Received: 26 February 2023 | Revised: 29 August 2023 | Accepted: 8 September 2023

DOI: 10.1002/aic.18259

AIChE JOURNAL

RESEARCH ARTICLE

Process Systems Engineering

Toward automatic generation of control structures for process flow diagrams with large language models

Edwin Hirtreiter | Lukas Schulze Balhorn | Artur M. Schweidtmann

Department of Chemical Engineering, Delft University of Technology, Delft, The Netherlands

Correspondence
Artur M. Schweidtmann, Department of Chemical Engineering, Delft University of Technology, Van der Maasweg 9, Delft 2629 HZ, The Netherlands.
Email: a.schweidtmann@tudelft.nl

Funding information
NWO, Grant/Award Number: 203.001.107

Abstract

Developing Piping and Instrumentation Diagrams (P&IDs) is a crucial step during process development. We propose a data-driven method for the prediction of control structures. Our methodology is inspired by end-to-end transformer-based human language translation models. We cast the control structure prediction as a translation task where Process Flow Diagrams (PFDs) without control structures are translated to PFDs with control structures. We represent the topology of PFDs as strings using the SFILES 2.0 notation. We pretrain our model using generated PFDs to learn the grammatical structure. Thereafter, the model is fine-tuned leveraging transfer learning on real PFDs. The model achieved a top-5 accuracy of 74.8% on 10,000 generated PFDs and 89.2% on 100,000 generated PFDs. These promising results show great potential for AI-assisted process engineering. The tests on a dataset of 312 real PFDs indicate the need for a larger PFD dataset for industry applications and hybrid artificial intelligence solutions.

KEYWORDS
artificial intelligence, control structure, deep learning, machine Learning, piping and instrumentation diagram, process flow diagram, transformer language model

1 | INTRODUCTION

Piping and Instrumentation Diagrams (P&IDs) are important engineering documents of chemical plants depicting the arrangement of process equipment, valves, piping, control structure, and instrumentation.[1] In contrast, Process Flow Diagrams (PFDs) focus on major equipment parts and material streams. While PFDs are typically used during the early stage conceptual design phase, P&IDs are developed in the basic design and detailed engineering phases. They are essentially the central document in every industrial chemical plant for storing, revising, and exchanging information.[2] The applications of P&IDs range from engineering and design, to hazard and operability studies, construction, operation, maintenance, and decommission.[2]

The development of P&IDs from PFDs is a tedious and time-consuming task that offers great potential to reduce costs and speed-up the development process.[3] Commonly, process engineers manually develop P&IDs adopting and modifying schemes from prior projects, design rules, and their experience utilizing Computer-Aided Design (CAD) software. However, this traditional development can be laborious because finding, manually adjusting, and transferring suitable technical solutions from old projects can be tedious and error-prone. Time constraints can lead to the adoption of nonoptimal solutions from previous projects and possible alternatives not being considered.[3] Unleashing the potential of computer algorithms assisting engineers in process engineering may help to reduce development times, reduce costs, increase safety, and avoid errors.

Researchers have been working on the automation of process development since the 90s. To assist the engineering process during the creation of P&IDs, multiple rule-based systems have been developed.[3-5] Modularization approaches of chemical plants commonly provide the underlying framework of rule-based systems and aim to accelerate process development.[6-8] The method proposed by Blitz et al.[4] asks a user to define certain inputs, such as material properties and process-specific requirements. Then, a P&ID is generated based on the user input and the underlying knowledge-based approach, which is implemented as a decision tree. Similarly, Uzuner et al.[3] and Obst et al.[5] also utilize a knowledge-based method, which is represented as a hierarchical decision tree. Uzuner et al.[3] first divide the chemical process into modules to reduce the complexity of the design problem. Secondly, design questions and options are used to guide the user to obtain a P&ID of the desired module. While the previous works demonstrate the potential of computer-assisted P&ID development, they have not yet been broadly adopted by industry.

One step toward the development of P&IDs is the synthesis of an appropriate control structure. In decentralized control, a controller adjusts manipulated variables based on observations of measured variables in order to follow a set point of a controlled variable or optimize another operating objective.[9] Typically, the development of a plant-wide control scheme starts by analysis of the degrees of freedom (i.e., the number of controllable variables).[10,11] In parallel, control tasks are defined that relate to decentralized operational targets, such as set-point tracking or disturbance rejection of product qualities or flow rates, and overarching economic and ecological objectives, such as maximum product output or minimum energy consumption.[9-11] To assist process engineers in developing (decentralized) control structures that specify all controllable variables and achieve the process objectives, several established methods exist including dynamic simulations[12] and relative gain array methods.[13] In addition, heuristic design procedures[10] and knowledge-based expert systems[14,15] have been proposed in the 90s. However, many expert systems in chemical engineering have not led to the expected major advances.[16] In particular, rule-based systems are often difficult to set up, maintain, and extend.[3,16]

Recent research and development in deep learning-based Artificial Intelligence (AI) applications promise improvements over expert systems revealing an outstanding performance in numerous disciplines, as highlighted by the following examples. In particular, Natural Language Processing (NLP), a subfield of AI focusing on natural language, with its powerful models (e.g. GPT-3,[17] T5[18]) showed breakthrough performance in many natural language tasks outperforming systems that previously used handcrafted rules.[17-20] We already see great speedups of AI-assisted workflows in many domains (e.g., deepL for translations or GitHub Copilot software development). Similarly, there are a large number of domains that explore the potential of ChatGPT, although there are no guarantees of correct predictions. Also, deep learning has outperformed rule-based approaches in organic chemistry. For example, transformer-based language models can accurately predict reaction outcomes based on string representations of reactions using the Simplified Molecular-Input Line-Entry System (SMILES) notation.[21-25]

In the context of process engineering, there exist a few very recent and promising methods, which learn patterns from existing PFDs and P&IDs.[26-29] Zhang et al.[26] and Zheng et al.[27] use the Simplified Flowsheet-Input Line-Entry System (SFILES)[30] notation to describe the flowsheet topologies as strings in conjunction with sequence alignment algorithms to identify design heuristics in process diagrams. Oeing et al.[28] propose an AI-assisted method to predict the subsequent equipment using a Recurrent Neural Network (RNN). Similarly, we proposed a methodology for auto-completion of flowsheets based on transformer language models.[29] To enable the use of NLP models, we utilized the SFILES 2.0[31] notation. While previous methods focus on the completion of incomplete process diagrams, there is no method available that enables the generation of PFDs with decentralized control structures directly from basic PFDs without control structures.

We propose a novel methodology to generate PFDs with decentralized control structures from PFDs without control structures as a first step toward the automatic generation of P&IDs. Notably, we provide a conceptual contribution as well as a proof-of-concept which demonstrates that predicting decentralized control structures on synthetic data is feasible. The underlying idea of our approach is to cast the control structure prediction as a translation task, the source language being the PFD without control structure and the target being the PFD with control structure. To leverage the potential of state-of-the-art sequence-to-sequence translation models, based on the transformer architecture,[19] we utilize the text-based SFILES 2.0 notation[31] to represent the topological information of PFDs.

The remainder of this article is structured as follows: Section 2 describes the fundamentals of the applied natural language model and summarizes the concept of the SFILES 2.0 string representation of chemical processes typologies. Thereafter, in Section 3 we describe the data acquisition. In Section 4 we introduce the transformer model adapted for predicting the control structure. Afterward, the results are discussed in Section 5 and demonstrated with an illustrative example in Section 5.3.

2 | BACKGROUND

This section summarizes the fundamentals of sequence-to-sequence models for the translation of natural language (Section 2.1). In Section 2.2, we highlight the transformer architecture as the state-of-the-art deep learning architecture for translation. Thereafter, the concept of the SFILES 2.0 notation, which enables a text-based representation of PFDs including control structures, is outlined in Section 2.3. The underlying idea for using SFILES 2.0 in combination with NLP methods was originally proposed by Vogel et al.[29] Vogel et al.[29] utilize a generative transformer model consisting of a decoder-only model structure. The decoder is fed with an incomplete PFD, represented as SFILES 2.0, and generates step-by-step the missing parts of the flowsheet. While the previous work proposed a flowsheet autocompletion methodology,[29] we developed a concept for the prediction of control structures in PFDs. For this purpose, we develop a sequence-to-sequence model with an encoder-decoder structure. That is, we map

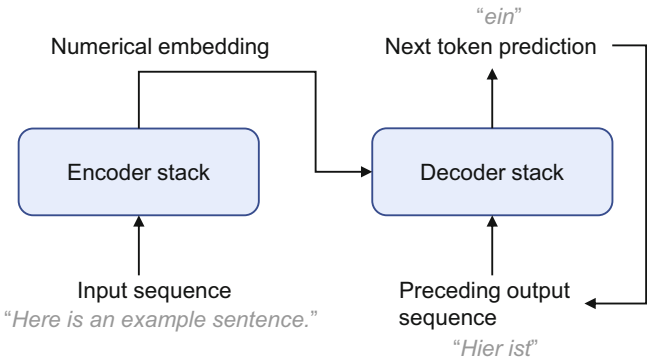


FIGURE 1 Encoder-decoder structure of sequence-to-sequence models.

the PFD without control structure directly to the PFD with control structure.

2.1 | Sequence-to-sequence models

Sequence-to-sequence models are machine learning models that map an input sequence to an output sequence. They are utilized in numerous NLP tasks, for example, in translation,[32] text summarization,[33] speech recognition,[34] and image captioning.[35]

Typically, a sequence-to-sequence model comprises an encoding and decoding stack as depicted in Figure 1. During encoding, a numerical embedding of the input sequence is determined, which is subsequently used by the decoder stack to generate the output sequence in an autoregressive way. The decoder iteratively processes the preceding output sequence together with the numerical embedding of the encoder to predict the next token (e.g., a word). The iterative decoding is stopped as the decoder predicts the end of the sequence as the next token.

During decoding, the decoder stack determines the probabilities for each token in its vocabulary, whereupon the next token of the output sequence is identified utilizing a decoding strategy (e.g., using the greedy or beam search decoding strategy). Greedy search selects the next token of a sequence based on the highest predicted probability at the decoding step. The greedy strategy is computationally cheap. However, it does not ensure a sequence with a maximal overall probability because sequences with a high probability can also contain some tokens with a low probability. To mitigate this issue the beam search algorithm was introduced in sequence-to-sequence models (e.g., References 36-38). Beam search selects and memorizes the N-best tokens at every decoding step creating a tree of possible output sequences. Every selected token is added separately to the preceding output sequence, and thus the decoder is prompted in total N-times to predict the output probabilities of the next tokens. Therefrom the decoder selects N tokens with the highest probabilities for the next decoding step and discards branches with a lower probability. In the end either the sequence with the overall highest probability is selected or the N-best sequences are returned to a user for selection. Choosing an appropriate beam size N is a trade-off between generating sequences with high probabilities and computational cost.

Training of sequence-to-sequence models is typically performed using the cross-entropy loss of the predicted output probabilities of the next tokens and the ground truth.[18] With the aid of the computed cross-entropy loss the parameters of the model are adjusted to improve the model performance. Teacher forcing[39] is commonly applied to correct the model at each decoding step, which forces the model to generate the ground truth corresponding to a given input sequence.[18]

2.2 | Transformer architecture

Originally, the underlying model architectures of sequence-to-sequence models comprised variations of RNNs.[36] To avoid vanishing or exploding gradients, long short-term memory[40] and gated recurrent neural nets[41] were also introduced. Recently, the transformer architecture,[19] which is based on the sequence-to-sequence model structure, revolutionized the field of NLP demonstrating breakthrough performances on numerous tasks.[17-19]

The transformer architecture[19] is based on the auto-regressive encoder-decoder model structure and was originally proposed to perform translation tasks. The transformer model relies entirely on attention mechanisms dispensing any recurrence or convolutions. Eliminating recurrence and using attention significantly reduces the number of sequential computations and enables fast parallel processing and model training.[19]

Attention, being an important core component of the transformer architecture, enables the model to efficiently capture the meaning of a token depending on the context present in the sequence. During model training, the weights of query, key, and value matrices are adjusted to learn the bidirectional context of words in a sequence. These matrices are used to compute a query **q**, key **k**, and value **v** vector from the input embedding. The resulting vectors are packed to query Q, key K, and value V matrices to efficiently compute the scaled dot-product attention. The implementation of the scaled dot-product attention in the transformer architecture includes a scaling factor d_k corresponding to the layer size:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1)$$

To allow the model to learn different representations of a single word, multihead attention is introduced. For this purpose, the queries, keys, and values are linearly projected to different dimensions and processed in parallel by multiple attention heads, which are thereafter concatenated. Thus, multihead attention enables the model model to learn multiple relations of a word within a sentence.

The attention mechanisms are not able to cover any positional information due to the absence of recurrence or convolutions in the transformer architecture. Therefore, a positional encoding, utilizing sine and cosine functions, is added to the input and output embedding to provide information about the position in the sequence.

The structure of the original transformer architecture comprises an encoder and a decoder stack each containing six identical layers.

The encoder layers consist of a multihead attention sublayer followed by a position-wise fully connected feed-forward network. Each sublayer is succeeded by layer normalization and the addition of a residual connection, which prevent "losing" information from the previous layer and facilitate the gradient flow. Residual connections are bypass connections around layers, which transfer information from previous layers. Without those layers, information from preceding layers may otherwise be diminished by operations in the subsequent layers. The structure of the decoder stack is similar to the encoder with the difference that the decoder contains two attention sublayers. The first attention sublayer is masked, which limits the decoder to attend only to already generated tokens and prevents the decoder "from glancing into the future." During training, the model is fed simultaneously with input and target data. Therefore, the masking algorithm prevents the model from simply taking the next token from the target sequence, but learns to predict the next token based on the information in the input sequence and the tokens already generated. The second attention sublayer, the encoder-decoder attention layer, performs multihead attention combining the numerical embedding of the last encoder layer and the results of the preceding self-attention layer.

2.3 | Graph- and text-based representation of process diagrams

This section briefly summarizes the graph- and text-based representation of process diagrams as SFILES 2.0.[29,31] Process diagrams (e.g., PFDs or P&IDs) of chemical plants can be represented as directed graphs.[31,42] Unit operations and control units can be illustrated as nodes in the graph, while material streams and signals are directed edges connecting the nodes. Figure 2 shows an illustrative example process containing a reactor with level control and a recycle loop with flow control. This process diagram can be converted to its corresponding graph representation as depicted in Figure 3. Notably, the two-stream heat exchanger (hex-1) is split into two nodes to distinguish the two separate material flows, which do not mix inside the heat exchanger. The control units are stored as nodes like unit operations.

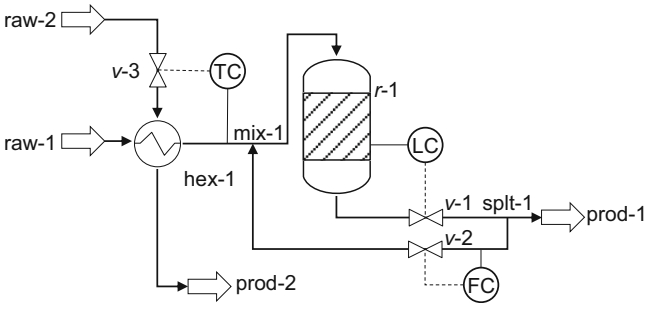


FIGURE 2 Exemplary chemical process diagram with branching, recycle stream, control units and different mass trains.

SFILES 2.0[31] is a text-based representation of process topologies, extending the original SFILES notation as proposed by d'Anterroches.[30] The SFILES notation is inspired by the SMILES notation, which is used for representing molecules as strings.[21,22] With SFILES 2.0 we can efficiently store the topological information of a process graph (e.g., Figure 3) as text, which enables us the application of advanced data processing methods, such as NLP models (Section 2.2). Converting the graph in Figure 3 to the SFILES 2.0 notation with our publicly available Github repository (https://doi.org/10.5281/zenodo.6901932)[43] results in the following string:

```
(raw)(hex){1}(C){TC}_1(mix)<1(r)[(C){LC}_2](v)
<_2(splt)[(prod)](C){FC}_3(v)1
<_3n|(raw)(v)<_1(hex){1}(prod).
```

The SFILES 2.0 notation is read from left to right with two consecutive unit operations respectively control units in parentheses implying a material flow in between. Branching in the process, for example after the stream splitter (splt), is represented by putting the individual branches in brackets (here prod), but omitting the brackets for the stream noted last at the branching point (here the recycle flow over C/FC). Material recycles are included in the SFILES 2.0 notation using a number # for the starting point (v) and <# for the corresponding target (mix). The heat exchanger is noted twice in the string with a number in braces, indicating that it is the same heat exchanger but two streams enter and leave the equipment. Independent material streams, such as the utility stream flowing through the heat exchanger compartment hex-1/2, are appended to the SFILES 2.0 string stream

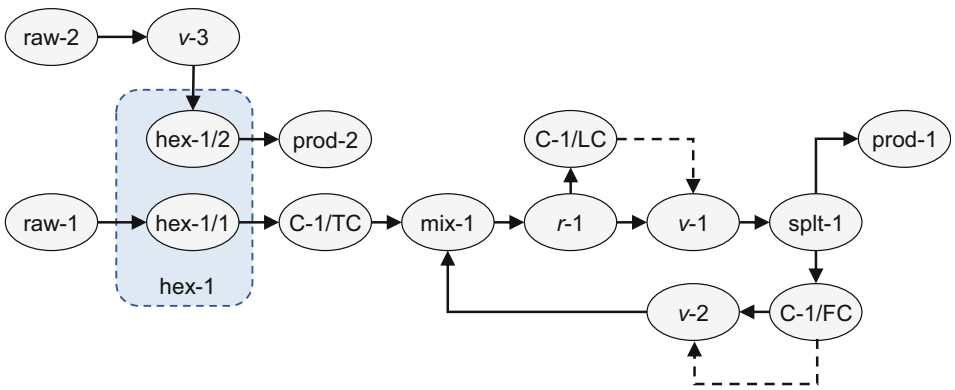


FIGURE 3 Graph representation of the process diagram in Figure 2.

separated with n|. Control units are inserted in the same way as unit operations with subsequent braces indicating the letter code of the instrument. Signal connections are implemented similarly to material recycles but include an underscore (_#, <_#). Currently, the SFILES 2.0 notation is capable of representing flowsheet topologies with their corresponding control structures. The integration of detailed information on material flows, present components, process dynamics, equipment sizing, special type unit operations, piping information, or operating points is at the moment not available in the SFILES 2.0 notation, but will be subject to further extensions.

3 | DATA

We use generated data and a dataset of real PFDs with control structures for model training and evaluation. Section 3.1 describes the generation algorithm of PFDs with decentralized control structures, which are utilized for pre-training the model. Subsequently, Section 3.2 summarizes the pre-processing of real PFDs with control structures derived from publicly available sources used for model fine-tuning. It should be highlighted that the generated PFDs and the mined PFDs only contain control structures. These diagrams are far less detailed than P&IDs available in the industry containing additional information, such as pipe classes, valves, or instrumentation.

3.1 | Generated data for pretraining

Typically, NLP models are trained on huge corpora of text that are publicly available on the internet. For example, Common Crawl (https://commoncrawl.org) is a publicly available database that extracts around 20 TB of text from the web every month.[18] Filtered and cleaned, data from Common Crawl was used as C4 (Colossal Clean Crawled Corpus) dataset with about 750 GB to pre-train the roughly 220 million parameters of the T5-base model.[18] Commonly, transfer learning techniques are employed to reduce this massive data demand for new applications.[18]

Although the SFILES 2.0 notation with its limited, small vocabulary is less complex than natural language, a reasonable amount of pretraining data is necessary to train the randomly initialized weights of the transformer model. Due to a completely different vocabulary of the SFILES 2.0 compared to natural language, we cannot leverage transfer learning on human language models. Also, there is no database of PFDs and P&IDs publicly available.

Since data availability is a major limitation of our method, we generate a large set of PFDs with decentralized control structures by extending the previously proposed approach by Vogel et al.[29] for generating several thousand PFDs in a time-efficient way. This generated dataset is used for pretraining in a transfer learning approach (see Section 5.1 for more details). The main goal of this data generation step is to obtain a dataset, which is possibly similar to real data. The generated data allows the model to learn the grammatical structure of SFILES 2.0 and to demonstrate the capabilities of the model

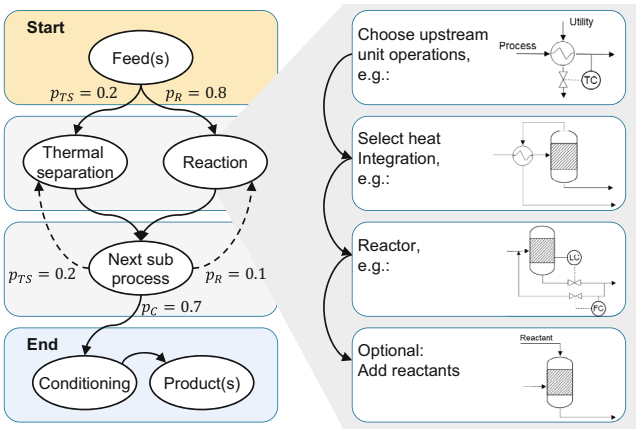


FIGURE 4 Generation scheme for a process flow diagram with control structure.

predicting decentralized control structures. For this purpose, we implement patterns including decentralized control structures of subprocesses, such as thermal separation or reaction, in which a chemical process is typically divided. These patterns are thereafter added together to create the PFDs with decentralized control structures of a chemical process consisting of multiple subprocesses. The construction of the PFD dataset follows a first-order Markov chain-like sampling process with fixed probabilities, that is, the selection of the next subprocess only depends on the current state. The probabilities, shown in Figure 4, for the transition between the subprocesses are selected based on our experience to generate realistic process flowsheets sufficient for pretraining the model. The utilization of fixed probabilities result in a general structure of the process flowsheet consisting of feed treatment, followed by reaction, thermal separation and final conditioning. Compared to Vogel et al.,[29] we add control structures based on several basic design heuristics (inspired by References 1, 44, and 45) for every generated subprocess. The utilized decentralized control patterns are included in Figures S1–S8. As illustrated in Figure 4, we initialize up to three feed streams, which may be preprocessed by inserting heat exchangers, pumps, compressors, or mixing units. Thereafter, a Markov transition selects either thermal separation or reaction as the next subprocess. Exemplary illustrated is the generation pattern for the reaction pattern: Firstly, upstream unit operations comprising heat exchangers, pumps, and compressors are selected. Thereafter, present heat exchangers may be pre-selected for heat integration utilizing a reactor outlet stream. In the next step, one of six stored reactor patterns with an optional material recycle stream is selected. Optionally, a second or third reactant is fed to the reactor in the final step completing the reaction pattern. In general, the patterns have several outlet streams transitioning to the "Next sub-process" state, which lead to multiple Markov transitions to subsequent subprocesses. Branches are either terminated after reaching the conditioning step or if the generation algorithm detects a node number exceeding 65, which prevents the generation of very large flowsheet graphs. Duplicates and process diagrams exceeding a node number of 100 are deleted. We selected a maximum node number of 100 such

TABLE 1 Dataset properties and training (tr), validation (val), test (te) splits used for the experiments.

	Generated data	Real data
samples_{tr}	100/1000/10,000/100,000	250
$\text{samples}_{\text{val}}$	1000	31
samples_{te}	1000	31
$\overline{n_{\text{nodes}}}$	52	37
$\sigma(n_{\text{nodes}})$	20	28
Vocabulary size	113	390

that the standard deviation and average node number of the generated dataset are in the same order of magnitude as our real PFD dataset (cf. Table 1). During the sampling process with fixed probabilities, it can happen that incorrect combinations of decentralized control structures occur, which can ultimately have a negative impact on the accuracy of the trained model.

The resulting process graphs with control structure are automatically converted to SFILES 2.0 using our graph to SFILES 2.0 algorithm.[43] In a subsequent step, the SFILES 2.0 with control structure are converted to SFILES 2.0 without control structure by removing all control instruments (abbrev. C) with their corresponding letter code in braces and signal connections identifiable by an underscore before the number #. Finally, the generated pretraining dataset consists of process diagrams without control structure (input data) and process diagrams with control structure (output data). Table 1summarizes the number of training/validation/test samples for model pretraining and key properties of the dataset. Besides the number of samples, Table 1 shows the average number of nodes $\overline{n_{\text{nodes}}}$, the SD of the number of nodes $\sigma(n_{\text{nodes}})$, and the vocabulary size. The inclusion of a more diverse set of control patterns and, in particular, letter codes to increase the vocabulary of the generated dataset could improve the positive effect of the model pre-training procedure. The complete generated dataset including the SFILES 2.0 PFDs with decentralized control structures and the corresponding PFDs without control structures are published.[46]

3.2 | Real data for fine-tuning

We collected 312 PFD images including control structures from publicly available sources including the Google and Bing image search engines* and extracted process diagrams from scientific literature using data mining.[47] These process flowsheets originate from various industry and academic domains, such as gas and oil plants, experimental setups, or batch operations. After the manual selection of process diagrams containing control structure, automatic object detection, and path exploration is performed using our flowsheet digitization algorithm.[48] Correcting faulty nodes and edges, adding the letter code of control units, and adding the connectivity of the unit operations and control structures is performed using LabelGraph, which is our custom extension to LabelImg.[49] In addition, this manual correction step in LabelGraph ensures the trustworthiness of the mined and digitized process flowsheets. In this step, process flowsheets with obvious faults, as well as duplicates, are deleted, but due to time limitations, a detailed check could not be performed. The resulting process graphs are converted to SFILES 2.0 using our code.[43] Then, all control structures are removed to build a dataset consisting of SFILES 2.0 without control structure as our input data and SFILES 2.0 with control structures as our output data. Key statistics of the dataset are denoted in Table 1. The table shows that the standard deviation of the number of nodes in the real data (28) is significantly higher than in the generated data (20) while the average number of nodes is smaller in the real data. This indicates that the sizes of the process diagrams vary more strongly in the real dataset. The table also highlights a significantly higher vocabulary size of the real dataset (390) compared to the generated dataset (113). The reason for this is mainly a diversity of additional, new letter codes, but also other new unit operations, which are not present in the generated data. To conclude, identifying good measures for the comparison of process diagrams (here, generated data and real data) is difficult. As a first step, we used the average node number, the SD of the number of nodes and the vocabulary size. In the future, an investigation of different measures, such as graph similarity, might be helpful for comparing two different process flowsheets.

3.3 | Data augmentation

Data augmentation methods are commonly applied to datasets to increase their size without the effort of manual labeling and to improve the robustness of machine learning models. In computer vision, images are rotated, cropped, or distorted to have multiple instances of the original image, which are from the computer's point of view completely different. In the field of NLP, data augmentation is more difficult since the meaning of the sentence has to be preserved. Techniques in NLP for data augmentation include for example, synonym replacement, back-translation, random insertion, deletion, or swapping of words.[50-52]

To augment the process diagram datasets, we modify the branching decision in the SFILES 2.0 generation algorithm to create different SFILES 2.0 strings representing the same process diagram.[53] This procedure is motivated by significant performance advances when using augmented SMILES in neural networks.[24,54,55] When generating augmented (noncanonical) SFILES 2.0, the branching decision is made randomly, whereas in the case of the determination of canonical SFILES 2.0 the branching decision is predetermined by assigning every node of the graph to a unique rank.[31] The resulting augmented SFILES 2.0 is grammatically correct and contains identical information as the canonical SFILES 2.0 and thus describes the same process

*Search keywords: "Piping and Instrumentation Diagram," "Rohrleitungs und Instrumentenfließbild," "P&ID," "R&I," "R+I."

flowsheet. During augmentation, only the uniqueness of the SFILES 2.0 representation is lost. For the augmented model training runs we roughly doubled the training data by generating a second SFILES 2.0 for every PFD in the input dataset. As an example, the PFD corresponding to Figure 2 is represented by the following canonical SFILES 2.0

```
(raw)(hex){1}(mix)<1(r)(v)(splt)[(prod)](v)
1n|(raw)(v)(hex){1}(prod),
```

which can be augmented to

```
(raw)(hex){1}(mix)<1(r)(v)(splt)[(v)1](prod)n
|(raw)(v)(hex){1}(prod).
```

To review more examples of augmented SFILES 2.0, we published the augmented dataset together with the non-augmented generated dataset.[46]

4 | CONTROL STRUCTURE PREDICTION MODEL

In the following section, we provide an overview of the general procedure to predict the control structure of PFDs utilizing a sequence-to-sequence transformer model. In Section 4.2, we describe the tokenizer that enables the model to process the SFILES 2.0 strings. Thereafter, key parameters of the utilized transformer architecture are briefly summarized in Section 4.3.

4.1 | Overview

Figure 5A presents an overview of the control structure prediction model, which is described in the following. Firstly, the PFD, which is subject to the development of a control structure, is converted to the corresponding SFILES 2.0 string as described in Section 2.3 (Step 1). Then, the SFILES 2.0 string is split into chunks of text using the SFILES 2.0 tokenizer as explained in Section 4.2 (Step 2). After converting the tokenized string to an input embedding and adding a positional encoding, the encoder stack computes a numerical embedding of the input string (Step 3). In Step 4, the decoder stack is initially prompted with a start-of-sequence token. In combination with the numerical embedding of the input sequence produced by the encoder, the decoder stack predicts the next token of the output sequence. The predicted token is then added to the preceding tokens of the output sequence and the decoder is again prompted to predict the next token (Step 5). This auto-regressive prediction of tokens is continued until an end-of-sequence token terminates the prediction process (Step 6). Lastly, the resulting SFILES 2.0 string is converted to its corresponding graph representation, the PFD with control structure. Eventually, this procedure could be implemented in CAD software packages to automatically generate the control structure of a drawn PFD as depicted in Figure 5B.

4.2 | Tokenization

Tokenizers are generally used in NLP to split text sequences into pieces that can be processed by the language model. The aim is to compress as many words of a language as possible into a fixed vocabulary while preserving the meaning of the words. Using the vocabulary, tokenizers convert the input sequence into a numerical vector, which can be processed by the NLP model. Different tokenization algorithms have been developed according to different languages and intended use cases. The most commonly used tokenization algorithms comprise word- and subword-based tokenizers, which split the text into words or parts of words and automatically build their vocabulary. Examples of popular subword-based tokenizers include byte-pair encoding[56] and SentencePiece.[57]

We perform the tokenization of the SFILES 2.0 using a custom tokenizer to preserve the inherent structure of the SFILES 2.0 notation, which is significantly different from natural language. Inspired by the SMILES tokenizer of Schwaller et al.,[23] we propose a SFILES 2.0 tokenizer, which, for instance, identifies unit operations (e.g., (hex)), stream tags (e.g., tout), letter codes (e.g., LC), material recycle (e.g., <#, #) and signal connections (e.g., <_#, _#). The SFILES 2.0 string is split into pieces using the following regular expression, which is used to search and match certain patterns in the SFILES 2.0:

```
(\(.+?\)|\{.+?\}|[<%_]
+\d+|\]|\[|\<\&\||(?<!)&\|n\
||(?<!&)(?<!n)\||&(?!\|)|\d).
```

For example, tokenizing the following SFILES 2.0 string

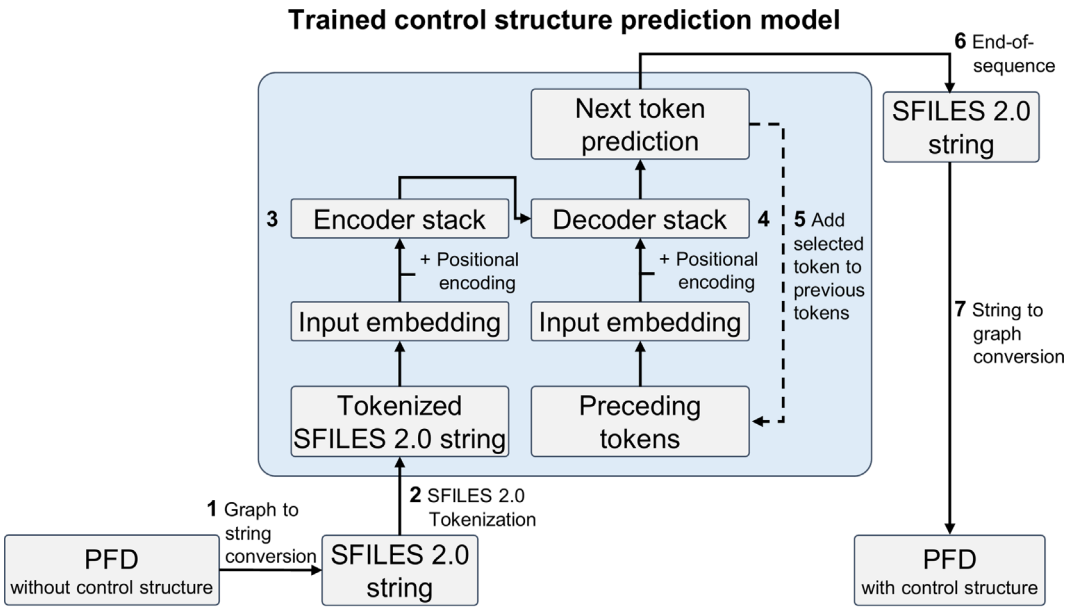
```
(raw)(hex){1}(C){TC}_1(mix)<1(r)[(C){LC}_2](v)
<_2(splt)[(prod)](C){FC}_3(v)1<_3n|(raw)(v)
<_1(hex){1}(prod)
```

results in

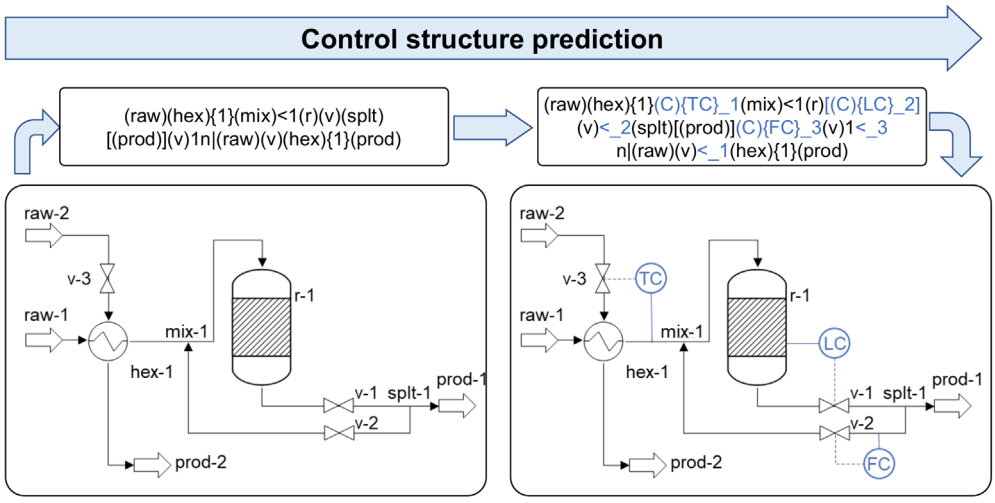
```
(raw), (hex), {1}, (C), {TC},_1, (mix), <1, (r),
[, (C), {LC},_2, ], (v), <_2, (splt), [, (prod), ],
(C), {FC}, _3, (v), 1, <_3, n|, (raw), (v),
<_1, (hex), {1}, (prod).
```

4.3 | T5-Model for control structure prediction

There exist multiple, publicly available sequence-to-sequence models, such as OpenNMT[58] or the T5 model.[18] We use the T5 transformer model,[18] a state-of-the-art model easily accessible through Hugging Face (https://huggingface.co), casting the control structure prediction as a translation task. Therefore, the employed model is a sequence-to-sequence model with an encoder-decoder structure as explained in Section 2.1. The T5 model is in large parts equivalent to the original transformer architecture proposed by Vaswani et al.[19] Modifications include removing the layer bias norm, placing layer normalization



(A)



(B)

FIGURE 5 Overview of the control structure prediction with the transformer model. (A) Conversion of the process flow diagram (PFD) to SFILES 2.0 (1). Processing of the input SFILES 2.0 with transformer model to predict the control structure (2–5). Conversion of the output SFILES 2.0 to the PFD including the corresponding control structure (6-7). (B) Example control structure prediction adapted from Reference 29.

outside the residual connections, and applying a different positional encoding.[18] Since the SFILES 2.0 vocabulary is limited to a few hundred entries, we utilize the T5-small version with originally about 60 million parameters. The T5-small model has an embedding size of 512, utilizes an eight-headed attention mechanism, and consists of six encoder and decoder layers each. Preliminary tests on a generated SFILES 2.0 dataset with around 10,000 samples indicate, that an even smaller architecture may be sufficient and advantageous. For this reason, we further decrease the model size of the T5-small model by reducing the embedding size to 128 and the number of encoder and decoder layers each to two. In summary, our model comprises roughly 7.9 million trainable parameters. Compared to other state-of-the-art language models, our model size is significantly smaller, but our vocabulary sizes are clearly smaller, too. We performed a hyperparameter optimization including the model size and several other model and training parameters based on a grid-search hyperparameter tuning run (cf. Table S1). The results of the hyperparameter tuning run suggest that smaller models lead to better results. Thus, the consideration of even simpler model architectures, such as simple RNNs, in addition to extensive hyperparameter tuning runs could be promising for future work. During model training, early stopping is utilized to prevent overfitting and unnecessary long training runs. Evaluation of the

TABLE 2 Top-*k* accuracy of the pretrained model on the generated test set.

Samples in training data	top-1 (%)	top-2 (%)	top-3 (%)	top-4 (%)	top-5 (%)
100	0.0	0.0	0.0	0.0	0.0
199[a]	0.0	0.0	0.0	0.0	0.0
1000	0.3	0.6	0.6	0.8	0.9
1958[a]	0.3	0.3	0.4	0.5	0.6
10,000	37.7	56.3	65.8	72.0	74.8
10,000[b]	17.8	30.2	38.9	44.2	48.0
19,573[a]	41.4	61.1	67.7	73.7	76.1
100,000	48.6	**71.3**	**81.4**	**86.7**	**89.2**
195,467[a]	**49.7**	70.6	80.9	85.7	87.5

[a]Augmented dataset.
[b]Removed valves in input training data.

model is performed by generating predictions with beam search as decoding strategy as described in Section 2.1. The beam width is set to five and those five, most probable predictions are returned to the user as recommendations for possible control structures of the provided PFD. The implementation of a constrained beam search would be possible to prevent the model from predicting unit operations, which are not present in the PFD. However, such an implementation is not applied in the following experiments.

5 | RESULTS AND DISCUSSION

This section summarizes the training procedure for pretraining and fine-tuning the control structure prediction model. Thereafter, the model is evaluated based on the top-*k* accuracy metric.

5.1 | Model training

We perform model pretraining with different generated training set sizes as denoted in Table 2. Additionally, an independent validation and test set is generated with 1000 samples each. During pretraining, we use a learning rate of $3 \cdot 10^{-4}$ and a batch size of 32. Model evaluation is performed depending on the dataset size every 500 steps for the training dataset containing 10,000 and 100,000 samples, every 25 steps for the dataset with 1000 samples, and every five steps for the dataset with 100 samples. Early stopping is applied with patience of 10 steps to prevent overfitting.

Subsequently, we fine-tune the pretrained model on real PFDs with control structures splitting the dataset into a train (80%), validation (10%), and test (10%) set. Model fine-tuning is performed with a reduced learning rate of $0.5 \cdot 10^{-4}$ and a batch size of 2. We evaluate the model every 20 steps and apply early stopping with patience of 40 steps.

Figure 6A illustrates exemplary the training and validation loss curves during model pre-training with a dataset size of 10,000 generated PFDs with decentralized control structures. The first few epochs

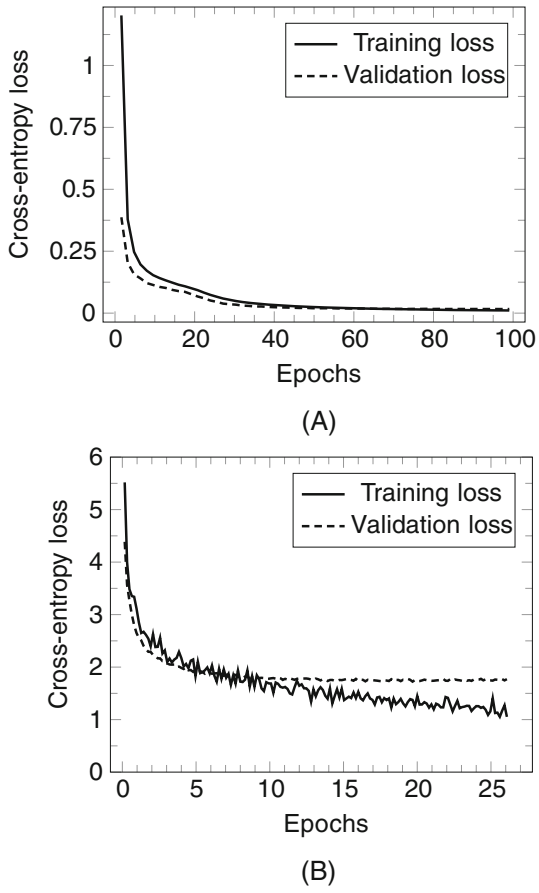


FIGURE 6 Training and validation loss curve during (A) pretraining with 10,000 training samples and (B) fine-tuning.

exhibit a steep decrease of both training and validation loss, whereupon the losses in the subsequent epochs asymptotically approach a constant value. The gap between training and validation loss is small, indicating a small generalization error, which is likely due to the limited variance in the generated dataset. Additionally, the samples of training and validation set are drawn from the same probability distribution and thus forming a representative validation set. The early

stopping callback detects no increase in the validation loss during model training. The small difference between training and validation loss is an indication that overfitting is not observed.

Figure 6B depicts the training and validation loss curves during model fine-tuning. Compared to Figure 6A a larger gap between training and validation loss curve and generally higher fluctuations are observed. This behavior is most likely related due to the training on real PFDs with control structures, which generally exhibit a higher complexity than the generated examples. As indicated in Table 1, the real data show higher variations in the number of nodes and due to additional other unit operations and letter codes in the control structures, resulting in an extended vocabulary size. Along with the small dataset size, the validation set is likely not representative. The early stopping callback detects an increase in the validation loss at around epoch 27 and thus terminates the model training run. The difference between training and validation loss is due to higher variations in the dataset within an expected range but still in the same order of magnitude. The experiments with different dataset sizes during pretraining resulted in qualitatively similar loss curves during pretraining and fine-tuning.

5.2 | Model evaluation

The model performance after pretraining on different generated dataset sizes is evaluated based on the top-*k* accuracy. Therefore, the top-5 predictions are determined with beam search decoding. A prediction is counted as true, if the target PFD with control structure is present in the top-*k* predictions of the model. The results, presented in Table 2, show that increasing the dataset size significantly improves the model performance. It is evident that a dataset size of 100 or 1000 samples is not sufficient for pretraining the control structure prediction model. With 10,000 generated process diagrams, we already reach a top-5 accuracy of roughly 75% on the test set consisting of generated data during pretraining. The top-5 accuracy can be increased up to 89.2% on the test set, consisting of generated data, when pre-training with 100,000 generated samples. Therefore we conclude, that the control structure prediction model learns the grammatical structure of SFILES 2.0, which are based on generated PFDs, and correctly gives recommendations for the control structure of unknown PFDs, corresponding to the generated dataset, through learning the patterns present in the training data. The transformer model learns conditional probabilities and essentially exploits common patterns in the data. It is highlighted that this generated dataset is only based on topological information of the PFDs and no information such as process dynamics, operating points, or equipment sizing is provided therein. Since this generated dataset is created under different assumptions than real PFDs with control structure, these results are not directly transferable to real conditions. Nevertheless, the results indicate successful learning of our proof-of-concept on synthetic data. In addition, the results indicate, that SFILES 2.0 data augmentation has positive effects on the model performance. Especially on the dataset with 10,000 samples, a significant increase in the top-1 accuracy is observed after augmentation.

Since valves are often omitted in PFDs, an additional pretraining run is performed with 10,000 training samples, where the entire control structure and all valves are removed from the input dataset. Thus, the model learns to predict not only the control structures but also the valves. The results, denoted in Table 2, indicate that it is significantly more difficult for the model to predict correct control structures. This causes the top-1 accuracy to decrease from 37.7% (10,000 input samples with valves) to 17.8% (10,000 input samples without valves). However, this demonstrates that the model is also capable of predicting correct valve positions, which are not necessarily present in the PFDs.

In a first experiment, we trained the control structure prediction model directly on 250 real PFDs with control structures. This approach did not yield useful results, as apparently the size of the dataset of 250 real P&IDs is not sufficient to train a transformer-based NLP model. In a second experiment, we applied a transfer learning method. We fine-tuned the control structure prediction model with real PFDs with control structures using checkpoints obtained from pretraining with generated data. Still, the results after fine-tuning revealed a top-5 accuracy of 0% on the test set of real PFDs with control structures. Therefore, based on the top-*k* metric, we cannot demonstrate the utility of the model on real data. While the trained model is not yet applicable to industrial applications, the result is consistent with the results from pre-training. In particular, the pretraining on a small number of generated PFDs with control structures indicates, that 100 or even 1000 training samples are not sufficient for reasonable results (cf. Table 2). The pretraining results highlight, that a sufficiently large number (here, 10,000) of training PFDs with control structures is necessary to enable the model learning patterns in the provided data. Depending on the complexity of real PFDs with control structures the amount of required training samples may be significantly larger due to the increasing complexity of the task.

Error sources and difficulties for the control structure prediction model are not only due to the small real dataset size but also to the dataset composition. The PFDs with control structures are derived from scientific literature and publicly available sources representing laboratory setups, but also chemical plants or fictive examples and may contain errors, incomplete control structures, and wrong, not standardized, letter codes. In addition, the real dataset, as described in Section 3.2, contains very heterogeneous and generally more complex PFDs with control structures. In combination with the small size of the dataset, this leads to errors in the model predictions, including added or missing unit operations, invalid SFILES 2.0, not connected material recycles, or signal connections. These errors could be partly mitigated by implementing a constrained beam search algorithm, which sets the probabilities of unit operations not present in the input sequence to zero and forces the model to add only the control structure and valves to the output sequence. Nevertheless, since for every section of a chemical plant exists at least one PFD and P&ID, we believe that there is enough data available in the proprietary domain to train our control structure prediction model making no arbitrarily changes in the PFD and predicting correct control schemes. Here, a major limitation is the availability of real flowsheet data to academia,

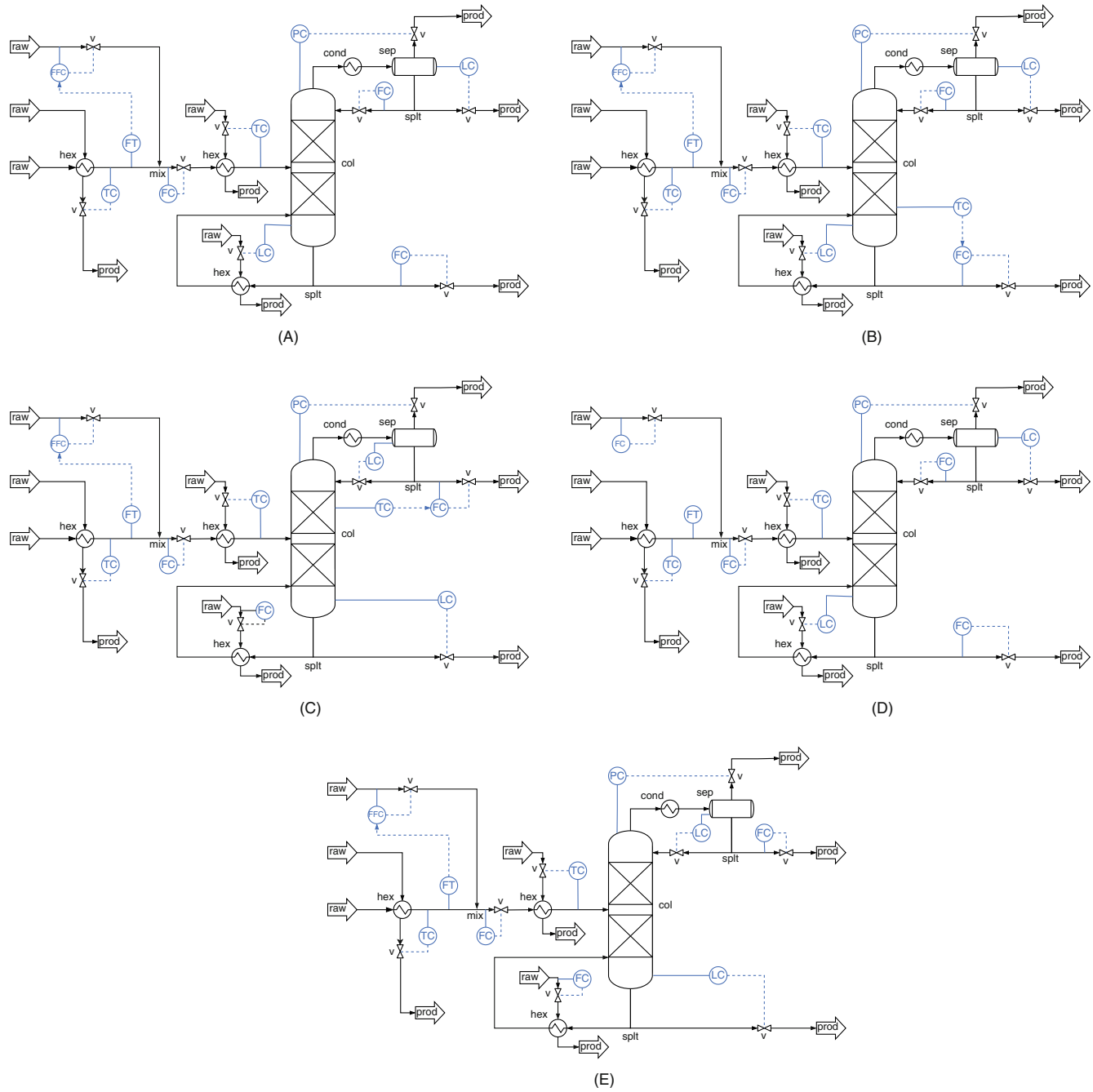


FIGURE 7 Control structure predictions (A–E) (in blue) of the model prompted with the process flow diagram (colored black) as input.

as most of the PFDs and P&IDs in the industry are protected confidential.

5.3 | Illustrative example

This section illustrates the model predictions on one representative sample taken from the independent test set. For this illustrative example, we use the model that has been trained on 10,000 training samples without data augmentation and fine-tuning. The model is prompted with a PFD (colored black in Figure 7) of the test dataset, as denoted in the following SFILES 2.0 string:

(raw)(hex){1}(mix)<&|(raw)(v)&|(v)(hex){2}(rect)<1<2[{tout}(cond)(sep)[(v)(prod)](splt)[(v)(prod)](v)1]{bout}(splt)[(v)(prod)](hex){3}2n|(raw)(hex){1}(v)(prod)n|(raw)(v)(hex){2}(prod)n|(raw)(v)(hex){3}(prod).

The model predicts with beam search decoding the following five, syntactically correct SFILES 2.0. These SFILES 2.0 contain the input PFD colored in black and the predicted, five most-likely control structures illustrated in blue:

1. (raw)(hex){1}(C){TC}_1(C){FT}_2(mix)<&|(raw)(C){FFC}_3<_2(v)&<_3|(C){FC}_4(v)<_4(hex){2}(C){TC}

_5 (rect) <1<2 [(C) {PC}_6] [(C) {LC}_7] [{tout} (cond) (sep) [(C) {LC}_8] [(v) <_6 (prod)] (splt) [(v) <_8 (prod)] (C) {FC}_9 (v) 1<_9] {bout} (splt) [(C) {FC}_10 (v) <_10 (prod)] (hex) {3}2n| (raw) (hex) {1} (v) <_1 (prod) n| (raw) (v) <_5 (hex) {2} (prod) n| (raw) (v) <_7 (hex) {3} (prod)

2. (raw) (hex) {1} (C) {TC}_1 (C) {FT}_2 (mix) <&| (raw) (C) {FFC}_3<_2 (v) & <_3| (C) {FC}_4 (v) <_4 (hex) {2} (C) {TC}_5 (rect) <1<2 [(C) {PC}_6] [(C) {LC}_7] [(C) {TC}_8] [{tout} (cond) (sep) [(C) {LC}_9] [(v) <_6 (prod)] (splt) [(v) <_9 (prod)] (C) {FC}_10 (v) 1<_10] {bout} (splt) [(C) {FC} _11<_8 (v) <_11 (prod)] (hex) {3}2n| (raw) (hex) {1} (v) <_1 (prod) n| (raw) (v) <_5 (hex) {2} (prod) n| (raw) (v) <_7 (hex) {3} (prod)
3. (raw) (hex) {1} (C) {TC}_1 (C) {FT}_2 (mix) <&| (raw) (C) {FFC}_3<_2 (v) & <_3| (C) {FC}_4 (v) <_4 (hex) {2} (C) {TC}_5 (rect) <1<2 [(C) {PC}_6] [(C) {LC}_7] [(C) {TC}_8] [{tout} (cond) (sep) [(C) {LC}_9] [(v) <_6 (prod)] (splt) [(C) {FC}_10<_8 (v) <_10 (prod)] (v) 1<_9] {bout} (splt) [(v) <_7 (prod)] (hex) {3}2n| (raw) (C) {FC}_11 (v) <_11 (hex) {3} (prod) n| (raw) (hex) {1} (v) <_1 (prod) n| (raw) (v) <_5 (hex) {2} (prod)
4. (raw) (C) {FC}_1 (v) <_1 (mix) <&| (raw) (hex) {1} (C) {TC}_2 (C) {FT}&_3| (C) {FC}_4 (v) <_4 (hex) {2} (C) {TC}_5 (rect) <1<2 [(C) {PC}_6] [(C) {LC}_7] [{tout} (cond) (sep) [(C) {LC}_8] [(v) <_6 (prod)] (splt) [(v) <_8 (prod)] (C) {FC}_9 (v) 1<_9] {bout} (splt) [(C) {FC}_10 (v) <_10 (prod)] (hex) {3}2n| (raw) (hex) {1} (v) <_2 (prod) n| (raw) (v) <_5 (hex) {2} (prod) n| (raw) (v) <_7 (hex) {3} (prod)
5. (raw) (hex) {1} (C) {TC}_1 (C) {FT}_2 (mix) <&| (raw) (C) {FFC}_3<_2 (v) & <_3| (C) {FC}_4 (v) <_4 (hex) {2} (C) {TC}_5 (rect) <1<2 [(C) {PC}_6] [(C) {LC}_7] [{tout} (cond) (sep) [(C) {LC}_8] [(v) <_6 (prod)] (splt) [(C) {FC}_9 (v) <_9 (prod)] (v) 1<_8] {bout} (splt) [(v) <_7 (prod)] (hex) {3} 2n| (raw) (C) {FC}_10 (v) <_10 (hex) {3} (prod) n| (raw) (hex) {1} (v) <_1 (prod) n| (raw) (v) <_5 (hex) {2} (prod)

Figure 7 illustrates the five model predictions. The PFD, colored black in Figure 7, contains two feed preheater, a mixing point of two material streams, and a distillation column. The model predicts a temperature-dependent control of the utility stream for both feed preheater. Mixing of the two raw material streams is, according to the model, most likely done with a flow ratio control. Furthermore, the model provides correct predictions of four different distillation column control schemes, which are included in the seven column control structures used to generate the data. The first prediction (Figure 7A) corresponds to the ground truth for the corresponding PFD fed to the model as input.

Apart from the correct predictions, Figure 7 illustrates limitations and errors of the control structure prediction model. In Figure 7D, the model inserts a flow transmitter and fails to predict a corresponding signal connection. In addition, the mixing of the material flows upstream of the distillation column could be problematic from a control perspective, as flow control is proposed here before and after mixing. This problem arises from model training with generated data, which is synthesized by adding small control patterns to a final PFD with control structure. The addition of the utilized control patterns may not result in a meaningful, correct control architecture, and furthermore, no long-range dependencies but only decentralized control structures are considered in the data generation procedure. In addition, correct predictions of the control structure of a distillation column setup depend not only on the topological structure, but also on additional information such as present components, material flows, or operating conditions. This means that control structure predictions considering only the topological structure of the process may appear correct, but are wrong when considering in detail light- and heavy-boiling components, azeotropic mixtures, or quality measures.

Since PFDs do not necessarily contain valves, we trained a model with 10,000 training samples, removing not only the entire control structure but also each valve in the input data. Given the PFD (colored black in Figure 8) and excluding any valve in the model input, the third prediction of the model, as illustrated in Figure 8, represents the ground truth. This shows that our model has also the potential to learn the positioning of valves in combination with the prediction of the control structure.

5.4 | Current limitations and future directions

Overall, our results show great potential for automatically predicting decentralized control structures for PFDs. However, the results also demonstrate several current limitations that need to be overcome for industry applications. A main issue related to the proposed model is the consideration of only topological information of the PFD for the prediction of an appropriate control structure. As already discussed in Section 5.3 specifically for the distillation setup, detailed information about the design characteristics, such as equipment sizing, temperature control (isothermal, adiabatic, non-isothermal), batch or, continuous operation, of every unit operation is necessary to develop a safe and reliable control structure. In addition, the SFILES 2.0 currently misses information about streams, present components, and operating points. Furthermore, process dynamics and operating objectives of a chemical plant need to be considered during the design of the control structure. This missing information severely limits the model performance and could lead to wrong predictions of the control structure.

To overcome the current limitations and challenges, we have identified four main development directions that need to be addressed in order to effectively apply the proposed model to real-world PFDs for control structure prediction in the future.

1. **Data availability:** To train NLP models effectively, a high quantity of training data needs to be accessible for the model. Especially for the prediction of control structures, the collaboration of industry and academia is essential to provide high-quality P&IDs for model training. Ideally, such training data should also be checked/curated thoughtfully. As an alternative, we also explore the automatic

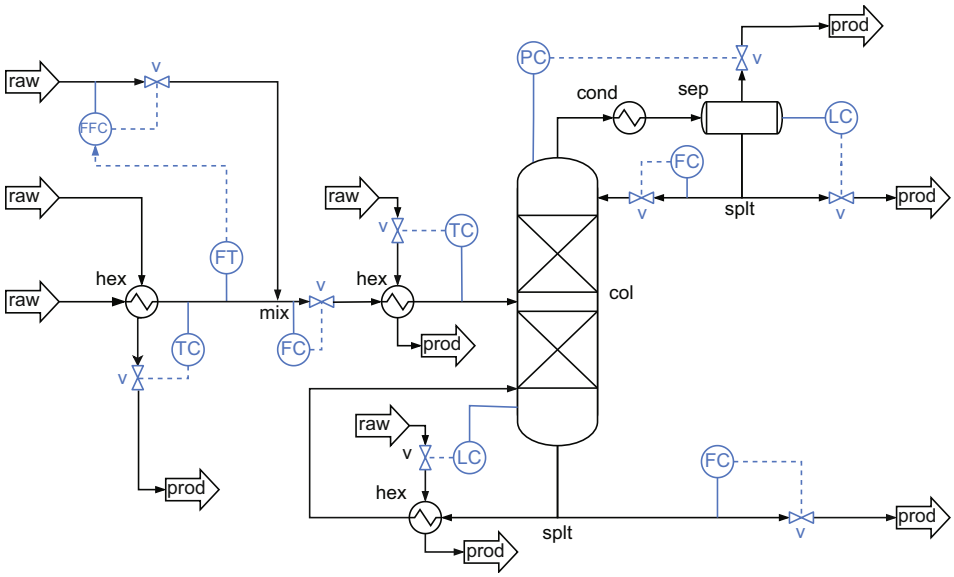


FIGURE 8 Control structure prediction (in blue) of the model prompted with the process flow diagram (colored black) as input.

mining of flowsheets from literature and patents[53] and their automatic digitization.[48]

2. **Inclusion of additional information:** To encode the required information for a successful control structure design, such as operating points, stream information, or present components, the SFILES 2.0 notation needs to be extended. Another possibility is a switch from the language-based to a graph-based model architecture, where the required information about the chemical process is encoded in a graph instead of a string. In our previous work, we showed already that flowsheets and flow information can be represented as graphs.[42] In addition, we already leveraged graph neural networks to learn from these flowsheet graphs.[42] These technologies are also promising in the context of the prediction of decentralized control structures.
3. **Hybrid AI solutions:** Chemical engineers have developed fundamental principles of modeling and control. Integrating these principles into the future AI algorithm has a great potential to improve extrapolability, increase safety and explainability, and reduce data requirements.[16,59] Thus, this integration is an important future research direction.
4. **Validity checks:** Control is safety-critical. Thus, future work requires validity checks for the training data as well as the model predictions. These can be guided by physical knowledge and rules from the process engineering domain,[16] AI approaches,[28] and hybrid AI approaches.[59]

Consequently, addressing the mentioned points in future research will make the model applicable to industrial applications. We envision an integration of our model in CAD software to assist engineers in developing PFDs with control structures.

6 | CONCLUSION

Predicting the control structure of PFDs with machine learning models is a promising strategy to accelerate the development of chemical processes. We propose a novel method of casting the prediction task as a translation task and leveraging the transformer architecture from the field of NLP. To apply NLP techniques, we represent the graph-based process diagrams in the text-based SFILES 2.0 notation. We successfully trained a fully data-driven sequence-to-sequence model to predict the decentralized control structure of generated chemical processes without relying on handcrafted rules. Experiments on 312 real PFDs with control structures indicate that for reasonable results larger datasets are necessary.

Future work should focus on the acquisition of a larger dataset of real PFDs with control structures, which can be used to fine-tune our model and leverage the possible advantages of transfer learning. Additionally, the context of the chemical process, such as operating conditions, basic control structures already present in the PFD, or stream information, may be included in advanced models to refine the prediction of the control structure. Furthermore, training the model on specific classes of plants such as petrochemical or utility systems could be promising since this would decrease the complexity of the prediction task. Besides the prediction of the control structure, extensions of the control structure prediction model could include, for example, pipe classes or valve types, to finally enable the automatic prediction of complete P&IDs. Moreover, validity checks may be included to further increase the accuracy of the model predictions. Ultimately, our model should not be seen as an alternative to the control engineer or existing rule-based systems. According to the UNESCO[60] AI may assist humans in decision-making for efficiency reasons, but an AI algorithm does not replace human responsibility in safety-critical decisions. Therefore, we envision a combination of the algorithm with other process development methods to assist the engineer with recommendations, reduce the number of manual tasks, and generally make process development more efficient.

AUTHOR CONTRIBUTIONS

Edwin Hirtreiter: data curation (lead); investigation (equal); methodology (equal); software (lead); validation (lead); visualization (lead);

writing – original draft (lead). **Lukas Schulze Balhorn:** conceptualization (supporting); investigation (equal); methodology (supporting); software (supporting); supervision (supporting); writing – review and editing (equal). **Artur M. Schweidtmann:** conceptualization (lead); funding acquisition (lead); project administration (lead); supervision (lead); writing – review and editing (equal).

ACKNOWLEDGMENTS

This publication is part of the project "ChemEng KG - The Chemical Engineering Knowledge Graph" with project number 203.001.107 of the research program "Open Science (OS) Fund 2020/2021" which is (partly) financed by the Dutch Research Council (NWO). We want to thank the anonymous reviewers for their time and effort in evaluating this article. This greatly helped us to clarify and improve our publication.

DATA AVAILABILITY STATEMENT

The synthetic data that support the findings of this study are openly available in zenodo at https://doi.org/10.5281/zenodo.7658798, reference number 7658798. The real training data, code, and trained models are not shared.

ORCID

Edwin Hirtreiter https://orcid.org/0000-0002-0693-925X
Lukas Schulze Balhorn https://orcid.org/0000-0001-7494-9110
Artur M. Schweidtmann https://orcid.org/0000-0001-8885-6847

REFERENCES

1. Towler GP, Sinnott RK. *Chemical Engineering Design - Principles, Practice and Economics of Plant and Process Design*. Elsevier/Butterworth-Heinemann; 2008.
2. Toghraei M. *Piping and Instrumentation Diagram Development*. 1st ed. John Wiley & Sons Inc; 2019.
3. Uzuner H, Schembecker G. Wissensbasierte erstellung von R&I-fließbildern. *Chem Ing Tech*. 2012;84:747-761. doi:10.1002/cite.201100230
4. Blitz H, Engelke J, Sonnenschein R, Schmidt-Traub H. Rechnergestützte konfigurierung von RI-fließbildern am beispiel von pumpen. *Chem Ing Tech*. 1994;66:470-475. doi:10.1002/cite.330660404
5. Obst M, Doherr F, Urbas L. Wissensbasiertes Assistenzsystem für modulares engineering. *Automatisierungstechnik*. 2013;61:103-108. doi:10.1524/auto.2013.0011
6. Fleischer-Trebes C, Krasberg N, Bramsiepe C, Kockmann N. Planungsansatz für modulare anlagen in der chemischen Industrie. *Chem Ing Tech*. 2017;89:785-799. doi:10.1002/cite.201600083
7. Hohmann L, Kössl K, Kockmann N, Schembecker G, Bramsiepe C. Modules in process industry – a life cycle definition. *Chem Eng Process Process Intens*. 2017;111:115-126. doi:10.1016/j.cep.2016.09.017
8. Eilermann M, Post C, Radatz H, Bramsiepe C, Schembecker G. A general approach to module-based plant design. *Chem Eng Res Des*. 2018;137:125-140. doi:10.1016/j.cherd.2018.06.039
9. Morari M, Arkun Y, Stephanopoulos G. Studies in the synthesis of control structures for chemical processes: part I: formulation of the problem. Process decomposition and the classification of the control tasks. Analysis of the optimizing control structures. *AIChE J*. 1980;26:220-232. doi:10.1002/aic.690260205
10. Luyben ML, Tyreus BD, Luyben WL. Plantwide control design procedure. *AIChE J*. 1997;43:3161-3174. doi:10.1002/aic.690431205
11. Ng C, Stephanopoulos G. Synthesis of control structures for chemical plants. *8th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems: Theory and Applications*. Vol 31. Elsevier; 1998:17-29. doi:10.1016/S1474-6670(17)41767-8
12. Seborg DE, Edgar TF, Mellichamp DA, Doyle FJ III. *Process Dynamics and Control*. 3rd ed. Wiley; 2011.
13. van de Wal M, de Jager B. A review of methods for input/output selection. *Automatica*. 2001;37:487-510. doi:10.1016/S0005-1098(00)00181-3
14. Williamson C. *Computer Aided Process Control Systems Synthesis Using Rule-Based Programming*. Dissertation. University of Canterbury. 1989.
15. Song JJ, Park SW. Intellite3: a knowledge based expert system for control structure synthesis. *Korea J Chem Eng*. 1990;7:198-209. doi:10.1007/bf02697352
16. Venkatasubramanian V. The promise of artificial intelligence in chemical engineering: is it here, finally? *AIChE J*. 2019;65:466-478. doi:10.1002/aic.16489
17. Brown TB, Mann B, Ryder N, et al. Language models are few-shot learners. *arXiv*. 2020:1-75. doi:10.48550/arXiv.2005.14165
18. Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J Mach Learn Res*. 2019;21:1-67. doi:10.48550/arXiv.1910.10683
19. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. *arXiv*. 2017:1-15. doi:10.48550/arXiv.1706.03762
20. Popel M, Tomkova M, Tomek J, et al. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nat Commun*. 2020;11:1-15. doi:10.1038/s41467-020-18073-9
21. Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J Chem Inf Model*. 1988;28:31-36. doi:10.1021/ci00057a005
22. Weininger D, Weininger A, Weininger JL. SMILES. 2. Algorithm for generation of unique SMILES notation. *J Chem Inf Comput Sci*. 1989;29:97-101. doi:10.1021/ci00062a008
23. Schwaller P, Gaudin T, Lányi D, Bekas C, Laino T. "Found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chem Sci*. 2018;9:6091-6098. doi:10.1039/c8sc02339e
24. Schwaller P, Laino T, Gaudin T, et al. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS Central Sci*. 2019;5:1572-1583. doi:10.1021/acscentsci.9b00576
25. Schwaller P, Petraglia R, Zullo V, et al. Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy. *Chem Sci*. 2020;11:3316-3325. doi:10.1039/c9sc05704h
26. Zhang T, Sahinidis NV, Siirola JJ. Pattern recognition in chemical process flowsheets. *AIChE J*. 2019;65:592-603. doi:10.1002/aic.16443
27. Zheng C, Chen X, Zhang T, Sahinidis NV, Siirola JJ. Learning process patterns via multiple sequence alignment. *Comput Chem Eng*. 2022;159:107676. doi:10.1016/j.compchemeng.2022.107676
28. Oeing J, Welscher W, Krink N, Jansen L, Henke F, Kockmann N. Using artificial intelligence to support the drawing of piping and instrumentation diagrams using DEXPI standard. *Digit Chem Eng*. 2022;4:100038. doi:10.1016/j.dche.2022.100038
29. Vogel G, Schulze Balhorn L, Schweidtmann AM. Learning from flowsheets: a generative transformer model for flowsheet autocompletion. *arXiv*. 2022:1-31. doi:10.48550/arXiv.2208.00859
30. d'Anterroches L. *Process Flow Sheet Generation and Design through a Group Contribution Approach*. Dissertation. Technical University of Denmark. 2006.
31. Vogel G, Schulze Balhorn L, Hirtreiter E, Schweidtmann AM. SFILES 2.0: an extended text-based flowsheet representation. *arXiv*. 2022:1-13. doi:10.48550/arXiv.2208.00778
32. Stahlberg F. Neural machine translation: a review. *J Artif Intell Res*. 2020;69:343-418. doi:10.1613/jair.1.12007
33. Nallapati R, Zhou B, CNd s, Gulcehre C, Xiang B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv*. 2016. doi:10.48550/arXiv.1602.06023

34. Chiu CC, Sainath TN, Wu Y, et al. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv*. 2017:1-5. doi:10.48550/arXiv.1712.01769
35. Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions. *arXiv*. 2014:1-17. doi:10.48550/arXiv.1412.2306
36. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger K, eds. *Advances in Neural Information Processing Systems*. Curran; 2014:1-9. doi:10.48550/ARXIV.1409.3215
37. Graves A. Sequence transduction with recurrent neural networks. *arXiv*. 2012:1-9. doi:10.48550/arXiv.1211.3711
38. Boulanger-Lewandowski N, Bengio Y, Vincent P. High-dimensional sequence transduction. *arXiv*. 2012; 1936:1-5. doi:10.48550/arXiv.1212
39. Williams RJ, Zipser D. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput*. 1989;1:270-280. doi:10.1162/neco.1989.1.2.270
40. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9:1735-1780. doi:10.1162/neco.1997.9.8.1735
41. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*. 2014:1-9. doi:10.48550/arXiv.1412.3555
42. Stops L, Leenhouts R, Gao Q, Schweidtmann AM. Flowsheet synthesis through hierarchical reinforcement learning and graph neural networks. *arXiv*. 2022:1-22. doi:10.48550/arXiv.2207.12051
43. Vogel G, Schulze Balhorn L, Hirtreiter E, Schweidtmann AM. Process-intelligence-research/SFILES2: v1.0.0. 2022. doi:10.5281/zenodo.6901932
44. Perry RH, Green DW, Maloney JO. *Perry's Chemical Engineers' Handbook*. 7th ed. McGraw-Hill; 1997.
45. Stichlmair J. *Konzeptuelle Prozesssynthese*. Lecture Script. Technische Universität München. 2020.
46. Hirtreiter E, Schulze Balhorn L, Schweidtmann AM. Supplementary data for: "Towards automatic generation of control structures for Process Flow Diagrams (PFDs) with Artificial Intelligence". 2023. doi:10.5281/zenodo.7658798
47. Schulze Balhorn L, Gao Q, Goldstein D, Schweidtmann AM. Flowsheet recognition using deep convolutional neural networks. In: Yamashita Y, Kano M, eds. *Proceedings of the 14th International Symposium on Process Systems Engineering - PSE 2021*. Vol 49. Elsevier; 2022:1567-1572. doi:10.1016/B978-0-323-85159-6.50261-X
48. Theisen M, Flores KN, Schulze Balhorn L, Schweidtmann AM. Digitization of chemical process flow diagrams using deep convolutional neural networks. *Digit Chem Eng*. 2023;6:100072. doi:10.1016/j.dche.2022.100072
49. LabelImg. LabelImg v1.8.1. 2018 https://github.com/heartexlabs/labelImg. Accessed September 10, 2022.
50. Sennrich R, Haddow B, Birch A. Improving neural machine translation models with monolingual data. In: Erk K, Smith NA, eds. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics; 2016:86-96. doi:10.18653/v1/P16-1009
51. Wei J, Zou K. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In: Inui K, Jiang J, Ng V, Wan X, eds. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics; 2019:6382-6388. doi:10.18653/v1/D19-1670
52. Feng SY, Gangal V, Wei J, et al. A survey of data augmentation approaches for NLP. *arXiv*. 2021:1-21. doi:10.48550/arXiv.2105.03075
53. Schulze Balhorn L, Hirtreiter E, Luderer L, Schweidtmann AM. Data augmentation for machine learning of chemical process flowsheets. *arXiv*. 2022:1-6. doi:10.48550/arXiv.2302.03379
54. Bjerrum EJ. SMILES enumeration as data augmentation for neural network Modeling of molecules. *arXiv*. 2017:1-7. doi:10.48550/ARXIV.1703.07076
55. Tetko IV, Karpov P, van Deursen R, Godin G. State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis. *Nat Commun*. 2020;11:5575. doi:10.1038/s41467-020-19266-y
56. Gage P. A new algorithm for data compression. *C Users J Arch*. 1994; 12:23-38. doi:10.5555/177910.177914
57. Kudo T, Richardson J. SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. In: Blanco E, Lu W, eds. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics; 2018:66-71. doi:10.18653/v1/D18-2012
58. Klein G, Kim Y, Deng Y, Senellart J, Rush A. OpenNMT: open-source toolkit for neural machine translation. *Proceedings of ACL 2017, System Demonstrations*. Association for Computational Linguistics; 2017:67-72.
59. Schweidtmann AM, Esche E, Fischer A, et al. Machine learning in chemical engineering: a perspective. *Chem Ing Tech*. 2021;93:2029-2039. doi:10.1002/cite.202100083
60. UNESCO. *Recommendation on the Ethics of Artificial Intelligence*. UNESCO; 2021.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Hirtreiter E, Schulze Balhorn L, Schweidtmann AM. Toward automatic generation of control structures for process flow diagrams with large language models. *AIChE J*. 2024;70(1):e18259. doi:10.1002/aic.18259