each character of each mixture can be considered as a single input datapoint. The input features described are the character itself and its neighbors, up to a certain radius in either direction, which is illustrated in Figure 1 In this example, a short mixture description, aspirin (99%), is represented as a matrix with 13 rows, one for each character in the text string. For each row, the character focus is shown in column 0, with its 5 neighbors on either side. The columns are fixed positions, and these are treated as input descriptors. Each row is independent, and the output value is the tagged category (i.e., junk, material, or quantity).

To keep the number of input features concise and avoid overtraining, a primary radius of 5 (i.e., 11 characters) is used, and the states are represented using one hot encoding[3] (for all possible ASCII values plus Greek letters and common symbols). An additional input option is used to indicate out of bounds. A larger secondary radius of 25 (i.e., 51 characters) captures the content at a lower resolution, designating each as alphabetical, numeric, whitespace, other, or out of bounds (total of 5 states), which are also one hot encoded. Note that the primary and secondary radii overlap deliberately, which helps resolve some overtraining issues, such as the difference between capital and lower case letters, which can be equivalent or not, depending on the context.

Model building is done by creating a single-layer neural network with an input vector size of 1641, a hidden layer size of 128, ReLU activation, and three output states using the cross-entropy loss function and the Adam optimizer. Implementation was done using Python 3.8 and PyTorch 1.4.[4] Model training proceeds until an accuracy of 99.5% is achieved.[5]

From a machine learning point of view, the neural network model construction is quite simple. Experimentation with using multiple hidden layers or larger layer sizes generally resulted in less predictive models, as well as consuming more space and training more slowly.

Post Processing. Once the model has been used to classify each character, the results are subjected to a series of algorithmic cleaning operations to remove some of the noise. The sequence of tagged characters is then partitioned into a hierarchy by associating material & quantity blocks, honoring nested brackets where applicable.

The cleaning steps are as follows:

(1) Divide the string into blocks separated by whitespace. For any block, if more than 50% of the characters were predicted to be material or quantity, reclassify all of the characters as that.
(2) Any isolated material or quantity character that has no neighbors of the same type is set to junk.
(3) Any whitespace character that is bordered by a material or quantity of the same type (i.e., M_M or Q_Q) is set to that type.
(4) Any consecutive block of a material or quantity is trimmed of punctuation such as commas and semicolons (by reclassifying them as junk).
(5) Any consecutive block of a material or quantity that has imbalanced nesting (due to parenthesis or square

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

bracket characters) is checked for bracket characters at the beginning or end of the block: if removing any of them helps balance the nesting count, it is reclassified as junk.

(6) Any consecutive block of quantity that has no numeric characters is all reclassified as junk.

Once the cleaning is complete, the tags are used as guidelines for the construction of a preliminary tree hierarchy: each node in the tree contains {material, quantity, children}, any of which can be blank. The tree is built up by designating a current head branch, to which new branches or sub-branches are added. Whenever a new branch level is created, it is seeded with a single blank node as a placeholder, which simplifies the implementation. The tagged characters are scanned from left to right and treated as follows:

if the character is tagged as junk:
- if the character is "(" or "[", then insert a child node and make that the new head branch
- else if the character is ″)″ or ″]″, then rewind the *head* branch to its parent
- otherwise, ignore

if the character is tagged as a material or quantity:
- collect all consecutive characters with the same tag into a block
- append a new node after current (with either material or quantity defined with the contents of the block) and make this the new head

The result of this basic treatment is a tree structure that often contains empty branches and typically features material-containing nodes adjacent to quantity-containing nodes. An example is shown in Figure 2. The follow-up step involves

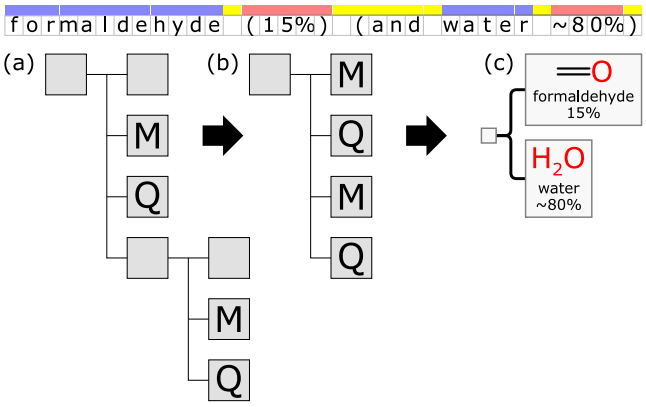


Figure 2. Stepwise construction of a mixture from its tagged description: (a) preliminary tree construction; (b) partially processed version of the initial tree; and (c) final mixture hierarchy with structures and quantities processed.

recursively cleaning up the initial tree, starting with the root branch:

iterate over child nodes:
- if the node has a quantity (but no material), look for the closest prior node that has a material (but no quantity), stop the search if anything with a quantity is found; if successful, merge the nodes (e.g., [M, Q,M] goes to [MQ, M])
- if the above search yielded nothing, look for the closest subsequent node that has a material (but no quantity), stop the search if anything with a quantity is found, and merge the nodes (e.g., [MQ, Q,M] goes to [MQ, QM])
- recursively process each child node
- delete child nodes that are blank
- any child node with more quantity instances than material instances gets flattened into its parent's hierarchy (e.g., [M(Q)] goes to [MQ], [M, M, M, (Q, Q, Q)] goes to [M, M, M, Q, Q, Q]
- if the number of child nodes with just a material is equal to the number of child nodes with just a quantity, pair them up in the same order (e.g., [M_1, M_2, M_3, Q_1, Q_2, Q_3] goes to [M_1Q_1, M_2Q_2, M_3Q_3])

Component Reconstruction. Once the tree processing algorithm described above is complete, the tree structure can be converted directly into the Mixfile data structure, which we have previously described in detail.[1] The material and quantity fields should ideally contain enough information to recreate the structure (if applicable) and to parse out the quantity with its corresponding modifiers and units.

For material names, there are three possibilities: (1) the name refers to a specific chemical that implies a structure; (2) the name refers to a common mixture mnemonic (e.g., "hexanes", "formalin") that implies a sub-branch; and (3) the name refers to a material that is not readily resolvable (e.g., a biologically sourced substance or a brand name).

The first resolution of material names is done by consulting a lookup table, which consists of entries manually created during the preparation of validation sets and those harvested from previously assigned mixtures. Even in cases where structures can be created programmatically, fetching from a lookup table is faster and this is useful for sustained references to commonly occurring chemicals.

Using a lookup table is the only practical option for trivial names of submixtures, and these need to be collated manually. Fortunately, there are relatively few of them in common use: examples include descriptions such as hexanes and xylenes, which are straightforward to describe as an ensemble of specific structures, without being too specific about their actual proportions (which are often not known).

When a name lookup fails, the next fallback is to use a name-to-structure tool, in this case OPSIN.[6] When the name is recognized, the tool produces a SMILES string, which is then converted to a Molfile with two-dimensional (2D) coordinates using RDKit.[7] This combination is highly effective for systematic names, but edge cases tend to accumulate in certain domains, especially outside of the druglike organic subset.

If programmatic name parsing fails, a further fallback is to look up the name in a large inclusive database such as PubChem. This data source is particularly well suited to the task as PubChem endeavors to err on the side of quantity, which is particularly valuable for nonstandard chemical names.

Quantities are parsed using a sequence of regular expressions, which is effective because the number of reasonable ways to express concentrations is limited. The text fragment needs to be converted into an ensemble of values, errors, ranges, units, and relations (=, ∼, <,>, ≤ or ≥). Valid units include moles per volume, grams per volume, and percentages and several related variations. There are a number of common ways to represent these units. Most concentration descriptions follow patterns similar to 99.9%, 10−15%, 1.0 M,

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

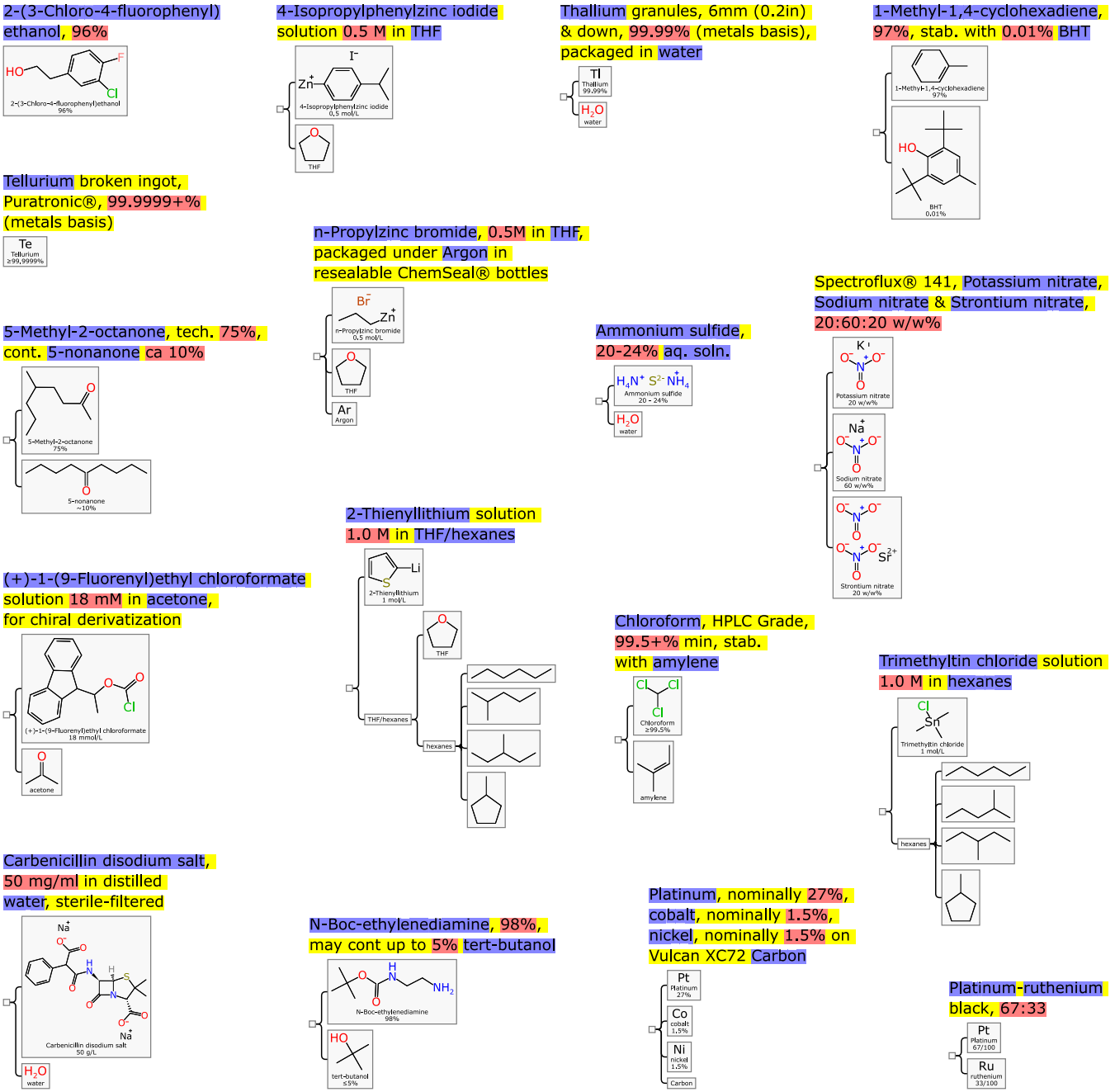


Figure 3. Representative examples of mixture descriptions with their corresponding hierarchical form. The labels are color-coded for material (blue), quantity (red), or junk (yellow).

<5ppm or some variation. The tables of regular expressions and encapsulating algorithms used to unpack quantities are mundane, and the regular expressions are included in the Supplementary Information (Table S1).

Training Data. For method development, training, and validation purposes, a large corpus of mixture names was accumulated over the course of several years. Most of these mixture names were obtained from fine chemicals catalogs and contributed inventory collections and were combined together with their origin identifiers removed. A total of ca. 32 000 unique mixture names were included in the final training set, which are available online.[8] This data includes the original names and the processed machine-readable mixture hierarchy. The mixtures are subjectively representative of fine chemicals collections, i.e., the majority of descriptions are variations on "substance (purity %)", with a long tail of increasingly complex mixtures and/or convoluted linguistic techniques for describing them. Representative examples are shown in Figure 3, which includes easy-to-parse examples that are very common, alongside more tricky cases that have few examples of their kind.

Due to the fact that there is no existing corpus of machine-readable mixture data to train an algorithm for this purpose, it was necessary to devise a bootstrapping technique for marking up the text data that can be easily collected. Each of the mixture descriptions in this collection was marked-up over the course of several years: for early model building exercises, mixtures were manually annotated by tagging each character

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

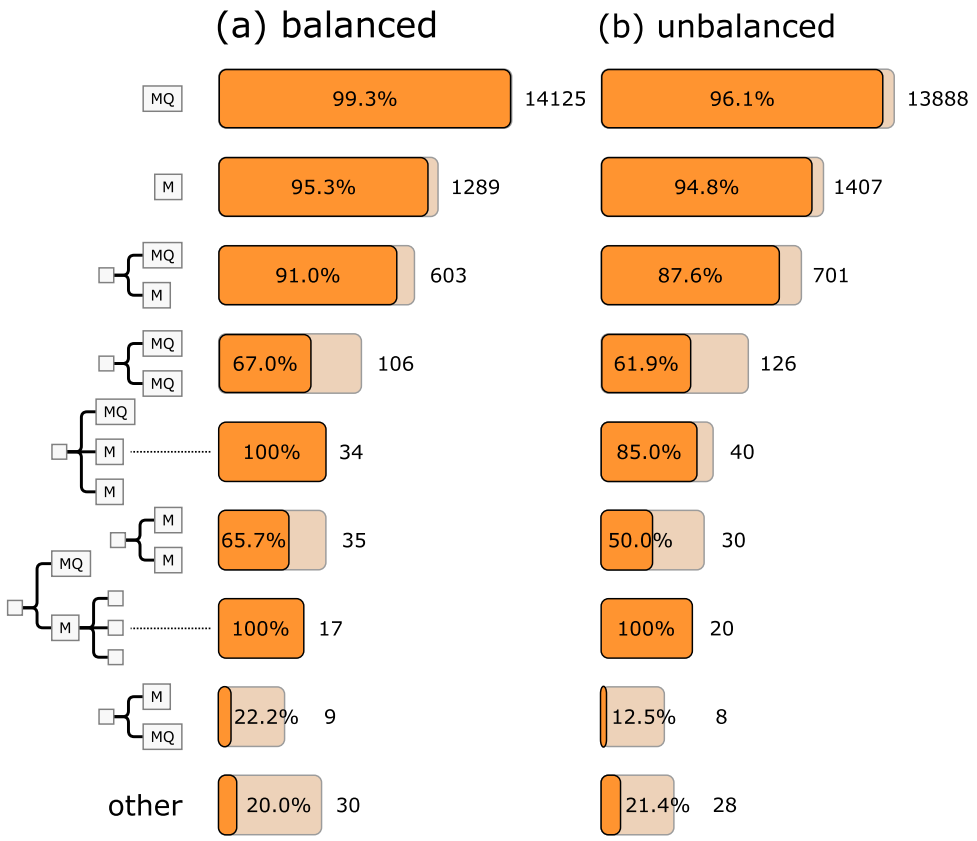


Figure 4. Correct predictions per mixture type based on a balanced training/testing set (a) and unbalanced (b). Percentages show the proportion for which the hierarchy and all names and quantities were correctly recovered. The mixture types are shown iconically, whereby *M* represents a component with just a material name and *MQ* with the material name and quantity.

with mnemonics J, M, or Q (junk, material, or quantity) depending on its role in the mixture hierarchy, with some support from regular expressions, which have since been deprecated. Once sufficient data was available to train a competent machine learning model, the outcome was applied to fresh mixture descriptions, and the post-processing algorithm was refined during this process. The correct results were added to the training set as-is, while the incorrect results were either fixed manually or captured in a later cycle.

A functional interactive user interface was developed for the purpose of applying the model to new data, with the goal of making it convenient to approve successful predictions, and to edit those that are incorrect. This process was executed numerous times, both for generating cross-validation statistics for the current collection and to markup new content. As the method and training data improved, the proportion of mixtures that could be assigned correctly increased, and eventually, all of the mixtures were marked up. One of the validation techniques described in the following section retroactively simulates this bootstrapped training data annotation.

RESULTS AND DISCUSSION

To bluntly evaluate the effect of using one set of mixtures as the training material to predict another set of text data, a set of 32 497 entries each consisting of an original text name (expressed as scientific English referring to components presumed also to be in English) and a fully specified mixture definition was split evenly into two halves. The first study was done by creating a balanced dataset by the following method:

- two mixtures were selected by picking the most similar[9] names, one of which is used to seed the training set and the other is used to seed the testing set
- for all remaining mixtures:
  - the mixture that has the highest overall similarity to the current testing set is moved to the training set
  - the mixture that has the highest overall similarity to the current training set is moved to the testing set
  - repeat until none remain

This simple partitioning algorithm ensures that name similarity is distributed evenly between the two sets, which provides the model training process with the most opportunities to learn patterns that are likely to be found in the testing set. For this reason, the metrics for this exercise can be expected to approximate the best-case scenario.

The model was built using text & mixture definitions from the selected training set and using a lookup database that was created implicitly by extracting all of the component names from the mixtures in the training set.

The prediction metrics overall are high: 15947 (98.1%) of the predictions are completely correct, i.e., the hierarchy, names, and quantities were all extracted correctly from the text, and the name was matched to the correct structure. Foregoing the name-to-structure requirement increases the success rate to 98.8%, which is higher because some of the structures are unable to be programmatically recreated or matched to names already in the lookup list.

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

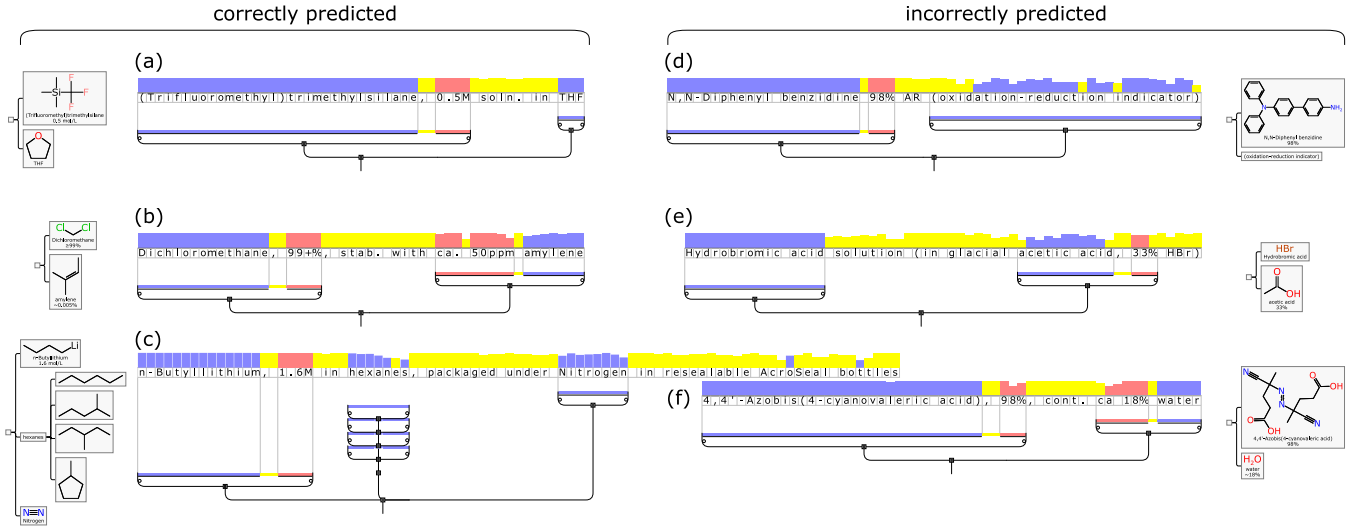


Figure 5. Examples of the correctly (a–c) and incorrectly (d–f) predicted name-to-mixture. Each label is color-coded with its predicted type, and underneath is shown the interpreted hierarchy after the post-processing steps have been applied.

To compare to the ideal case, an explicitly unbalanced set was created using a similar strategy, with reversed polarity: the training and testing sets are seeded with two mixtures that are most different, and the iterative partitioning is done by selecting mixtures that are most similar to those that are already in the set itself rather than its counterpart, encouraging them to diverge. This partitioning strategy seeks to maximize the frequency with which mixtures in the testing set exhibit patterns that are not well represented in the training set. The metrics are reduced relative to the balanced set: 15 424 (94.9%) of the predictions are completely correct or 96.1% without the name-to-structure requirement.

The degree of prediction performance loss is enough that it is certainly well worth updating the model to include mixtures that are as similar as possible to the data of interest. The success rate is still high enough to be optimistic about the effectiveness of building models from stock data and applying them to custom data collections. Better results would be anticipated if the workflow involves importing from fresh data sources in batches and retraining the model periodically.

It should be noted that the long tail of common mixture descriptions tends to skew heavily in favor of single-component mixtures. In this training set, ca. 95% of the examples describe just one component, most of which have an associated concentration. Mixtures for which more than one component is explicitly specified do tend to be more difficult to interpret but having a single component does not mean that extraction is straightforward. Many of these descriptions are accompanied by a large amount of obfuscating text that is not necessarily easy to discern from a material description, and many of them represent quantities in ways that are confusing to a machine learning algorithm (or for that matter to an expert human in many cases).

To provide some insight as to how predictive the method is based on the mixture type, Figure 4 compiles the eight most abundant mixture hierarchy patterns in the test sets (with the rest lumped into *other*) and their successful prediction rates for both the balanced & unbalanced schemes.

Several specific examples are shown in Figure 5: in each case, the incoming description of the mixture is shown, and the predicted tags are shown above, color-coded as yellow (junk), blue (material), or red (quantity). The height of each bar is proportional to the certainty of the prediction for each character. Underneath the description is shown the hierarchy, which is obtained from applying the post-processing algorithm to the predicted tags, which is also color-coded by type. To the side is shown the final assembly of the mixture hierarchy. The first three cases (5a, 5b, 5c) are correctly interpreted, whereas as the other three (5d, 5e, 5f) are examples of when the prediction did not give the right interpretation.

(a) The first component (*trifluoromethyltrimethylsilane*) is correctly tagged by the model, as is the concentration (*0.5M*), which is parsed into an internal representation and assigned to the previous material. The separator text (*soln. in*) is tagged as junk to be ignored, and the *THF* solvent is identified as the second material.

(b) The two components *dichloromethane* and *amylene* are correctly tagged, as are the concentrations *99+%* and *ca. 50ppm*, which are correctly parsed. Because the sequence contains two material and two quantity blocks, they are paired with each other.

(c) The three primary components are *n-butyllithium*, *hexanes*, and *nitrogen*. For the *hexanes* component, one of the characters was mispredicted by the model (the letter *e* was tagged as junk) but this was corrected during the post-processing step. Unlike the other two components, *hexanes* was matched to a predefined lookup list, which lists 4 of the main components of this common mixture, which are in turn represented in the resulting hierarchy. The presence of *nitrogen* is represented in the mixture without any designation of the fact that it occupies the volume above the liquid layer: there is not currently a way to record this distinction but it will be addressed in future work

(d). The *N,N-diphenyl benzidine* component and its concentration are correctly recognized but the model falsely predicts that the term *oxidation–reduction indicator* is the name of a material, whereas it is actually auxiliary information. This could be corrected by having more examples in the training set where this information is specified as junk.

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400–22409

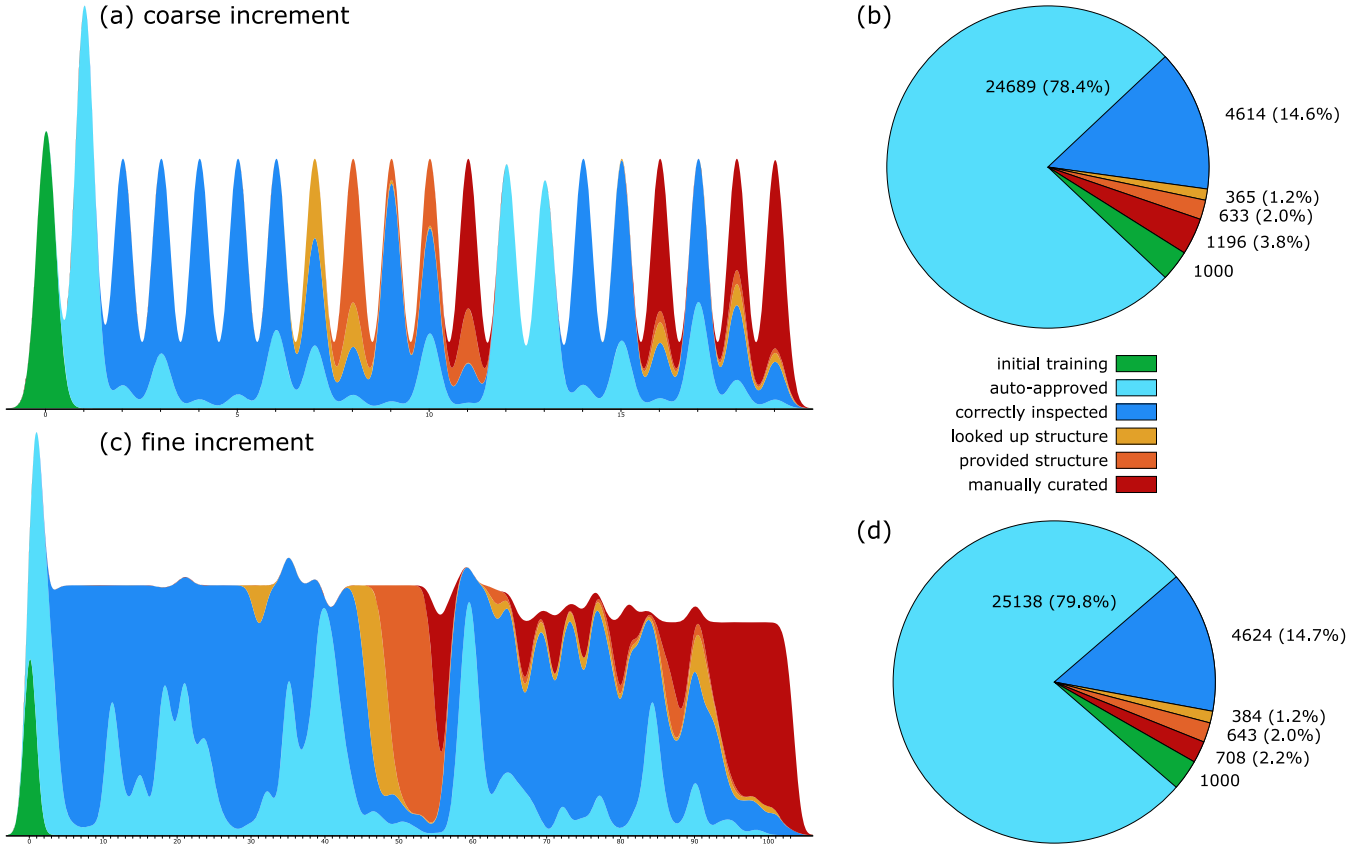


Figure 6. Progressive assignment of mixtures from text descriptions, color-coded by type as cumulative sum charts (a and c) for two similar experiments using larger and smaller batch sizes, respectively. For clarity, heights are plotted on a log scale with a Gaussian blur. The pie charts (b and d) show the overall totals.

(e) The two components *hydrobromic acid* and *acetic acid* are identified correctly, but the concentration (*33%*) is listed after *acetic acid*, alongside a mnemonic shorthand that indicates to the reader that it actually refers to the former rather than the latter, and so the algorithm assigns the quantity to the wrong material. This level of perception nuance beyond what the post-processing algorithm is capable of.

(f) The two components are identified correctly and at a glance it appears that the concentrations are as well. However, the designation of *98%* for *4,4′-azobis(4-cyanovaleric acid)* most likely refers to its purity, rather than its proportion in the overall solution, and so the marked-up mixture description produced by the algorithm is misleading.

A more in-depth validation exercise involves a progressive study, which can be thought of as a retroactive simulation of how the training data was originally created, and also how new data can be marked up by leveraging pre-existing training data. The simple iteration loop can be stated as:

- select some number of entries to use as an initial training set
- build a model and apply it to all remaining mixture instances
- move some of the predicted mixtures to the training set
- repeat

The exercise becomes more interesting with the introduction of a new concept: it is possible to devise a very conservative algorithm to identify when a mixture prediction is so obviously correct that it is not necessary to ask a human expert to verify it. The basic criterion for meeting this condition is that the mixture hierarchy is uncomplicated, all of the components have names that are successfully resolved as structures, and that any residual "junk" characters are essentially just punctuation. With these criteria in place, the rate of false positives is sufficiently close to zero to justify automatically marking such cases as correct. While these high-confidence results are a subset of the overall correct predictions, the fact that they can be detected automatically means that the results from each train/predict cycle contain some number of results that can be automatically promoted to the training set for the next iteration. The next iteration therefore has a larger, richer, and more diverse training set and therefore may be able to correctly observe more patterns within the remaining content, and so some of these results may also turn out to be high-confidence predictions.

In this way, it is possible to run multiple cycles of training set enrichment without any expert intervention. At some point, however, the number of high-confidence predictions will slow down or stall. The next fallback is to simulate the intervention of an expert and pick out some number of results that were predicted correctly. It is not necessary to go through all of the remaining predictions each time, since intervention to augment the training data improves the model, which can have a spill-over effect of dragging previously incorrectly predicted mixtures into the correct category, and some of these may qualify for automatic approval, thus not requiring human intervention.

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

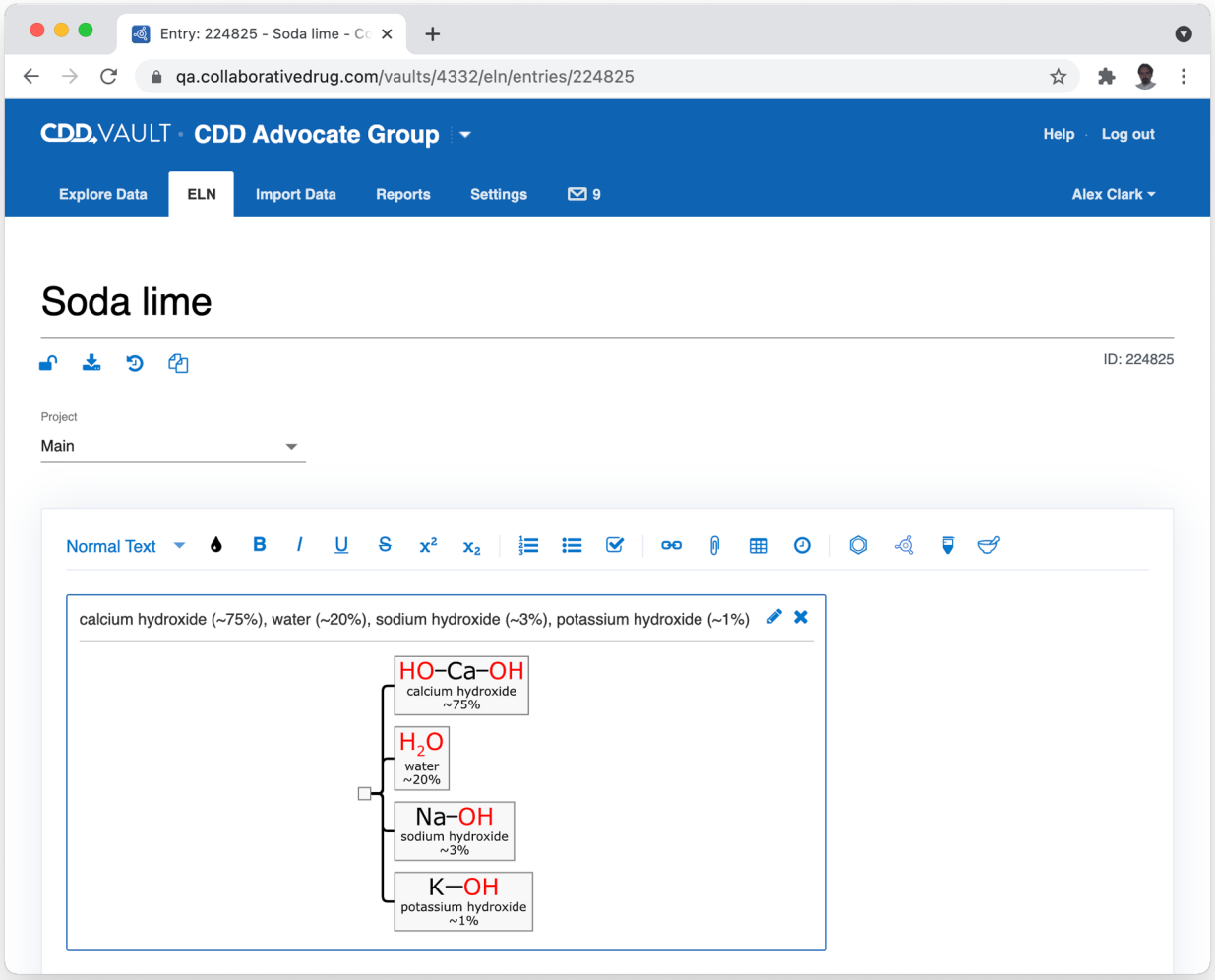


Figure 7. Inline mixtures implemented in CDD Vault ELN: the data is created using a combination of user editing, database lookup, and the machine learning algorithm described in this work. It is stored as a Mixfile and rendered on demand within the web page.

This process was run under two sets of conditions shown in Figure 6: in each case, a set of 1000 mixtures was selected as the seed, and these were chosen to have maximum name similarity to one another to make the evaluation challenging, i.e., minimal overlap with mixtures in the testing set. In case 6a, a minimum of 500 mixtures was assimilated into the training set each iteration: all confident predictions were added, and then if any more were required to make up the quota, these were taken from correct predictions first, followed by those that would be correct if the right structures could be found by a PubChem name search, followed by those for which the structures need to be provided. If the threshold was still not met, mixtures for which the prediction was wrong were chosen.

This triage process simulates the level of user involvement needed to import new content and mark it up to a machine-readable form: the amount of effort needed to confirm a correct prediction is small but in aggregate is not negligible; fixing a mixture by providing structures is moderately labor intensive and can usually be accelerated with the help of name lookup, whereas editing or curating the specific mixture hierarchy from scratch is the most time consuming.

The peaks for each assignment iteration in Figure 6a are color-coded according to the method used to select the mixtures to move from the testing set into the training set for the next round. The heights are plotted in log scale for clarity, while the corresponding pie chart (Figure 6b) shows the proportional scale for the whole sequence. At the completion of 19 iterations, 24689 mixtures (78.4%) in the test set were able to be assigned confidently, without requiring a user to check them. 4614 mixtures (14.6%) were predicted correctly and user-approved, while smaller numbers were ushered in by providing missing structures. 1196 (3.8%) of the mixtures required more significant user intervention. This whole process simulates growing a database from 1000 low-diversity mixtures to around 32 000 in moderate-sized increments.

A finer-grained simulation is shown in Figure 6c,d: rather than adding 500 mixtures per iteration, the batch size is dropped to 100 when there are sufficient correct predictions or 50 if it is necessary to draw from the pool of the incorrectly assigned results, i.e., the more expensive cases. Because the model building/prediction gets more opportunities to learn from additional patterns and then leverage these for the

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409

remaining mixtures in the test set, the number of manually entered mixtures drops significantly, to 708 (2.2%) down from 1196 (3.8%) in the coarse set. This increment size is not necessarily suitable for practical scenarios, since more than 100 iterations were required to mark up the entire collection, and the number of times that a simulated expert needs to confirm/reject predictions is higher than when larger batches are used. Nonetheless, it does bolster the claim that increasing the amount of pattern induction between iterations significantly improves prediction performance, and it can be considered to be a simulation of a real-world scenario where small batches of mixtures are added to a growing inventory database.

CONCLUSIONS

The method we described is suitable for converting chemical mixture descriptions into a marked up form that is appropriate for informatics purposes. It is useful on an ad hoc basis when a single-mixture description needs to be inserted into an electronic lab notebook entry and at scale when marking up a legacy inventory management system. The success rate for the simple descriptions that make up the large majority of fine chemicals is extremely high, while the performance decreases upon venturing further out into the long tail of complex mixtures. Much of the reduction in the recall rate is due to the use of ambiguous or unconventional description styles with few if any examples that can be used to train upon.

The mixture data structure that is created by the machine learning method is based on the openly available Mixfile format, which is accompanied by open source tools for editing and manipulating the content.[1] The marked-up data used for training these methods has also been made openly available. The Mixfile format, and the data created using it, has been designed intentionally to serve as the first of many upstream sources for the Mixtures InChI (MInChI) notation.[10] The data that has been generated for training purposes to fine-tune the text-to-mixture algorithm is also being used to assemble a validation set for the MInChI project.

The near-term objective for this toolset is to ramp up the quantity of data that has been marked up into machine-readable mixture form, both public and private. This is a key component of the community-building strategy that we have outlined in various presentations:[11] until there is a critical mass of mixture data that is ready to use for informatics purposes, there is relatively little incentive for the typical user to take the time to learn how to create new content in this form, even if they have suitable tools available. By providing the managers of inventories the ability to markup all of their existing content with a relatively small amount of effort, numerous kinds of further data analyses become possible right away, and scientists are informed by a precedent for proper data creation. An early example of this being implemented in a commercial context is shown in Figure 7, which shows the CDD Vault[12] electronic lab notebook with an inserted Mixfile object rendered graphically. In this case, the 4 indicated components of soda lime were created using the hierarchical format in which each has a structure, name, and concentration. The software has automatically labeled the mixture with a text label that describes the contents, which is essentially the inverse of the method described in this work. Creating text labels is far more straightforward than interpreting the text.

A longer-term goal of this machine learning project is to keep pace with developments in the InChI community for representing more complicated substances, which step outside of the realm of the well-defined small molecules that make up most fine chemicals. These include polymers,[13] nanomaterials,[14] biomaterials,[15] enumerative sets,[16] reaction products,[17] and many other scenarios where some facts are known about the nature of a component but for various reasons it is not possible to represent it as an atomic connection table. Establishment of a suitable data structure for describing these substances is a subject of active research, for which several domain-specific standards are expected to emerge in the near future. This is particularly relevant for a category of substances often referred to as UVCBs (unknown or variable composition, complex reaction products, or biological materials).[18] These substances are of paramount importance for regulation and safety purposes, and yet they are commonly stored in databases as hundreds of thousands of instances of descriptive text, which is intractable to any form of informatics. It is our intention to contribute to efforts to develop both the representation capabilities and machine learning models so that we can also markup these high-difficulty cases. We also intend to investigate interoperability with the FDA structured product labeling (SPL) format, which can be used to describe complex mixtures such as formulations, along with a substantial amount of regulatory metadata.[19]

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsomega.1c03311.

Regular expressions used for parsing quantities (Table S1) (PDF)

AUTHOR INFORMATION

Corresponding Author

Alex M. Clark − *Collaborative Drug Discovery, Inc., Burlingame, California 94010, United States;* orcid.org/0000-0002-3395-4666; Email: alex@collaborativedrug.com

Authors

Peter Gedeck − *Collaborative Drug Discovery, Inc., Burlingame, California 94010, United States*

Philip P. Cheung − *Collaborative Drug Discovery, Inc., Burlingame, California 94010, United States*

Barry A. Bunin − *Collaborative Drug Discovery, Inc., Burlingame, California 94010, United States*

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.1c03311

Notes

The authors declare the following competing financial interest(s): Collaborative Drug Discovery is creating commercial products for handling mixtures, in addition to contributing to open source, open data and community efforts to encourage adoption of mixture informatics.

The input data used for this method is available from GitHub.[8]

ACKNOWLEDGMENTS

This research was supported by the National Institutes of Health grant 2R44TR002528-02. We thank Leah R. McEwen for initiating the Mixtures InChI project within IUPAC & the InChI Trust and codifying the format specification.

REFERENCES

(1) Clark, A. M.; McEwen, L. R.; Gedeck, P.; Bunin, B. A. Capturing mixture composition: an open machine-readable format for representing mixed substances. *J. Cheminf.* **2019**, *11*, No. 33.

(2) Mixture components are generally expressed as having *concentrations*, which implies a proportionality with other components.

(3) One hot encoding is the conventional way to represent categories for which one choice is valid. For 3 states the target values are [1, 0, 0], [0, 1, 0] and [0, 0, 1].

(4) https://pytorch.org (accessed Apr 21,2021).

(5) Accuracy counted as the fraction of input datapoints for which the specified tag is given the highest value. The calculation is done using the training data, hence the high minimum threshold

(6) (a) Lowe, D. M.; Corbett, P. T.; Murray-Rust, P.; Glen, R. C. Chemical Name to Structure: OPSIN, an Open Source Solution. *J. Chem. Inf. Model.* **2011**, *51*, 739−753. (b) https://opsin.ch.cam.ac.uk (accessed Apr 21, 2021).

(7) (a) Bento, A. P.; Hersey, A.; Félix, E.; Landrum, G.; Gaulton, A.; Atkinson, F.; Bellis, L. J.; De Veij, M.; Leach, A. R. An open source chemical structure curation pipeline using RDKit. *J. Cheminf.* **2020**, *12*, No. 51. (b) https://www.rdkit.org (accessed April 21, 2021).

(8) https://github.com/cdd/mixtures/blob/master/reference/trainingdata2021.json.gz(accessed April 21, 2021).

(9) For performance reasons, similarity is determined by counting the number of common characters at the beginning of each string, e.g. [benzene, benzoin] would have a value of 4. For subsets with the same number of initial characters in common, the Levenshtein metric is computed.

(10) https://github.com/IUPAC/MInChI (accessed April 21,2021).

(11) Four online presentations can be found at (a) https://www.youtube.com/watch?v=PcAJ4HoRnFU. (b) https://www.formulation.org.uk/f4p1programme/253-past/2020/form4p1/783-form4p1-clark.html. (c) https://www.youtube.com/watch?v=0ILc0owuEzQ (starting at 1:05:00). (d) https://nci.rev.vbrick.com/#/videos/43d16f38-cb2e-4245-ab46-31089882d9a0 (starting at 2:22:00),(accessed June 17, 2021).

(12) http://collaborativedrug.com accessed June 17, 2021).

(13) (a) Lin, T.-S.; Coley, C. W.; Mochigase, H.; Beech, H. K.; Wang, W.; Wang, Z.; Woods, E.; Craig, S. L.; Johnson, J. A.; Kalow, J. A.; Jensen, K. F.; Olsen, B. D. BigSMILES: A Structurally-Based Line Notation for Describing Macromolecules. *ACS Cent. Sci.* **2019**, *5*, 1523−1531. (b) Adams, N.; Winter, J.; Murray-Rust, P.; Rzepa, H. S. Chemical Markup, XML and the World-Wide Web. 8. Polymer Markup Language. *J. Chem. Inf. Model.* **2008**, *48*, 2118−2129.

(14) Lynch, I.; Afantitis, A.; Exner, T.; Himly, M.; Lobaskin, V.; Doganis, P.; Maier, D.; Sanabria, N.; Papadiamantis, A. G.; Rybinska-Fryca, A.; Gromelski, M.; Puzyn, T.; Willighagen, E.; Johnston, B. D.; Gulumian, M.; Matzke, M.; Green Etxabe, A.; Bossa, N.; Serra, A.; Liampa, I.; Harper, S.; Tämm, K.; Jensen, A. C. Ø.; Kohonen, P.; Slater, L.; Tsoumanis, A.; Greco, D.; Winkler, D. A.; Sarimveis, H.; Melagraki, G. Can an InChI for Nano Address the Need for a Simplified Representation of Complex Nanomaterials across Experimental and Nanoinformatics Studies? *Nanomaterials* **2020**, *10*, 2493.

(15) Zhang, T.; Li, H.; Xi, H.; Stanton, R. V.; Rotstein, S. H. HELM: A Hierarchical Notation Language for Complex Biomolecule Structure Representation. *J. Chem. Inf. Model.* **2012**, *52*, 2796−2806.

(16) *Proposal description*. http://www-jmg.ch.cam.ac.uk/inchi/Variable_InChI.pdf (accessed April 21, 2021).

(17) Grethe, G.; Blanke, G.; Kraut, H.; Goodman, J. M. International chemical identifier for reactions (RInChI). *J. Cheminf.* **2018**, *10*, No. 22.

(18) (a) Vermeulen, R.; Schymanski, E. L.; Barabási, A.-L.; Miller, G. W. The exposome and health: Where chemistry meets biology. *Science* **2020**, *367*, 392−396. (b) Escher, B. I.; Stapleton, H. M.; Schymanski, E. L. Tracking complex mixtures of chemicals in our changing environment. *Science* **2020**, *367*, 388−392.

(19) https://www.fda.gov/industry/fda-resources-data-standards/structured-product-labeling-resources (accessed July 31,2021).

https://doi.org/10.1021/acsomega.1c03311
ACS Omega 2021, 6, 22400−22409