doi.org/10.26434/chemrxiv.11448177.v1

Automated Extraction of Chemical Synthesis Actions from Experimental Procedures

Alain C. Vaucher, Federico Zipoli, Joppe Geluykens, Vishnu H Nair, Philippe Schwaller, Teodoro Laino

Submitted date: 24/12/2019 • Posted date: 26/12/2019

Citation information: Vaucher, Alain C.; Zipoli, Federico; Geluykens, Joppe; Nair, Vishnu H; Schwaller, Philippe; Laino, Teodoro (2019): Automated Extraction of Chemical Synthesis Actions from Experimental Procedures. ChemRxiv. Preprint. https://doi.org/10.26434/chemrxiv.11448177.v1

Experimental procedures for chemical synthesis are commonly reported in prose in patents or in the scientific literature. The automatic extraction of the details necessary to reproduce and validate a synthesis in a chemical laboratory is quite often a tedious task, requiring extensive human intervention. Here, we present a method to convert unstructured experimental procedures written in English to structured synthetic steps (action sequences) reflecting all the steps needed to successfully conduct the corresponding chemical reactions. To achieve this, we design a set of synthesis actions with predefined properties and a deep-learning transformer-based model to convert experimental procedures to a textual representation of action sequences. The model is first pretrained on vast amounts of data generated automatically with a custom rule-based natural language processing approach, and then refined on a smaller set of manually annotated samples. Predictions on our test set resulted in a perfect match of the full action sequence for 64.5% of sentences.

File list (4)

Automated extraction of chemical synthesis actions from experimental procedures

Alain C. Vaucher,[*,†,‡] Federico Zipoli,[†,‡] Joppe Geluykens,[†] Vishnu H. Nair,[†] Philippe Schwaller,[†] and Teodoro Laino[†]

†*IBM Research, Zurich, Switzerland*

‡*Main contributors*

E-mail: ava@zurich.ibm.com

Abstract

Experimental procedures for chemical synthesis are commonly reported in prose in patents or in the scientific literature. The automatic extraction of the details necessary to reproduce and validate a synthesis in a chemical laboratory is quite often a tedious task, requiring extensive human intervention. Here, we present a method to convert unstructured experimental procedures written in English to structured synthetic steps (action sequences) reflecting all the steps needed to successfully conduct the corresponding chemical reactions. To achieve this, we design a set of synthesis actions with predefined properties and a deep-learning transformer-based model to convert experimental procedures to a textual representation of action sequences. The model is first pretrained on vast amounts of data generated automatically with a custom rule-based natural language processing approach, and then refined on a smaller set of manually annotated samples. Predictions on our test set resulted in a perfect match of the full action sequence for 64.5% of sentences.

Introduction

In chemistry like in other scientific disciplines, we are witnessing the growth of an incredible amount of digital data, leading to a vast corpus of unstructured media content — including articles, books, images and videos — rarely with any descriptive metadata. While scientists developed several technologies for analyzing and interacting with unstructured data, quite often these solutions rely on identifying and utilizing inherent rules typical of each data item at the cost of an important human effort. Currently, the processing of unstructured data is pivotal to the work of many scientists: it transforms this data into a structured form that can be used for different types of applications, from deep search to automation.

The availability of structured chemical data is especially important for automation due to the latest reviving use of robots in the context of organic synthesis.[1–4] Structured data is also important to stimulate the design of predictive models for optimizing reaction procedures and conditions, similar to the success of the AI-guided reaction prediction schemes[5–8] for organic molecules.

In fact, although some simple organic reaction data are widely presented in well-structured and machine readable format, this is not the case for the corresponding chemical reaction procedures which are reported in prose in patents and in scientific literature. Therefore, it is not surprising if their conversion into a structured format is still a daunting task with only few attempts made for the extraction of organic chemistry procedures,[3] and with the most successful ones into the more simple domain of synthetic inorganic chemistry.[9–11] As a consequence, the design of an automated conversion from unstructured chemical recipes for organic synthesis into structured ones is a desirable and needed technology.

Ultimately, with such an algorithm, a machine could ingest an experimental procedure and automatically start the synthesis in the lab, provided that all the necessary chemicals are available. Also, if applied to a large collection of experimental procedures, the conversion to structured synthesis actions could prove interesting for the analysis of reaction data, and could facilitate the discovery of patterns and the training of machine learning models for

new organic chemistry applications.

In this work, we focus on the conversion of experimental procedures into series of structured actions, with an emphasis on organic chemistry. To do so, we first identify general synthesis tasks covering most of the operations traditionally carried out by organic chemists. We implement and discuss several computational approaches for the extraction of such structured actions from experimental procedures. Rule-based models represent a good starting point for this endeavor, but they are quite sensitive to the formulation of the rules and to noise in the experimental procedures, such as typing errors or grammar mistakes.[3] We therefore introduce a deep-learning model to translate experimental procedures into synthesis actions. We pretrain it on data generated with rule-based models and refine it with manually annotated data.

In doing so, our goal is for the sequence of actions to correspond to the original experimental procedure as closely as possible, with all the irrelevant information discarded. This means that an extracted action sequence contains, in principle, all the details that a bench chemist would require to conduct a reaction successfully either by humans or with the help of robotic systems.

Retrieving information from the chemistry literature has received a lot of attention over the last decades (see, for instance, Refs. 12–17). The automated extraction and analysis of synthesis procedures has so far been studied mainly in the context of inorganic materials.[9–11] For instance, Mysore et al. apply text-mining tools to convert synthesis procedures to action graphs.[9] Otherwise, one of the predominant goals is to mine information from patents, papers and theses, and save it as structured data in databases in order to make chemical knowledge searchable and enable queries about materials or properties. Due to the complex syntax of chemical language, a lot of effort is put into the development of methods for named entity recognition,[13,17–20] which allows the detection of compounds, actions, or experimental conditions in texts from the chemistry literature. The Reaxys database,[21] which contains reaction data (such as reagents, solvents, catalysts, temperatures, and reaction duration)

extracted from publications and patents, is also worth mentioning in this context. The focus of the present work is different: instead of scanning texts in search of relevant pieces of information, we focus on experimental procedures, which we convert, as a whole, into a structured, automation-friendly format.

Recently, Cronin and co-workers developed a robotic system able to perform organic synthesis autonomously,[3] requiring a synthetic scheme described in the so-called chemical descriptive language (XDL). They implement a rudimentary tool for translating a given procedure into XDL that follows the identification of key entities in the text and assembling the corresponding list of operations. This approach is exposed to linguistic challenges that makes a correct interpretation difficult. As a consequence, creating the XDL schemes remains largely manual. Here, we present a robust model to convert experimental procedures into structured action sequences that do not require human verification. Our deep-learning model for the extraction of action sequences is available free of charge on the cloud-based IBM RXN for Chemistry platform.[22]

Results

Synthesis Actions

The experimental procedures we consider in this work represent single reaction steps. To conduct the full synthesis of a molecule, several such reaction steps are combined. The following is an example of a typical experimental procedure that is to be converted to automation-friendly instructions:

> *To a suspension of methyl 3-7-amino-2-[(2,4-dichlorophenyl)(hydroxy)methyl]-1H-benzimidazol-1-ylpropanoate (6.00 g, 14.7 mmol) and acetic acid (7.4 mL) in methanol (147 mL) was added acetaldehyde (4.95 mL, 88.2 mmol) at 0° C. After 30 min, sodium acetoxyborohydride (18.7 g, 88.2 mmol) was added. After 2 h, the reaction mixture was quenched with water, concentrated in vacuo, diluted*

with ethyl acetate, washed with aqueous sodium hydroxide (1 M) and brine, dried over sodium sulfate, filtered and concentrated in vacuo. The residue was purified by column chromatography on silica gel eluting with a 10-30% ethyl acetate/n-hexane gradient mixture to give the title compound as a colorless amorphous (6.30 g, 13.6 mmol, 92%).

From such an experimental procedure, our goal is to extract all relevant information to reproduce this reaction, including details about work-up. The structured format into which we convert this information consists of a sequence of synthesis actions. It is to be noted that restricting syntheses to the sequential execution of actions prevents us from supporting non-linear workflows. However, such branched synthesis procedures are rare when considering single reaction steps. Furthermore, they can partly be remedied by the choice of actions, as will be explained below.

The predefined set of synthesis actions must be flexible enough to capture all the information necessary to conduct the chemical reactions described in experimental procedures. The actions we selected are listed in Table 1. Each action type has a set of allowed properties. For instance, the `PH` action can be further specified by a substance, a quantity, and/or a pH value (and nothing else). The properties allowed for each action type are listed and explained in the Supplementary Note 1 and Supplementary Table 1.

Most action types listed in Table 1 correspond to actual synthesis operations with direct equivalents in the wet laboratory. We note that drying and washing, in organic synthesis, correspond to different operations depending on their context. In particular, the additional properties attached to the two types of drying are different and we therefore define two action types for drying, `DryInVacuum` and `DryWithMaterial`. `MakeSolution` describes the preparation of a separate solution. This enables us to support experimental procedures that require solutions or mixtures to be prepared separately for use in another action. Accordingly, `MakeSolution` is important in ensuring the compatibility with a linear sequence of actions, by avoiding the necessity to consider multiple reactors in an action sequence. We ignore

Table 1: Action types available for information extraction from experimental procedures.

Action name	Description
`Add`	Add a substance to the reactor
`CollectLayer`	Select aqueous or organic fraction(s)
`Concentrate`	Evaporate the solvent (rotavap)
`Crystallize`	Re-crystallize a solid from a solvent
`Degas`	Purge the reaction mixture with a gas
`DryInVacuum`	Dry a solid under vacuum
`DryWithMaterial`	Dry an organic solution with a desiccant
`Extract`	Transfer compound into a different solvent
`Filter`	Separate solid and liquid phases
`MakeSolution`	Mix several substances to generate a mixture or solution
`Microwave`	Heat the reaction mixture in a microwave apparatus
`Partition`	Partition the reaction mixture by adding two immiscible solvents
`PH`	Change the pH of the reaction mixture
`PhaseSeparation`	Separate the aqueous and organic phases
`Purify`	Purification (chromatography)
`Quench`	Stop reaction by adding a substance
`Reflux`	Reflux the reaction mixture
`SetTemperature`	Change the temperature of the reaction mixture
`Sonicate`	Agitate the solution with sound waves
`Stir`	Stir the reaction mixture for a specified duration
`Wait`	Leave the reaction mixture to stand for a specified duration
`Wash`	Wash (after filtration, or with immiscible solvent)
`Yield`	Phony action, indicates the product of a reaction
`FollowOtherProcedure`	The text refers to a procedure described elsewhere
`InvalidAction`	Unknown or unsupported action
`NoAction`	The text does not correspond to an actual action

information about glassware and apparatus on purpose, as this is largely imposed by the availability of equipment or the scale of the reaction, and the reaction success should not depend on it.

A few action types do not actually correspond to laboratory operations, but are convenient when retrieving information from experimental procedures. The `FollowOtherProcedure` action type is selected when the text refers to procedures described elsewhere, in which case no actual actions can be extracted. `NoAction` is assigned to text that does not relate to a synthesis operation, such as nuclear magnetic resonance (NMR) data, sentences describing

the physical properties of the reaction mixture, or procedures in other languages than English. `InvalidAction` indicates that a text fragment is relevant but cannot be converted to one of the actions defined above.

When determining the actions corresponding to an experimental procedure, it is important to consider that some actions are implicit. For instance, in the sentence *"The organic layer was dried over sodium sulfate"*, the phase separation and collection of the organic layer is implicit (no verb) and will result in a `CollectLayer` action preceding `DryWithMaterial`. Similarly, *"23 g of aluminum chloride in 30 mL of dichloroethane was heated to 50 °C."* corresponds to three actions (`MakeSolution`, `Add`, `SetTemperature`) although the sentence contains only one verb (*"heat"*).

A single action type may cover a wide range of formulations present in experimental procedures. For instance, an `Add` action can be expressed using the English verbs "add", "combine", "suspend", "charge", "dilute", "dissolve", "mix", "place", "pour", and "treat", among others. As an additional example, a `Concentrate` action can be described in terms of concentrating a solution, evaporating a solvent, as well as removing a solvent or distilling it off.

Furthermore, an English verb may correspond to different actions depending on its context. For instance, "heat" may, on the one hand, indicate a punctual change in temperature for subsequent actions, or, on the other hand, inform that the reaction mixture should be heated for a specified duration. In the former case, we convert it to a `SetTemperature` action, and in the latter case to a `Stir` action. Another example is the verb "remove", which may relate to `Concentrate` when associated with a solvent or to `Filter` in the context of a filtration.

It is important to note that there can be multiple ways to assign actions to some synthesis operations. For example, the `Quench` and `PH` actions can, in principle, both be formulated as `Add` actions. Also, a `Partition` action can be expressed as two `Add` actions followed by a `PhaseSeparation` action. In such cases, we want to preserve the intent of the original

experimental procedure and keep the variant closest to the text.

Computationally, actions can be stored as items associating the action type with a set of properties (complying with the available properties for each action type). For practical purposes, we define a bijective conversion to and from a textual representation of the actions. This textual representation is concise and easily understandable. It contains, for each action, all the non-empty properties of that action. With that format, the textual representation of the actions corresponding to the experimental procedure quoted above is shown in Table 2.

Table 2: Action sequence resulting from the example experimental procedure given above.

1	`MakeSolution` with methyl 3-7-amino-2-[(2,4-dichlorophenyl)(hydroxy)methyl]-1H-benzimidazol-1-ylpropanoate (6.00 g, 14.7 mmol) and acetic acid (7.4 mL) and methanol (147 mL);
2	`Add` SLN;
3	`Add` acetaldehyde (4.95 mL, 88.2 mmol) at 0° C;
4	`Wait` 30 min;
5	`Add` sodium acetoxyborohydride (18.7 g, 88.2 mmol);
6	`Wait` 2 h;
7	`Quench` with water;
8	`Concentrate`;
9	`Add` ethyl acetate;
10	`Wash` with aqueous sodium hydroxide (1 M);
11	`Wash` with brine;
12	`DryWithMaterial` sodium sulfate;
13	`Filter` keep filtrate;
14	`Concentrate`;
15	`Purify`;
16	`Yield` title compound (6.30 g, 13.6 mmol, 92%).

Models for action sequence extraction

We studied several models for the automated extraction of action sequences from experimental procedures.

A first possibility is to parse the text for information about operations, compounds, quantities, and other conditions. This can be achieved by inspecting the structure of the

sentences in the experimental procedures to detect the relevant pieces of information with the help of rules. In this work, we look into two such rule-based methods (see Methods for details). These models require meticulous work when formulating extraction rules. Still, they do not always lead to an ideal conversion of experimental procedures into action sequences: it is virtually impossible to define rules covering every possible way to describe a synthesis, while at the same time being robust to noise in the experimental procedures.

To improve the quality of extracted actions, we also look into machine learning for this task. As machine learning models learn from data instead of rules, they are more flexible than rule-based models. This usually results in a greater robustness to noise. In our case, the training data can even be provided by the rule-based models in an initial phase. Concretely, we combine the action sequences generated by rule-based approaches into a pretraining dataset used for the initial training of the machine learning model. We then refine the pretrained model with manually annotated samples of higher quality.

The source of the experimental procedure data and all the above-mentioned approaches for action sequence extraction are detailed in Methods.

We evaluate all the approaches on the test set of the annotation dataset. This set is made up of sentences that are more complex than the average, since the sentences selected for annotation represent cases that the rule-based models struggled with (see Methods).

In Table 3, we show three metrics to compare the models for action sequence extraction. The validity is a measure of syntactical correctness of the textual representation of actions. It is given as the fraction of predictions that can be converted back to actions (as defined in Table 1) without error. The BLEU score[23] is a metric commonly used to evaluate models for machine translation. We adapted its calculation in order not to penalize predictions containing less than four words. The full-sentence accuracy is the fraction of predictions that are identical with the ground truth (i.e., all the actions extracted for a given sentence are correct). In the Supplementary Data 1, the interested reader may find the experimental procedure sentences from the annotation test set with the corresponding actions extracted

by the different models.

Table 3: Metrics for the extraction of actions, evaluated for the approaches introduced in this work on the annotation test set. All values are given in %.

Model	Validity	BLEU score	Full-sentence accuracy
Combined rule-based model	100.0	46.7	25.3
Pretrained translation model	100.0	53.5	25.0
Refined model (no augmentation)	99.7	83.0	61.6
Refined model (augmented)	100.0	84.3	61.9
Refined model (augmented, unique)	100.0	84.6	61.3
Ensemble refined model	100.0	**85.3**	**64.5**

As is to be expected, the combined rule-based model and the deep-learning model pretrained on the rule-based data have a similar performance. Upon manual inspection, it appears that the higher BLEU score of the deep-learning variant can be explained by sentences that the rule-based model classified as `InvalidAction` and that the pretrained model was partially able to predict correctly. Refining the translation model results in a considerable improvement, more than doubling the fraction of sentences that are converted correctly. All three refinement experiments produce models of similar accuracy, none of which is clearly better than the others. By combining the three refined models in an ensemble model, we obtain a significantly improved accuracy. In the following, we only consider this ensemble model for analysis and discussion.

Inspection of the actions extracted by the refined translation model provides interesting insight into its strengths and weaknesses. Out of the 35.5% incorrectly predicted action sequences, the differences are often limited to a single action. In some cases, it is even ambiguous whether the prediction or the annotation is better. In other cases, however, the predictions are clearly incorrect. For instance, the model sometimes incorrectly identifies words or word groups as compounds, such as “LCMS” or “reaction mixture can”. Also, in some cases, the model fails to detect that a sentence corresponds to an invalid action. This problem is difficult to alleviate, since `InvalidAction`s in the annotations often correspond to unusual and infrequent operations or formulations.

To better understand the errors of the model, we take advantage of the ability of the model to make multiple suggestions for translation with a beam search. This is especially interesting for the sentences that the model is least confident about. The five best action sequences suggested by the refined model for all the sentences in the annotation test set can be found in the Supplementary Data 2.

Discussion

The present work demonstrates the ability of a transformer-based model to extract action sequences from experimental procedures written in prose. Training such a model on automatically generated data is already sufficient to achieve a similar accuracy as the rule-based approaches that produced that data. Enhancing the training data with manually annotated samples rapidly shows the advantage of a data-driven approach, since a small set of annotations already leads to a dramatic improvement in accuracy. The ability of the model to learn a complex syntax with a different set of properties for each action type avoids the necessity to design a complex deep-learning model taking into account multiple output types and demonstrates the power of the transformer architecture.

This work represents an important first step towards the automatic execution of arbitrary reactions with robotic systems. Before this is possible, however, it will be necessary to develop methods to infer information missing from experimental procedures. For instance, experimental procedures sometimes do not specify the solvents used for some operations, their quantities, or operation durations.

While the actions defined in this work are able to cover a large majority of experimental procedures, we are aware of some shortcomings of our approach. The choice to only support linear sequences of actions prevents us from addressing cross-references over long distances in the text. The `MakeSolution` and `CollectLayer` partly alleviate this disadvantage by encapsulating the preparation of a solution taking place in a separate flask, and by allowing

for combining multiple solvent fractions generated during work-up, respectively. Also, the current format does not allow operations that depend on the state of the system, in particular formulations indicating *until when* an operation must be performed (*"until the color disappears"*, *"until half the solvent has evaporated"*, and so on).

Another limitation originates in our specific choice of action types (Table 1) and corresponding properties, which does not yet allow for a 100% coverage of the operations in organic chemistry. This limitation can be alleviated by extending the action definitions, which is a process guided mainly by time and experience. In the Supplementary Note 2, we give a few examples of such limitations, as well as suggestions for addressing them.

Improving the automated extraction of action sequences is an ongoing effort, involving refinement of the rules to generate data for pretraining the deep-learning model and annotation of more samples for refining it. A future strategy for the selection of the sentences to annotate will be to choose the ones that the deep-learning model is least confident about.

Although we focused on experimental procedures for organic chemistry extracted from patents, the approach presented in this work is more general. It can be extended to other sources, such as experimental sections from scientific publications, as well as other fields, such as solid-state synthesis.

Methods

Experimental Procedure Data

As a source of experimental procedures, we selected the Pistachio dataset, version 3.0.[24] This dataset contains information related to more than 8.3 M chemical reactions, 6.2 M of which are associated with an experimental procedure.

For each reaction, the Pistachio dataset also contains other information such as patent details and reaction classes, as well as information extracted from the experimental procedures.

Rule-based model 1: Actions derived from Pistachio

For each experimental procedure, the Pistachio dataset contains a list of actions and associated information. Pistachio defines action types similar to the ones in Table 1. The information associated with the Pistachio actions is not operation-specific; the set of properties is common to all action types. It consists, most importantly, of a list of compounds and associated quantities, as well as fields for the temperature, duration, or atmosphere. To convert these actions to our format, we map, where possible, the action types, and post-process the data attached to these actions. For instance, each compound attached to a *`Heat`* action in Pistachio is converted to an `Add` action that is prepended to the `Stir` or `SetTemperature` action.

This approach to the generation of actions from experimental procedures is a good starting point, but limits us to the information detected by Pistachio and reported in the dataset. In particular, some actions relevant to us are not detected, such as all pH-related operations. Also, the Pistachio dataset contains no information about the relationships between compounds in a sentence.

Rule-based model 2: Actions from custom rule-based NLP

We developed a custom rule-based natural language processing (NLP) algorithm for the extraction of operations with associated chemical compounds, quantities, and reaction conditions from experimental procedures.

In a first step, the algorithm processes a text independently of the actions defined in Table 1. It detects operations by searching for verbs corresponding to synthesis operations, defined in a custom list. By analyzing the context of these verbs, the algorithm determines the associated compounds and quantities, as well as additional operation conditions. It also identifies the role of the compounds in the sentence (subject, direct object, etc.), and the relationships between compounds.

In a second step, the operations and associated information are post-processed to map

them to the action types of Table 1. This post-processing is similar to the one of the Pistachio-derived actions detailed above. For this task, information about the relationships between components and their role in the sentence are very useful. For instance, they indicate in what order compounds must be added, independently of what comes first in the sentence (for instance, *"To X is added Y"* or *"Y is added to X"* are equivalent). Also, it allows us to group compounds and convert them to `MakeSolution` actions when they belong together in the text (as in *"A solution of X in Z is added to a solution of Y in Z."*).

This approach to the extraction of actions from text is more flexible for our purposes than deriving the actions from Pistachio, since it can easily be modified or extended. In addition, it allows us to ingest experimental procedures from other sources than the Pistachio dataset.

Combined actions from rule-based models

Starting from a single experimental procedure, both rule-based approaches described above will generate two sequences of actions that may be different. An analysis of the generated actions rapidly uncovers their respective strengths and shortcomings. On the one hand, in our experience, the Pistachio-generated actions are better at extracting `Yield` actions, or at detecting under what atmosphere reactions are conducted. Our custom NLP approach, on the other hand, can cover a broader vocabulary of operations, and supports `MakeSolution` actions.

Combining both sources has the potential to generate actions that are better than each of the approaches taken separately. Formulating an algorithm to accomplish this in a clever way, however, is not straightforward. In this work, the combined dataset appends `Yield` actions from the Pistachio-based extraction to the actions generated by our custom NLP algorithm.

Annotations

To improve on the quality of training data based on the rule-based models, we generated higher-quality action sequences by manually annotating sentences from experimental procedures.

We developed an annotation framework based on the `doccano` annotation tool.[25] Annotators can open the framework in a web browser and navigate through sentences from experimental procedures. The page shows the sentence to annotate and a readable representation of the actions associated with it. An annotator can add new actions, reorder them, or edit them by opening a separate view. Figures 1 and 2 illustrate what a user of the annotation framework sees.

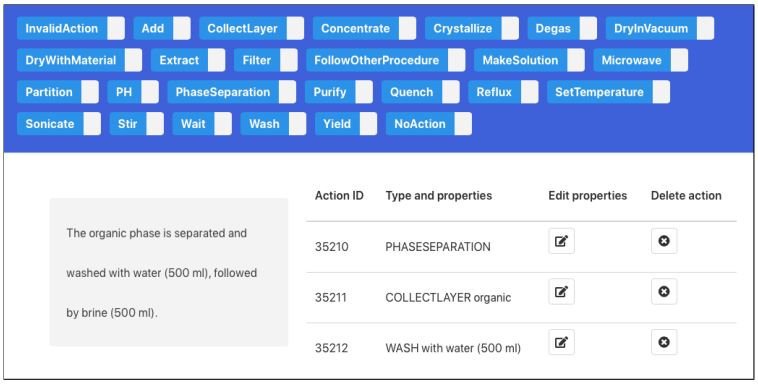


Figure 1: Main view of the annotation framework. The sentence to annotate is displayed on the left-hand side, with the corresponding pre-annotations on the right-hand side. A `Wash` action is missing and can be added by clicking on the corresponding button at the top.

The annotation framework is pre-loaded with samples that are pre-annotated by combining action sequences from both rule-based models as described above. The samples to annotate are not selected at random, but mainly represent categories of experimental procedures for which the rule-based extraction of actions encounters difficulties. One such category relates to verbs whose meaning is highly context-dependent, and another one includes sentences containing "followed by", which the rule-based models usually struggle with.

To ensure consistency among the annotated samples, a single annotator reviewed all the annotations.

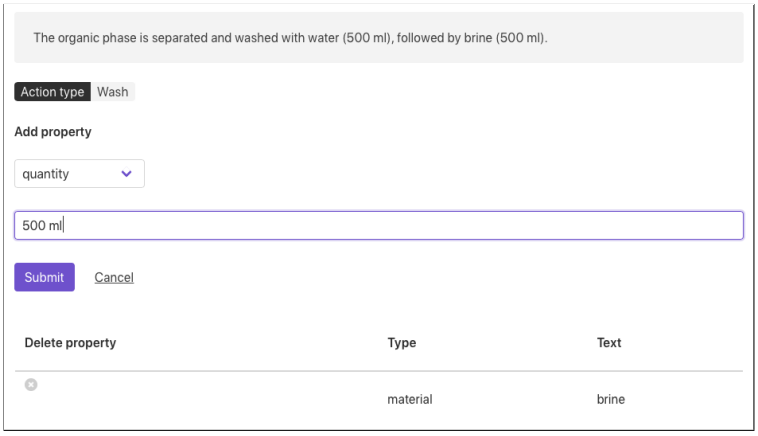


Figure 2: Action edition view of the annotation framework. The properties of the action missing in Fig. 1 are edited by adding the fields associated to the `Wash` action.

Data augmentation on the set of annotated samples increases the number of data points available for refinement in order to minimize overfitting. We augment the data by substituting compound names and quantities, as well as durations and temperatures, in the annotated data. The substitutes are selected at random from lists that we compiled from a subset of the Pistachio dataset.

Machine learning model

We formulate the extraction of action sequences from experimental procedures as a sequence-to-sequence translation, in which experimental procedures are translated to the textual representation of the actions defined in Table 1.

Restricting the output to a textual form is no limitation, since the textual representation of actions can easily be converted back to the action type and associated properties without loss. Furthermore, doing so allows for an easier and more flexible setup than designing a custom architecture for sequential prediction of actions and corresponding properties; this also means that established model architectures for sequence-to-sequence translation can be applied with few modifications.

Experimental procedures usually contain very few cross-sentence dependencies. We therefore translate experimental procedures sentence by sentence. This simplifies the learning task

and limits the requirements on the model architecture. In the few cases where knowledge of the neighboring sentences would be relevant, the missing information can normally be determined from the context as a post-processing step when combining the sentences. As an example, from the sentence *"The solution mixture is filtered and concentrated."*, it is clear that the filtrate is kept rather than the precipitate. For *"The solution mixture is filtered. It is then concentrated."*, this fact can be inferred by noticing that the `Filter` action is followed by a `Concentrate` action, which indicates that the phase to keep after filtration must be the filtrate.

The deep learning model for the conversion of experimental procedures to action sequences relies on a transformer-based translation model. The model is implemented with the `OpenNMT-py` library.[26] The library suggests a set of default parameters for the transformer architecture, which we adopted with a few changes. First, we reduced the model size by decreasing the number of layers from 6 to 4, the size of the hidden states from 512 to 256, and the size of the word vectors from 512 to 256. Second, we changed the values of the parameters `max_generator_batches` to 32, `accum_count` to 4 and `label_smoothing` to 0. Third, we chose the source and target vocabularies to be identical, and accordingly our model shares their embeddings.

The translation model is pretrained on the action sequences generated by combining the NLP and Pistachio approaches as described above. We apply the algorithm to a random subset of 1.0 M experimental procedures and remove sentences resulting in `InvalidAction`s. This provides more than 4.0 M pairs of sentences and corresponding action sequences, which are split into training, validation, and test sets of size 3.2 M, 0.4 M, and 0.4 M, respectively.

The vocabulary is created from the training set with the `SentencePiece` library,[27,28] by setting a target vocabulary size of 16000. The source and target strings are then tokenized using the `SentencePiece` model generated during the generation of the vocabulary.

The model is then pretrained for 350000 steps.

A total of 1764 annotated samples are split into training, validation and test sets of size

1060, 352, and 352, respectively. Based on this data, training is continued for the final model of the pretraining step. Three experiments are run. In the first experiment, the training set containing 984 samples is used as such ("no augmentation"). In the second experiment, the dataset is augmented as described above to produce 20000 samples ("augmented"). In the third experiment, the duplicates contained in the augmented dataset are removed, which results in 13761 samples ("augmented unique"). The validation and test sets are not augmented.

All three models are trained for 30000 steps, with checkpoints saved every 1000 steps. For each experiment we then select the model checkpoint leading to the highest accuracy. We also combine the model checkpoints selected in this fashion into an ensemble model.

Author contributions

The project was conceived and planned by T.L. and A.C.V. and supervised by T.L. F.Z. designed the custom rule-based NLP model. A.C.V. implemented and trained the other models. J.G. set up the annotation framework. All the authors were involved in discussions about the project and annotated the dataset. A.C.V. reviewed all the annotations and wrote the manuscript with input from all authors.

Data availability

The data on which the models for the extraction of action sequences were trained are available from NextMove in the Pistachio dataset.[24] The rule-based and hand-annotated action sequences are available from the authors upon request.

Code availability

The trained models can be freely used online at `https://rxn.res.ibm.com` to extract action sequences from experimental procedures.

References

(1) Peplow, M. Organic synthesis: The robo-chemist. *Nature* **2014**, *512*, 20–22.

(2) Trobe, M.; Burke, M. D. The Molecular Industrial Revolution: Automated Synthesis of Small Molecules. *Angew. Chem. Int. Ed.* **2018**, *57*, 4192–4214.

(3) Steiner, S.; Wolf, J.; Glatzel, S.; Andreou, A.; Granda, J. M.; Keenan, G.; Hinkley, T.; Aragon-Camarasa, G.; Kitson, P. J.; Angelone, D.; Cronin, L. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science* **2019**, *363*, eaav2211.

(4) Coley, C. W. et al. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science* **2019**, *365*, eaax1566.

(5) Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* **2018**, *555*, 604–610.

(6) Coley, C. W.; Green, W. H.; Jensen, K. F. Machine Learning in Computer-Aided Synthesis Planning. *Acc. Chem. Res.* **2018**, *51*, 1281–1289.

(7) Schwaller, P.; Laino, T.; Gaudin, T.; Bolgar, P.; Hunter, C. A.; Bekas, C.; Lee, A. A. Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction. *ACS Cent. Sci.* **2019**, *5*, 1572–1583.

(8) Schwaller, P.; Petraglia, R.; Zullo, V.; Nair, V. H.; Haeuselmann, R. A.; Pisoni, R.; Bekas, C.; Iuliano, A.; Laino, T. Predicting retrosynthetic pathways using a com-

bined linguistic model and hyper-graph exploration strategy. **2019**, Preprint at https://arxiv.org/abs/1910.08036.

(9) Mysore, S.; Kim, E.; Strubell, E.; Liu, A.; Chang, H.-S.; Kompella, S.; Huang, K.; McCallum, A.; Olivetti, E. Automatically Extracting Action Graphs from Materials Science Synthesis Procedures. **2017**, Preprint at https://arxiv.org/abs/1711.06872.

(10) Huo, H.; Rong, Z.; Kononova, O.; Sun, W.; Botari, T.; He, T.; Tshitoyan, V.; Ceder, G. Semi-supervised machine-learning classification of materials synthesis procedures. *npj Comput. Mater.* **2019**, *5*, 62.

(11) Mysore, S.; Jensen, Z.; Kim, E.; Huang, K.; Chang, H.-S.; Strubell, E.; Flanigan, J.; McCallum, A.; Olivetti, E. The Materials Science Procedural Text Corpus: Annotating Materials Synthesis Procedures with Shallow Semantic Structures. **2019**, Preprint at http://arxiv.org/abs/1905.06939.

(12) Hawizy, L.; Jessop, D. M.; Adams, N.; Murray-Rust, P. ChemicalTagger: A tool for semantic text-mining in chemistry. *J. Cheminf.* **2011**, *3*, 17.

(13) Swain, M. C.; Cole, J. M. ChemDataExtractor: A Toolkit for Automated Extraction of Chemical Information from the Scientific Literature. *J. Chem. Inf. Model.* **2016**, *56*, 1894–1904.

(14) Krallinger, M.; Rabal, O.; Lourenço, A.; Oyarzabal, J.; Valencia, A. Information Retrieval and Text Mining Technologies for Chemistry. *Chem. Rev.* **2017**, *117*, 7673–7761.

(15) Kim, E.; Huang, K.; Tomala, A.; Matthews, S.; Strubell, E.; Saunders, A.; McCallum, A.; Olivetti, E. Machine-learned and codified synthesis parameters of oxide materials. *Sci. Data* **2017**, *4*, 170127.

(16) Kim, E.; Huang, K.; Saunders, A.; McCallum, A.; Ceder, G.; Olivetti, E. Materials

Synthesis Insights from Scientific Literature via Text Extraction and Machine Learning. *Chem. Mater.* **2017**, *29*, 9436–9444.

(17) Weston, L.; Tshitoyan, V.; Dagdelen, J.; Kononova, O.; Trewartha, A.; Persson, K. A.; Ceder, G.; Jain, A. Named Entity Recognition and Normalization Applied to Large-Scale Information Extraction from the Materials Science Literature. *J. Chem. Inf. Model.* **2019**, *59*, 3692–3702.

(18) Lowe, D. M.; Sayle, R. A. LeadMine: a grammar and dictionary driven approach to entity recognition. *J. Cheminf.* **2015**, *7*, S5.

(19) Leaman, R.; Wei, C.-H.; Lu, Z. tmChem: a high performance approach for chemical named entity recognition and normalization. *J. Cheminf.* **2015**, *7*, S3.

(20) Korvigo, I.; Holmatov, M.; Zaikovskii, A.; Skoblov, M. Putting hands to rest: efficient deep CNN-RNN architecture for chemical named entity recognition with no hand-crafted rules. *J. Cheminf.* **2018**, *10*, 28.

(21) Reaxys. `https://www.reaxys.com`, (Accessed Dec 13, 2019).

(22) IBM RXN for Chemistry. `https://rxn.res.ibm.com`, (Accessed Dec 20, 2019).

(23) Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, Pennsylvania, USA, 2002; pp 311–318.

(24) Nextmove Software Pistachio. `http://www.nextmovesoftware.com/pistachio.html`, (Accessed Nov 19, 2019).

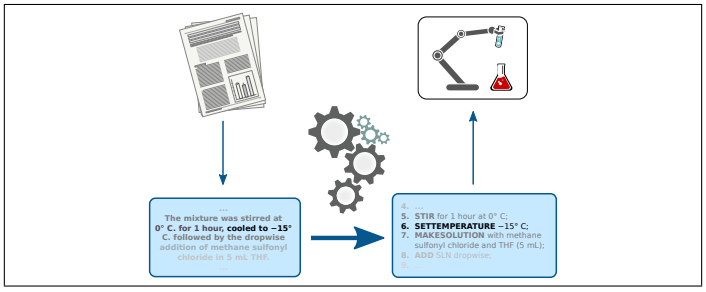
(25) Doccano Annotation Tool. `https://doccano.herokuapp.com`, (Accessed Nov 19, 2019).

(26) OpenNMT-py library, version 0.9.2. `https://github.com/OpenNMT/OpenNMT-py`, (Accessed Nov 19, 2019).

(27) Kudo, T.; Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Brussels, Belgium, 2018; pp 66–71.

(28) SentencePiece library, version 0.1.83. `https://github.com/google/sentencepiece`, (Accessed Nov 19, 2019).

Graphical TOC Entry



Supplementary Information for "Automated extraction of chemical synthesis actions from experimental procedures"

Alain C. Vaucher,* Federico Zipoli, Joppe Geluykens, Vishnu H. Nair, Philippe Schwaller, and Teodoro Laino

IBM Research, Zurich, Switzerland

E-mail: ava@zurich.ibm.com

- Supplementary Data 1: Sentences from the annotation test set with action sequences extracted by the different approaches.
- Supplementary Data 2: Top 5 suggestions predicted by the deep-learning prediction model for sentences from the annotation test set.

Supplementary Note 1: Properties of actions

The Supplementary Table 1 lists the properties associated with the actions presented in Table 1 of the main article. The variable type "chemical" represents the name of a chemical, to which one or several quantities can be attached. For instance, *"0.036 g of sodium azide (0.56 mmol) was added."* results in an `Add` action, where the property "material" contains the name *"sodium azide"* attached with the quantities *"0.036 g"* and *"0.56 mmol"*.

Supplementary Table 1: Action types and corresponding properties.

Action name	Variable name	Variable type
`InvalidAction`	error	string
`Add`	material	chemical
	dropwise	boolean
	temperature	string (optional)
	atmosphere	string (optional)
	duration	string (optional)
`CollectLayer`	layer	string
`Concentrate`	(none)	
`Crystallize`	solvent	chemical
`Degas`	gas	string (optional)
	duration	string (optional)
`DryInVacuum`	duration	string (optional)
	temperature	string (optional)
`DryWithMaterial`	material	string (optional)
`Extract`	solvent	chemical
	repetitions	integer
`Filter`	phase_to_keep	string (optional)
`FollowOtherProcedure`	(none)	
`MakeSolution`	materials	list of chemicals
`Microwave`	duration	string (optional)
	temperature	string (optional)
`Partition`	material_1	chemical
	material_2	chemical
`PH`	material	chemical
	ph	string (optional)
`PhaseSeparation`	(none)	
`Purify`	(none)	
`Quench`	material	chemical
`Reflux`	duration	string (optional)
	dean_stark	boolean
`SetTemperature`	temperature	string
`Sonicate`	duration	string (optional)
	temperature	string (optional)
`Stir`	duration	string (optional)
	temperature	string (optional)
	atmosphere	string (optional)
`Wait`	duration	string
	temperature	string (optional)
`Wash`	material	chemical
	repetitions	integer
`Yield`	material	chemical
`NoAction`	(none)	

Supplementary Note 2: Discussion of the current set of actions

Defining the action types and corresponding properties is a compromise. On the one hand, we want to keep the action vocabulary concise in order to limit the complexity and make the models easy to train. On the other hand, we would like to cover as many operations as possible.

Currently, sentences containing unsupported operations are either converted to an `InvalidAction`, or the incompatible piece of information is ignored. The most common examples are the following:

- Currently, `Add` considers a main reactor containing a reaction mixture to which other substances are added. In some experimental procedures, however, the reaction mixture is poured into another solution instead. While in many cases the directionality of the addition operation does not matter, sometimes this differentiation may be important, such as when the reaction mixture is to be added to something else dropwise. To support this, a property specifying the directionality of the addition could be added to `Add`.

- `SetTemperature` indicates a punctual change in temperature, while in some procedures the change in temperature happens over a specified duration. Such an example would be *"The reaction mixture is cooled to room temperature over 5 hours."* Currently, such a sentence would be converted to a `Stir` action, meaning a punctual change to room temperature followed by stirring for five hours. In most cases, this distinction is irrelevant. Still, to make it explicit, a property for the duration could be added to `SetTemperature`.

- The `Purify` action type currently has no properties. It could be extended to include information about how the purification should proceed.

- Some experimental procedures specify an atmosphere for reflux operations, or for operations converted to `SetTemperature`. Both `Reflux` and `SetTemperature` currently

have no support for that, but a corresponding property could be added if desired.

- Experimental procedures sometimes specify a temperature or the speed of addition for quenching or pH operations. Support for these can be achieved by considering "dropwise" and "temperature" properties for `PH` and `Quench`, as for the `Add` action.

- Currently, the only available action type for drying solids is `DryInVacuum`, which implicitly assumes that drying happens under reduced pressure. Nevertheless, in some experimental procedures, solids are dried with air or inert gases, which is not supported by the current set of actions. To support both, `DryInVacuum` could be converted to an action `DrySolid` including, as parameters, whether drying happens under reduced pressure or under a specific atmosphere.

- Some operations occurring in experimental procedures are not covered by the current set of actions. This could be remedied by adding corresponding actions types and properties. Examples of such operations are trituration and hydrogenation.

Other files

supplementary_data_1_actions_for_test_set.txt (271.10 KiB)	view on ChemRxiv • download file
supplementary_data_2_top_5_sequences.txt (261.26 KiB)	view on ChemRxiv • download file