Domain-informed Language Models for Process Systems Engineering

Vipul Mann

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2024

Abstract

Domain-informed Language Models for Process Systems Engineering

Vipul Mann

Process systems engineering (PSE) involves a *systems-level* approach to solving problems in chemical engineering related to process modeling, design, control, and optimization and involves modeling interactions between various systems (and subsystems) governing the process. This requires using a combination of mathematical methods, physical intuition, and recently machine learning techniques. Recently, language models have seen tremendous advances due to new and more efficient model architectures (such as transformers), computing power, and large volumes of training data. Many of these language models could be appropriately adapted to solve several PSE-related problems. However, language models are inherently complex and are often characterized by several million parameters, which could only be trained efficiently in data-rich areas, unlike PSE. Moreover, PSE is characterized by decades of rich process knowledge that must be utilized during model training to avoid mismatch between process knowledge and data-driven language models. This thesis presents a framework for building domain-informed language models for several central problems in PSE spanning multiple scales. Specifically, the frameworks presented include molecular property prediction, forward and retrosynthesis reaction outcome prediction, chemical flowsheet representation and generation, pharmaceutical information extraction, and reaction classification. Domain knowledge is integrated with language models using custom model architectures, standard and custom-built ontologies, linguistics-inspired chemistry and process flowsheet grammar, adapted problem formulations, graph theory techniques, and so on. This thesis is intended to provide a path for future developments of domain-informed language models in process systems engineering that respect domain knowledge, but leverage their computational advantages.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This thesis is a culmination of the last 4 years that were filled with excitement, learning, and perseverance. The ebbs and flows of my Ph.D. journey has imbibed in me a confidence to pursue research and humility to acknowledge its vastness. I attribute all of my learnings to my advisor, Professor Venkat Venkatasubramanian, who I found to be the most inspiring, motivating, and supporting person throughout my Ph.D journey. I am grateful to Prof. Venkat for his collaborative mentoring, numerous industry and academic collaborations, teaching opportunities, and most of all, for letting me have a part of his infectiously inspiring ideas. It was due to interesting discussions with him that there was never a moment of monotonocity and the long journey did not feel *that long*.

I am also thankful to our collaborator, Professor Rafiqul Gani from the Technical University of Denmark, without whose support, advising, and mentoring, two of the major contributions in this thesis would not have been possible. I have been fortunate to learn from his great attention to detail, both in terms of conducting and presenting research. After two years of collaboration, I still continue to learn from every interaction with him.

I wish to express my sincere gratitude to Professor Arun K. Tangirala, Professor Shankar Narasimhan, and Professor Raghunathan Rengaswamy, who first introduced me to research during my undergraduate studies at the Indian Institute of Technology Madras. I have utmost gratitude and respect for the effort that they put in early in my research career that shaped my research mindset, critical thinking, and most importantly, writing.

I would like to thank our collaborators Dr. Shekhar Viswanath and Dr. Shankar Vaid-

yaraman from Eli Lilly, and Professor Thomas Friedli and Matteo Bernasconi from University of St. Gallen, Switzerland, for their enriching discussions and contributions to my work. It was due to interactions with them that I gained important industry perspective on my research.

I also wish to thank the past and current members of the Complex Resilient Intelligent Systems lab for their support throughout the years. I wish to thank my childhood friends, Chetan and Yogesh, and my friends from undergrad, Piyush and Sanjay, for constantly being a friendly escape from research over the years. I would also like to sincerely thank my former colleague and mentor, Ganesh Sankaran, for training me rigorously in professional programming that made several aspects of my thesis easier.

Finally, and most importantly, I would like to thank my family for supporting me during the numerous ups and downs of my journey over the last 4 years, and my wife, Minal, for being my biggest cheerleader, a sounding board for several research and non-research ideas, and a *partner* in the truest sense. This thesis would not have been possible without their support.

Dedication

Dedicated to my family, and Minal,

for their unwavering love and support,

that made this thesis possible.

Chapter 1: Introduction

Process systems engineering (PSE) is a subdomain in chemical engineering that involves applying *systems thinking* principles for process modeling, design, and control. Systems thinking focuses on the design, integration, and management of complex systems, where there is often a need to study the relationships and interactions within a system as a whole rather than modeling individual components in isolation. Process systems engineering, therefore, involves the formulation of processing engineering related problems from a systems thinking standpoint, which either results in a precise mathematical formulation that could be solved using algorithms based on the mathematical structure of the problem, or formulation based on physical intuition, as defined by Sargent (1983) [1].

The scope of PSE is quite broad and traditionally involves problems related to process synthesis, design, analysis, operations, control, optimization, and so on. With increasing development and adoption of computer-aided, data-driven methods in various engineering domains, the scope of PSE has expanded further to areas such as pharmaceutical product discovery, chemical product design, integration of product-process design, techno-economic and sustainability-focused design, and many more [2, 3]. Gani et al. [4] provides a multi-layered view of PSE with an inner core layer comprising process-product-related activities, the middle layer involving resource-efficiency-related activities, and an outer layer focusing on societal sustainability factors.

For solving a wide range of problems on multiple scales, the PSE community has been developing and using advanced numerical methods such as optimization, multivariate analysis, uncertainty quantification, integer programming, and dynamic programming for a long time. Recently, there has been a surge in the use of machine learning and deep learning approaches to solve these problems due to their advantages over traditional mathematical

formulations, especially in terms of computational efficiency. AI and ML-based methods have been developed for problems in computer-aided molecular design (CAMD), reaction network generation and synthesis planning, ontology-based material informatics, process optimization, pharmaceutical drug development and discovery, and many more. Within the area of machine learning, language models are an emerging class of models that were originally proposed for modeling natural language, where the primary goal is to build data-driven models for natural language-oriented tasks such as machine translation, word representation, text summarization, sentiment classification, and other text data-related problems. Language models offer a new alternative mathematical formulation for solving several related problems in PSE from product development to process synthesis and optimization after appropriate adaptations, and thus have started to appear in the PSE domain.

However, the advantages of language models in their current form could only be realized if abundant training data are available since they are often characterized by parameters on the order of several millions. However, PSE traditionally being not a big-data field compared to natural language domain where terabytes of training data is readily available, efficiently training such language models becomes challenging. In addition, language models are typically purely data-driven in nature, suitable for domains like computer science where a physics and chemistry-based understanding of the underlying systems does not exist. On the other hand, this is not the case with PSE and other engineering domains, where extensive domain knowledge is readily available. Thus, for the success and wider adoption of language models in PSE, it is imperative that they not be developed in isolation, leading to systems that are disconnected from theory.

In this thesis, in order to integrate prior domain knowledge with language models, domain-informed language modeling frameworks are developed for solving a range of multiscale problems in process systems engineering. Such domain-informed language models are demonstrated for problems including molecular property prediction, computer-aided reaction synthesis, process flowsheet representation and generation, pharmaceutical information extrac-

tion, and the study of the network of organic chemistry. The thesis comprises a total of 6 chapters. Chapter 2 presents a molecular property prediction framework that involves learning dense vector representations for molecules using a word2vec-like framework called grammar2vec that is based on the SMILES grammar. These representations are used to build interpretable machine learning models for prediction of physical properties such as boiling point and critical temperatures. Chapter 3 presents chemistry-informed transformer models for efficiently solving forward and retrosynthesis reaction prediction problems using chemistry-informed representations and chemistry-adapted transformer architecture with nearly state-of-the-art performance. The forward and retrosynthesis reaction prediction problems are formulated as sequence-to-sequence and tree-to-sequence modeling problems, respectively. Chapter 4 presents a novel hierarchical framework for chemical flowsheet representation at three levels with a base level using text-based, hypergraph-based, and hypergraph-connected flowsheet representations based on process ontology. A novel chemical flowsheet grammar inspired by linguistics that could be used for consistent and accurate process flowsheet synthesis is also developed. Chapter 5 presents an approach named schema-based unsupervised semantic information extraction (or SUSIE) that automatically extracts relevant information from pharmaceutical documents and creates knowledge graphs that represent important information semantically. This framework has been tested on public and proprietary documents on a variety of tasks such as entity recognition, relation extraction, text summarization, and efficient search. Chapter 6 presents a study of the network of organic chemistry from a graph-theoretic standpoint, where reactions are represented as hypergraphs. A study on the time-evolution of network statistics and its applications in reaction classification is presented. Finally, a summary of the thesis, concluding remarks, and perspective on the future directions appear in the Epilogue.

Chapter 2: Molecular Property Prediction

Thermodynamic properties are crucial for efficient process design, optimization, control, and monitoring, the latter two being extremely important in safety-critical conditions. A reliable (and often preferable) means of acquiring properties is measuring them through controlled experimentation that achieves the desired level of accuracy and precision. However, with increasing complexity and number of compounds synthesized at a rapid pace, along with a combinatorially large number of possible mixtures, it is nearly impractical to perform costly and time-consuming experiments for all of them. An alternative solution for mitigating this issue is to build fairly accurate data-driven models that could be used to estimate such properties. As argued by Venkatasubramanian and Mann [5], the most useful data-driven methods are those that combine domain knowledge (in the form of symbolic information) with numeric machine learning.

Quantitative structure-activity/property relationships (QSAR/QSPR) methods utilize the correlations between molecular properties and their structural descriptors for data-driven property estimation. Although, these methods are statistical in nature and are characterized by difficult mathematical formulations and property-specific nature that limited their wider generalization capabilities [6], the commonly used Group Contribution (GC)-based methods, which could be regarded as a special class of QSAR/QSPR methods, are simple, easy to use and predictive in nature. The underlying assumption in group contribution methods is that the property of a compound is a function of its molecular structure and the property can be determined by summing the contributions of the groups representing the molecule for a specific property. GC-based methods have limitations from the standpoint of accuracy, applicability to complex molecular structures, isomer distinction, and so on [7].

Recently, machine learning (ML) methods have emerged as powerful alternative for tack-

ling the thermodynamic property estimation problem. These methods generally use artificial neural networks, support vector regression, or deep learning approaches involving autoencoders, variational autoencoders, graph neural networks, and so on. Such methods have been reportedly used for predicting a wide range of properties such as CO_2 solubility, density and viscosity of potassium lysinate and the mixed solutions with monoethanolamine [8], predicting standard enthalpies of formation for hydrocarbons [9], density and viscosity of biofuel compounds [10], and and, GC-based machine learning modeling of 25 pure component properties of organic compounds [11].

One of the important steps in building such data-driven ML models is using an appropriate representation for molecules that captures their underlying structural and chemical characteristics. This task could either be performed manually based on domain knowledge by using expert-curated features for each property individually, or performed automatically in a latent space, referred to as representation learning [12]. Such representations (or molecular features) are fed to the data-driven models with the target variable as the property of interest for a molecule. In the area of chemistry and drug discovery, the common representations for molecules are Morgan fingerprints [13], SMILES strings [14], molecular graphs [15], and SMILES grammar [16]. At the present, there is no consensus on which molecular representation is most suited for machine learning-based property prediction problems, making this an interesting field of research.

A Grammar2vec framework has been developed to address such issues related to molecular representation, which generates dense vector representations of molecules using the grammar rules defining SMILES representations to develop ML-based property estimation models. The performance of these models is evaluated in terms of SMILES and SMILES grammar-based representation of molecules. Their performance is benchmarked on two thermodynamic properties – normal boiling point (T_b) and critical temperature (T_c). To ensure a fair comparison (fixing the number of features in each representation), the natural language analogy is invoked, and molecules are looked at as sentences, and the underlying atomic units are seen as

words to generate their fixed-size vector representations. These vector representations are the same for each molecule (irrespective of the property being predicted), which are then used to build separate ML-based regression models. In addition, a systematic study is performed using Shapley values [17] to understand the feature importance and prune the model further, giving rise to a simpler and relatively interpretable model. This approach of generating dense vector representations of molecules using the grammar rules underlying the SMILES representation is referred to as the Grammar2vec framework.

2.1 Problem statement and objectives

The thermodynamic property estimation problem is formulated as a regression task in which the objective is to create a machine learning-based regression model between the regressors and the target variable as,

$$\hat{y}_i = \mathbf{f}(\mathbf{x_i}, \boldsymbol{\beta}) + e_i \tag{2.1}$$

where (for the i^{th} molecule) the target variable $\hat{y}_i$ is the predicted thermodynamic property of interest; $\mathbf{x_i}$ is the vector representation of the molecule and is of dimension $m \times 1$; $\boldsymbol{\beta}$ is a vector of regression coefficients (with appropriate dimensions depending on the model-form) that is estimated; and e_i is the noise in the prediction that could be attributed to the measurement errors in the training data and/or the modeling inaccuracies. The former could be inferred by performing an uncertainty analysis as demonstrated for GC-models in [18].

Based on Equation 2.1 above, the two important aspects for estimating the target value y_i accurately are – choosing an appropriate functional transformation $\mathbf{f}(.)$ and a using rich molecular representation $\mathbf{x_i}$. To model the former, a support vector regression (SVR) model with a radial basis function (RBF) kernel is used, while for the latter, a rich, property-agnostic molecular representation $\mathbf{x_i}$ is generated using the proposed Grammar2vec framework. In addition, since our objective is to build interpretable machine learning models, the feature

importance and contribution towards the predictions are analyzed by computing Shapley values. This information is used to prune the model and simplify it as much as possible while retaining a similar performance, thus giving rise to a relatively simpler yet powerful model architecture. The kernel SVR model, the Grammar2vec framework, and Shapley values are described in detail in Sections 2.2.3, 2.2.2, and 2.2.4 respectively. An overview of the proposed algorithmic framework is presented in Figure 2.1.

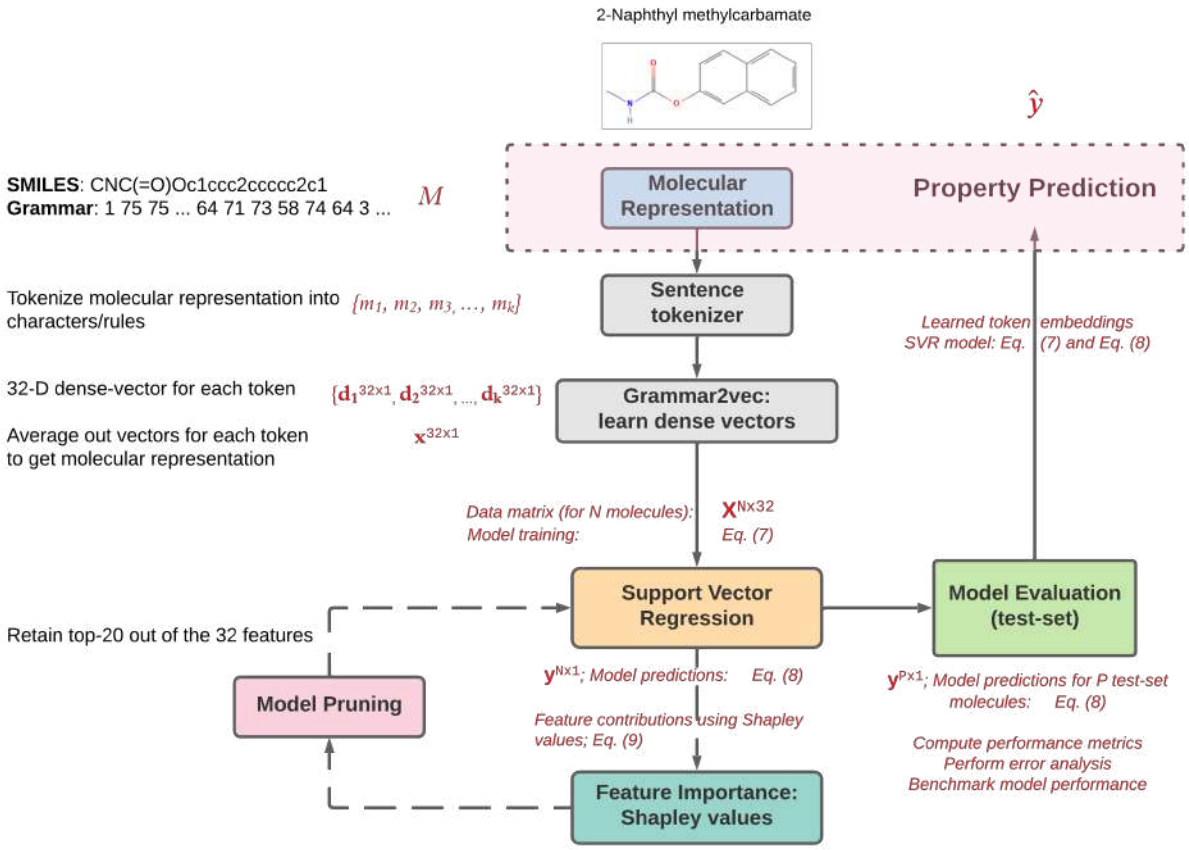


Figure 2.1: An overview of our algorithmic framework used for the property prediction problem.

For the example molecule 2-Napthyl methylcarbamate, the two different representations based on the SMILES and SMILES grammar representations are indicated in Figure 2.1. These representations are tokenized (split into individual characters) and passed to two separate Grammar2vec models as 'words' that learns their respective 32-dimensional (dense)

vector representations which are then transformed to a single 32-dimensional dense vector representing the entire molecule by performing element-wise averaging. These 32 features are then used in the kernel SVR model to estimate the property of interest (say T_c). While this model could directly be used for property prediction, the model is simplified by first computing the Shapley values to understand the relative importance of the 32 features towards the regression task and then prune the model to drop the relatively unimportant features. The final, simplified model is then used for the property prediction task. The equations and transformations required at each stage are indicated in Figure 2.1. The following sections provide details on each of these modules.

2.2 Methods underlying grammar2vec-based property estimation

In this section, a brief overview of the various methods used in our work at different stages in the algorithmic framework described in Figure 2.1 is provided. These methods include the different molecular representations, the natural language analogy and the proposed Grammar2vec framework, the kernel SVR model used for estimating the properties, and the Shapley values approach used for computing feature importance and performing model pruning.

2.2.1 Molecular representations: SMILES and SMILES grammar

The SMILES representation [14] is commonly used in deep-learning approaches including drug discovery, because of their ability to encode molecular structural information as text using short ASCII strings. Several approaches that have utilized this representations in frameworks adapted from natural language process include the SMILES transformer [19], Mol2vec [20] for generating molecular representations inspired from [21], and SMILES2vec [22] for chemical property estimation using deep recurrent neural networks, and several other works on property estimation [23, 24, 25, 26]. The major shortcoming of the SMILES representation, however, is their lack of explicit incorporation of the entire molecular structure of

the molecule (including the 3D structure and stereochemistry). It is often assumed (incorrectly) that the machine learning model would discover the underlying relationships between the SMILES characters, resultig in information loss and consequently suboptimal model performance. This issue is addressed to a large extent by utilizing the underlying SMILES grammar production rules.

The SMILES grammar specifies the underlying production rules that are required to generate the SMILES string of a given molecule. These productions rules are much more detailed and richer when compared to individual, purely character-based SMILES characters (or tokens). The SMILES grammar is similar to the context-free grammar (CFG) commonly used in the area of natural language processing, first formalized by Chomsky [27]. A CFG is a finite collection of recursive rules (or productions) that defines the set of all well-formed sentences in a language. Formally, a CFG is a 4-tuple $G = (V, \Sigma, R, S)$, where V is a finite non-empty set of non-terminal symbols, Σ is a finite set of terminal symbols, R is a finite non-empty set of rules, and S is a designated start symbol. Each rule has a left-hand side (a single non-terminal) and a right-hand side (a sequence of one or more non-terminal or terminal symbols). A CFG can also be deduced for SMILES strings and was used for molecule optimization in [16] in which the parse tree is used as a representation for the molecules [28]. A representative subset of the SMILES grammar is shown in Figure 2.1 and the set of rules required to generate the SMILES string for propene (CC=C) is shown in Figure 2.2 in the form of a parse tree. A SMILES grammar-based representation could be constructed from the grammar parse tree by extracting the sequence of production rules in a depth-first strategy as indicated in the caption of Figure 2.2.

Table 2.1: A subset of the SMILES grammar productions. The complete SMILES grammar used in this work is presented in the Appendix A.

S.No	Production rules
1	SMILES ⟶ CHAIN
2	CHAIN ⟶ CHAIN BRANCHED_ATOM
3	CHAIN ⟶ CHAIN BOND BRANCHED_ATOM
4	CHAIN ⟶ BRANCHED_ATOM
5	BRANCHED_ATOM ⟶ ATOM RINGBOND
6	BRANCHED_ATOM ⟶ ATOM
7	BRANCHED_ATOM ⟶ ATOM BB
8	BRANCHED_ATOM ⟶ ATOM RB
9	BB ⟶ BRANCH
10	RB ⟶ RINGBOND
11	BRANCH ⟶ (CHAIN)
12	RINGBOND ⟶ DIGIT
13	BOND ⟶ =
14	ATOM ⟶ AROMATIC_ORGANIC
15	ATOM ⟶ ALIPHATIC_ORGANIC
16	AROMATIC_ORGANIC ⟶ c
17	ALIPHATIC_ORGANIC ⟶ C
18	ALIPHATIC_ORGANIC ⟶ O

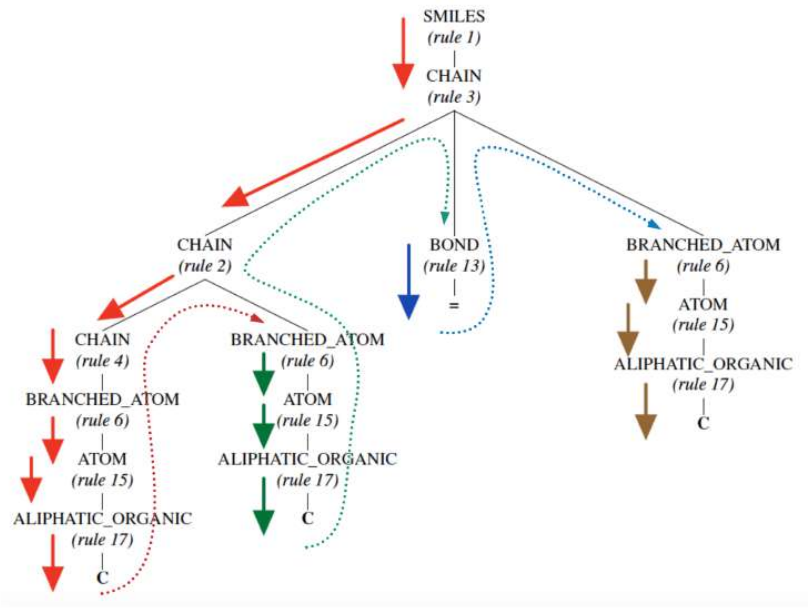


Figure 2.2: The parse-tree obtained for propene (SMILES representation: `CC=C`) using SMILES grammar productions in Table 3.1. The productions are extracted from the parse tree in a depth-first manner, resulting in the grammar representation for propene to be 1, 3, 2, 4, 6, 15, 17, 6, 15, 17, 13, 6, 15, 17

The SMILES grammar-based representations incorporate chemical and structural information about molecules, which cannot be done with purely character-based SMILES strings, and are shown to be superior from an information-theoretic standpoint [29]. The grammar rules, that often correspond to underlying structural chemistry of molecules, explicitly encode that information in the representation making them much more richer as compared to SMILES strings. The complete SMILES grammar used in our work is presented in the Appendix A. These grammar rules are transformer into dense, numeric vectors using the Grammar2vec framework as described in the next section.

2.2.2 Grammar2vec framework

Natural language processing techniques involve transforming text into its equivalent numeric representation that preserves the underlying properties (context, meaning, structure) to the maximum possible extent. One approach to achieve this is generating word embeddings that learn dense vector representations of text (or words) using neural networks by

preserving the contextual information of words in a corpus. Word2vec [21] is one of the most commonly used methods used to generate word embeddings and is at the core of the Grammar2vec framework. It encodes words in a high dimensional vector space such that words with higher semantic similarity are closer in the vector space. The Word2vec model can be trained using two different approaches – skip-gram and a continuous bag of words (CBOW). The skip-gram approach involves using the current word to predict its contextual words, whereas the CBOW approach involves predicting the current word based on the context words. In our work, the CBOW approach is used because of its better performance and faster computation.

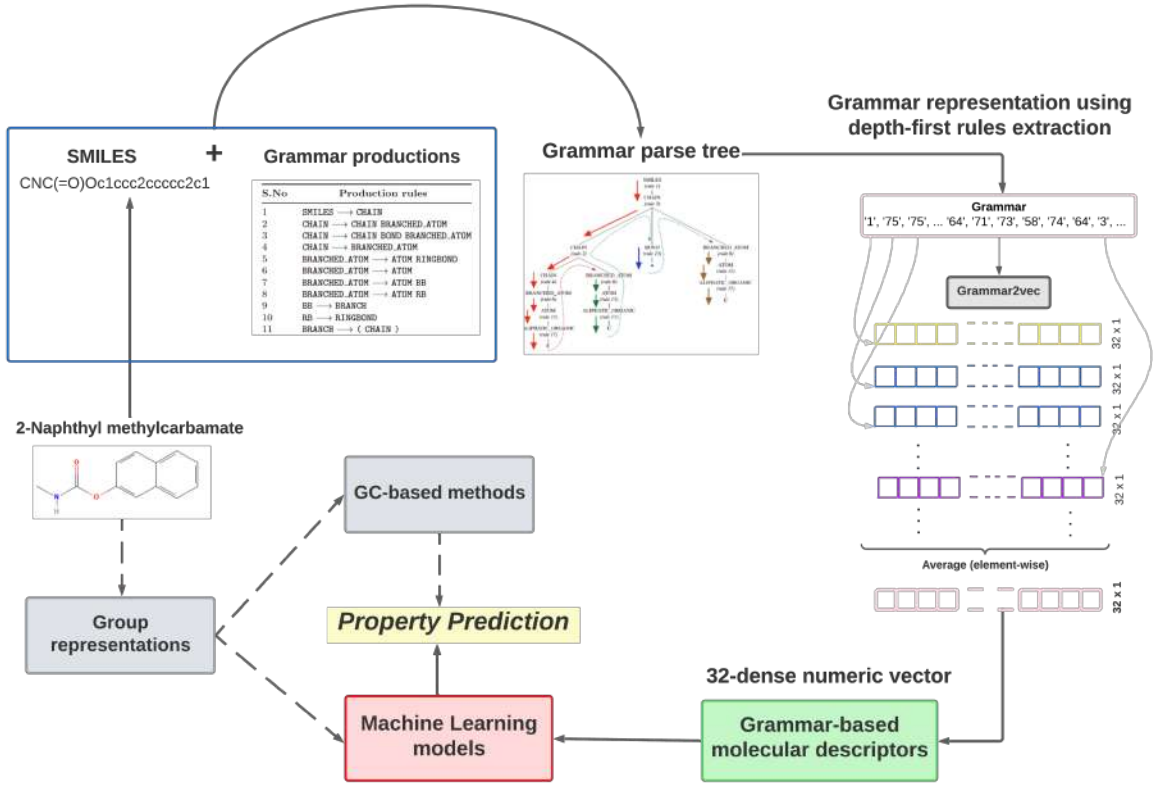


Figure 2.3: An overview of the proposed Grammar2vec framework for generating dense, numeric vector representation of molecules utilizing the underlying structural information in the form of SMILES grammar productions. The grammar-based descriptors are fed to machine learning models as features (or regressors).

The steps involving the Grammar2vec framework are highlighted in Figure 2.3. Grammar2vec generates vector embeddings of molecules by considering grammar-representations of

molecules as sentences and the individual grammar rules as separate words. For instance, consider the compound 2-Naphthyl methylcarbamate with SMILES string CNC(=O)Oc1ccc2ccccc2c1. The 'molecular sentence' based on the grammar representation is generated by using the SMILES grammar representation as

'1' '75' '75' '75' ... '74' '64' '3' ... '3' '7' '66' '3' '6' '71' ... '39' '81'

where each production rule is equivalent to the 'molecular word'. Treating these representations as sentences and the individual constituent units as words, the Word2Vec model is trained using the gensim package in Python [30] and learn the vector representations of each of these molecular words. The vector representation for the entire molecule (molecular sentence) is obtained by averaging out the word embeddings learned across all the constituent words, a standard approach for learning sentence embeddings in natural language processing. A similar approach involving purely character-based SMILES strings (without grammar) is the idea behind SMILES2vec [22]. Similarly, [31] used Word2Vec model with bond-strings as input to generate molecular descriptors that were used for property estimation in a deep learning framework.

Since the performance of the Grammar2vec representation is compared against the SMILES strings-based dense representation, two separate Word2vec models are trained on a corpus of nearly $\sim 15,000$ molecules for $10,000$ iterations to acquire dense vector representations of size 32 from molecular sentences.

Remark 1: Note that the actual characters in the molecular sentences do not matter; instead their relative position and the context words are important while learning their word embeddings. Therefore, the individual tokens could be replaced to any arbitrary string as long as they map to the same word uniquely.

Remark 2: Just like the group contribution (GC)-based approaches, if a molecule is not present in the dataset, its vector representation could still be generated since the dense vectors are obtained using models trained on the individual bits in the molecular sentences

comprising SMILES grammar rules or the SMILES characters. Therefore, any new molecule could be represented as long as these individual bits that it is composed of are seen by the model during training (a fair assumption of fixed vocabulary size, as in natural language). In fact, in our test set, there were molecules that were never seen by the model during the training stage.

2.2.3 Support vector regression for property estimation

To estimate the thermodynamic properties of molecules, a kernel support vector regression (SVR) model [32] is used to build a regression model between the molecular descriptors and the correspond property values. The kernel SVR framework is chosen due to its ability to model complex, non-linear interactions between features even under small sample conditions and generalizability to out-of-sample datasets. The general idea behind the kernel SVR method is to first map the original data into a high dimensional space using kernels, and then find an optimal decision boundary or separating hyperplane by minimizing the error or the distance between the observed and the predicted values by formulating this as a constrained optimization problem.

For the simple linear SVR case (the non linear SVR described in subsequent paragraphs is used), the regression function $\mathbf{f}(.)$ in Equation 2.1 is assumed to be linear and is defined as,

$$\mathbf{f}(\mathbf{x_i}, \boldsymbol{\beta}) = \mathbf{x_i}^T \boldsymbol{\beta} + b \tag{2.2}$$

where b is an additional coefficient that corresponds to the bias in the predictions, $\mathbf{x_i}$ is a numeric vector representing a molecule, and $\boldsymbol{\beta}$ is a vector of unknowns that need to be estimated. The objective is to minimize the norm of the coefficients $\boldsymbol{\beta}$ such that the residuals are within a given limit ϵ. Slack variables are introduced for each point in the constraints equation to ensure the constraints are satisfied and a solution exists for the constrained optimization problem (known as soft-margin SVR). The primal formulation of the optimization

problem is thus given by,

$$\min_{\boldsymbol{\beta}} ||\boldsymbol{\beta}||^2 + C \sum_{i}^{N} (\zeta_i + \zeta_i^*) \tag{2.3}$$

subject to,

$$y_i - (\mathbf{x_i}^T \boldsymbol{\beta} + b) \leq \epsilon + \zeta_i, \quad (\mathbf{x_i}^T \boldsymbol{\beta} + b) - y_i \leq \epsilon + \zeta_i^*$$

where, C is a parameter that controls the penalty imposed on points that lie outside the ϵ margin, and ζ_i and ζ_i^* are non-negative slack variables that define the maximum tolerable error without imposing a penalty on the regression errors. The Lagrange dual formulation for this problem, that is computationally easier to solve, is given by,

$$\min_{\lambda} \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} (\lambda_j - \lambda_j^*)(\lambda_k - \lambda_k^*)(\mathbf{x_j}^T \mathbf{x_k}) + \epsilon(\lambda_j + \lambda_j^*) - \sum_{j=1}^{N} y_j (\lambda_j - \lambda_j^*) \tag{2.4}$$

subject to,

$$\sum_{j=1}^{N} (\lambda_j - \lambda_j^*) = 0, \quad 0 \leq \lambda_j \leq C, \quad 0 \leq \lambda_j^* \leq C$$

where λ_j and λ_k are non-negative Lagrange multipliers. The regressed (predicted) values, $\hat{y}_i$, for an input $\mathbf{x_i}$, is given by

$$\hat{y}_i = \sum_{j=1}^{N} (\lambda_j - \lambda_j^*)(\mathbf{x_j}^T \mathbf{x_i}) + b \tag{2.5}$$

Instead of using the linear assumption for the regression function $\mathbf{f}(.)$ in Equation 2.1, a kernel function is used that transforms the data into higher dimensions to learn a non-linear function in the original space. This is done by replacing the inner products or dot products in the linear SVR formulation above with kernels. The radial basis function (RBF) kernel given by,

$$K(x_j, x_k) = exp(-\gamma ||x_j - x_k||^2) \tag{2.6}$$

where the parameter γ controls the width of the kernel is used. The Kernel matrix is an $n \times n$ matrix where each element corresponds to the inner product of the transformed data points in higher dimensions. Replacing the inner products with the RBF kernel function, the dual problem thus becomes,

$$\min_{\lambda} \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} (\lambda_j - \lambda_j^*)(\lambda_k - \lambda_k^*) K(x_j, x_k) + \epsilon(\lambda_j + \lambda_j^*) - \sum_{j=1}^{N} y_j(\lambda_j - \lambda_j^*) \tag{2.7}$$

subject to,

$$\sum_{i=1}^{N} (\lambda_j - \lambda_j^*) = 0, \quad 0 \leq \lambda_j \leq C, \quad 0 \leq \lambda_j^* \leq C$$

The regressed (predicted) values, $\hat{y}_i$, for an input $\mathbf{x_i}$, is given by

$$\hat{y}_i = \sum_{j=1}^{N} (\lambda_j - \lambda_j^*) \; exp(-\gamma ||\mathbf{x}_j - \mathbf{x}_i||^2) + b \tag{2.8}$$

Therefore, Equations 2.7 and 2.8 characterize our trained non-parametric regression model, and the regressed values for thermodynamic properties of a give molecule could be estimated using Equation 2.8 for a given molecule represented as $\mathbf{x_i}$.

2.2.4 Interpretable ML using Shapley values

To improve the interpretability of the kernel SVR model used in the regression task, understanding the feature importances and their contribution towards the estimated values is of interest to us. Such analysis is necessitated due to the inherent complexity of the kernel SVR model, primarily due to the high-dimensional feature transformation using the radial basis function (RBF) kernel. This renders the straightforward evaluation of the support vectors for understanding the feature importance nearly impossible. Shapley values, a concept from cooperative game theory used to compute the contribution of each player to the final payout, is therefore used to understand the feature importances for the property estimation

task [33].

Shapley values are a measure of the average marginal contribution of a feature across all possible coalitions (or feature combinations). To quantify the importance of a given feature, different feature coalitions are simulated and the predicted value for the different contributions are averaged and subtracted from the predicted value with the given feature in the coalition. This computation is performed for all possible coalitions, and the Shapley value is the average of all the marginal contributions to all possible coalitions. Formally, the Shapley value for a feature j is defined as,

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} (\hat{f}(x_{+j}^m) - \hat{f}(x_{-j}^m)) \tag{2.9}$$

where $\hat{f}(x_{+j}^m)$ is the prediction for x with a random number of feature values replaced by feature values from a random data point z except for the respective value of feature j; x_{+j}^m is identical to x_{-j}^m except that the value x_j^m is taken from the random sample z in x_{-j}^m; features on the left of x_j have values from the original observations and those on the right of x_j take their values from a random instance; and M is the number of instances generated. This procedure is repeated M times for all the features and feature importances are computed.

These concepts have been applied in various areas including machine learning for understanding ML models [34, 35, 36, 37]. Following a similar approach, Shapley values were computed for our model and inferences derived from them were also used to prune the model by retaining only the important features contributing the most to the model performance as depicted in the algorithmic framework in Figure 2.1. The Shapley values were computed using the SHAP package in Python [33].

2.3 Dataset and model training

In this section, a detailed overview of the dataset and the model training aspects is provided. Specifically, a description of the dataset for the two properties of interest is initially

presented, followed by an explanation of the model training and hyperparameter tuning aspects for the kernel SVR models. Additionally, a visual comparison of the learned feature vectors for the dense vector embeddings of the molecular words obtained using Grammar2vec is included.

2.3.1 Dataset description

The dataset used in our work is based on data used in the recent work by Alshehri et al. [11] for predicting normal boiling point (T_b) and critical temperature (T_c). The dataset contains experimentally measured values of these properties with molecules represented as SMILES strings along with their groups-based representations indicating the various first-order, second-order, and third-order functional groups and the number of times they are present in each molecule. There are 200 first-order groups, 150 second-order groups, and 74 third-order groups, and hence, a total of 424 different functional groups. The data on normal boiling point for molecules consists of 3510 different pure compounds (molecules) whereas the data for critical temperature is much smaller with just 858 molecules. These molecules were further preprocessed to remove molecules that either could not be processed by RDKit or were not parsed by the SMILES grammar. The resulting final data had 3488 and 800 molecules for T_b and T_c, respectively.

Though the developed framework is applicable for predicting any thermodynamic property of a molecule as long as there is enough training data available, models were built only for predicting T_b and T_c because these two properties provide enough variety of challenges that are generally encountered – modeling difficulty, limited data availability, and performance on out-of-sample examples. Moreover, limiting ourselves to just two (but important) properties would help in performing a detailed analysis of the underlying models and the molecular representations trained to make them more interpretable.

2.3.2 Grammar2vec and regression model hyperparameters

As explained in Section 2.2.2, the Grammar2vec model was used to learn dense vector representations for molecules using SMILES grammar and Word2vec using SMILES strings. Vector embeddings of molecules with different sizes were trained, namely $8, 16, 32$, and 64 (resulting in the molecular representations $\mathbf{x}^{8\times1}, \mathbf{x}^{16\times1}, \mathbf{x}^{32\times1}$, and $\mathbf{x}^{64\times1}$, respectively), and evaluated the performance of learned representation of various sizes on a validation set in the property prediction task. The best model performance was achieved using an embedding of size 32, and hence, the size of the vector representations was fixed at 32 for performing final model training and detailed model analysis. These 32-dimensional dense feature vectors $\mathbf{x}^{32\times1}$ were used in the support vector regression (SVR) framework for training separate regression models for estimating T_b and T_c.

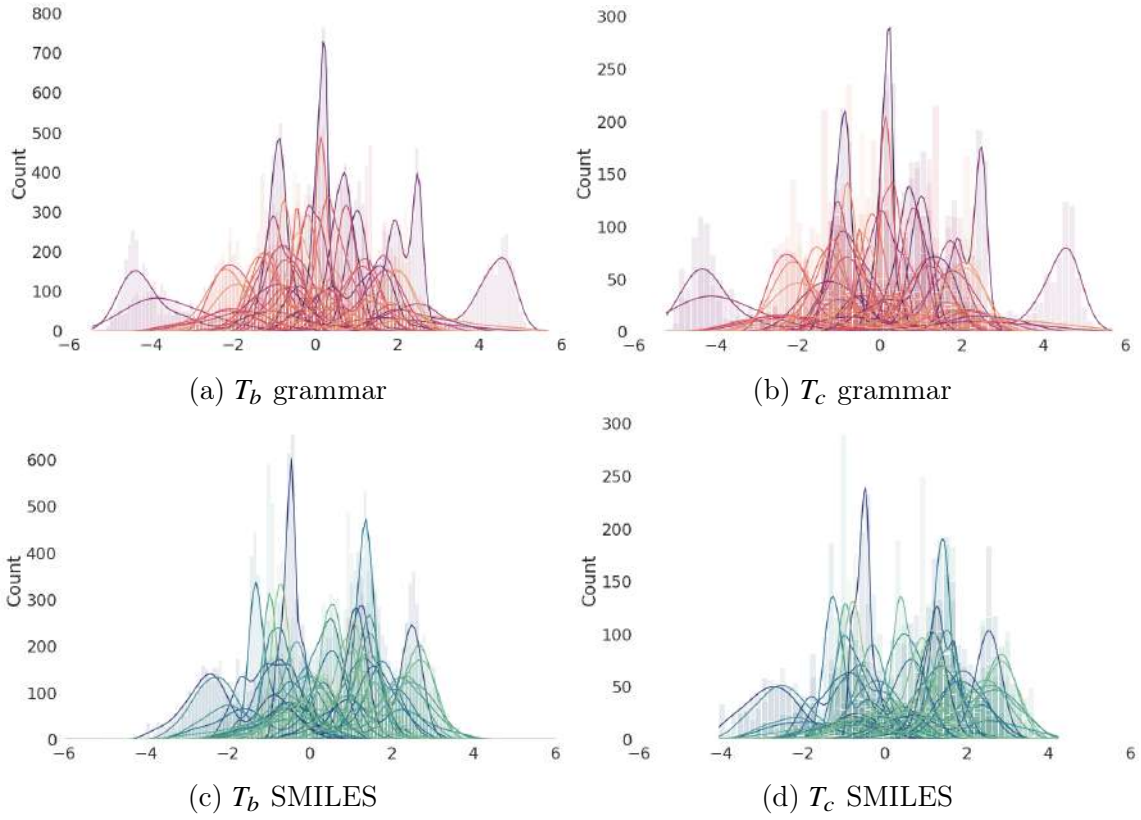
To perform modeling, the dataset was split into training and test sets using a 95/5 split where the training set is used for model building and the test set is used only for reporting model performance statistics. The training set is further subdivided to perform 5-fold cross-validation to search for optimal hyperparameter values using a randomized search in the following range: $C : 500 - 50000, \gamma : 0.01 - 0.2$, and $\epsilon : 0.1 - 1$. The hyperparameters are tuned separately for all the 4 models (using models trained on two separate representations each for T_b and T_c estimation). Therefore, each property prediction model is characterized by three hyperparameters that need to be tuned using the given dataset.

Remark 3: Since Grammar2vec is an unsupervised learning technique, the word embedding models were trained on the entire dataset containing nearly $\sim$ 15k molecules. One may choose to train this model on an even bigger dataset with millions of molecules such as the Zinc dataset or the USPTO reactions dataset.

2.3.3 Learned molecular representations

Since a machine learning model is being used for the regression task and using a 32-dimensional dense vector representation for molecules as input to the regression model, each

individual feature should capture different aspects of a molecule. In other words, each feature should ideally focus on distinct aspects of a molecule, and that should be unique to a given feature. While there must be some overlap between what each feature captures, an ideal, rich representation would minimize this overlap (or similarities) across features. The similarity or dissimilarity across features is qualitatively assessed by examining their distribution plots (histograms). The greater the disparity between the feature distributions, the more extensive the representation is expected to be. It is hypothesized that a rich representation would be one that encapsulates the most concentrated information within each feature.



(a) T_b grammar (b) T_c grammar

(c) T_b SMILES (d) T_c SMILES

Figure 2.4: Histograms for the 32 features based on the SMILES grammar and SMILES representations for molecules in the T_b and T_c datasets. The vertical axis corresponds to the frequency and the horizontal axis represents the actual numeric values of the feature vector elements.

The histograms for the dense vector-based representations for molecules learned using the Grammar2vec and Word2vec model (for SMILES) are shown in Figure 2.4. Histograms for 32-dimensional molecular representations obtained using SMILES grammar and SMILES are

plotted. There are 32 shades of colors in each sub-figure and each of them corresponds to a different feature.

2.4 Results

Detailed results obtained using our algorithmic framework are presented in this section. An information-theoretic analysis is first presented in Section 2.4.1 for comparing the various learned representations in an ML-independent manner that is rooted in fundamental analysis of uncertainty and conditional information gain associated with the representations. The detailed regression statistics, error analysis, and benchmarking the performance againts group contributions (GC)-based methods are presented in Section 2.4.2. The feature importance and subsequent model pruning results and analysis based on Shapley values are in Section 2.4.3 and Section 2.4.4, respectively.

2.4.1 Information theory

In representation learning, it is important to preserve as much information about the underlying entity as possible, and typically, the representations that are richer and have higher information perform well when used in machine learning frameworks [29]. Representations are important because they guide ML models in discovering the underlying, hidden patterns much more easily if a richer representation is used, as opposed to an obscure, difficult to interpret representation. We, therefore, turn to information theory since it offers a quantitative approach for quantifying the amount of information associated with a given communication channel [38] that could be adapted for chemical representations [29].

Two central concepts in information theory are the Shannon entropy and the conditional entropy (or information gain). Shannon entropy measures the amount of uncertainty associated with a given signal in terms of bits of information and by definition, a higher uncertainty translates to higher Shannon entropy or information capacity. Higher Shannon entropy points towards the higher information-carrying capacity of a communication channel. On the other

hand, conditional information gain or conditional entropy is a measure of the amount of uncertainty associated with a signal when a part of it is known (hence, conditional). Typically, a lower conditional information gain is desirable since it translates to a lower reconstruction error or loss. Figure 2.5 depicts these two concepts schematically along with the mathematical equations for their computation.

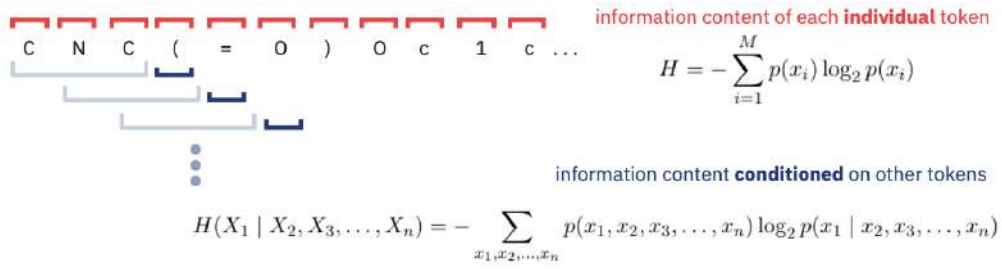


Figure 2.5: Shannon entropy and conditional entropy depicted schematically on the SMILES string representation for 2-Naphthyl methylcarbamate.

To utilize these concepts from information theory and analyze chemical representations from this perspective, individual tokens (or characters) in various representations are studied as as random variables, and therefore, the molecular representation becomes a sequence of random variables, $X_1, X_2, \ldots, X_n$, where n is the length of the representation for a given molecule and X_i could take any of the M possible tokens defined in the vocabulary of the representation. Tokens are the individual production rules for the SMILES grammar representation, or characters for the SMILES representation. For instance, consider the same compound 2-Naphthyl methylcarbamate. The different representations for this compound along with the associated random variables or tokens are as follows:

- SMILES (‘C’ ‘N’ ‘C’ ‘(’ ‘ = ’ ‘O’ ‘)’ ‘O’ ‘c’ ‘1’ ‘c’ ... ‘c’ ‘2’ ‘c’ ‘1’): $X_i^{SMILES} \in \{‘C’, ‘N’, ‘(’, ‘ = ’, ‘O’, ‘)’, ‘c’, ‘1’, ‘2’\}$, where $M = 9, n = 22$
- Grammar (‘1’ ‘75’ ‘75’ ‘75’ ... ‘3’ ‘7’ ‘66’ ‘3’ ‘6’ ‘71’ ... ‘39’ ‘81’): $X_i^{Grammar} \in \{‘1’, ‘3’, ‘4’, ‘6’, ‘7’, ‘8’, ‘16’, ‘39’, ‘40’, ‘58’, ‘62’, ‘64’, ‘65’, ‘66’, ‘69’, ‘71’, ‘73’, ‘74’, ‘75’, ‘81’\}$, where $M = 20, n = 77$

The Shannon entropy and conditional information gain are computed using the entire set of nearly 15k molecules in the dataset since this is an unsupervised approach and does not require the target property values. The required (conditional) probability distributions are estimated for the two representations (SMILES, and grammar) based on the co-occurrence matrices at different orders of conditioning (up to an order $\eta = 5$). An order $\eta = 1$ corresponds to the Shannon entropy, order $\eta = 0$ corresponds to Shannon entropy when the random variables follow a uniform distribution (theoretical limit of information capacity), and orders $\eta > 1$ correspond to conditional entropy when $\eta - 1$ preceding random variables are known and is computed using the conditional entropy formula in Figure 2.5.

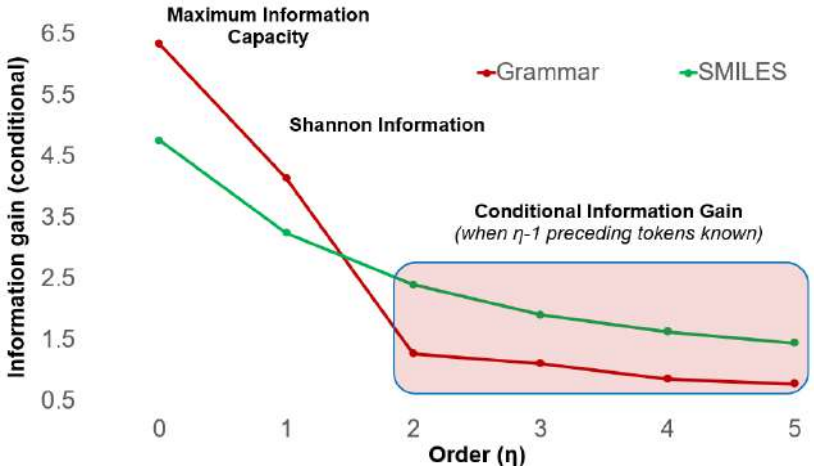


Figure 2.6: Information theoretic analysis results for the SMILES and grammar representations.

These results are presented in Figure 2.6 and it is observed that even though the theoretical and observed Shannon entropy is higher for the grammar representation, the conditional information gain drop significantly for the grammar representations for orders $\eta \geq 2$ and remains much lower than that of the SMILES representations even at increasingly higher orders. This points towards an important property of the grammar representations – when prior information (about preceding tokens) is known for the grammar representations, the amount of uncertainty associated with the other tokens for a molecular representation is reduced significantly which is not so much the case with SMILES representations. This could be

attributed to the hierarchical structure (as shown in Figure 2.2) and the underlying grammar that is much more efficient in eliminating the infeasible tokens than the other representations.

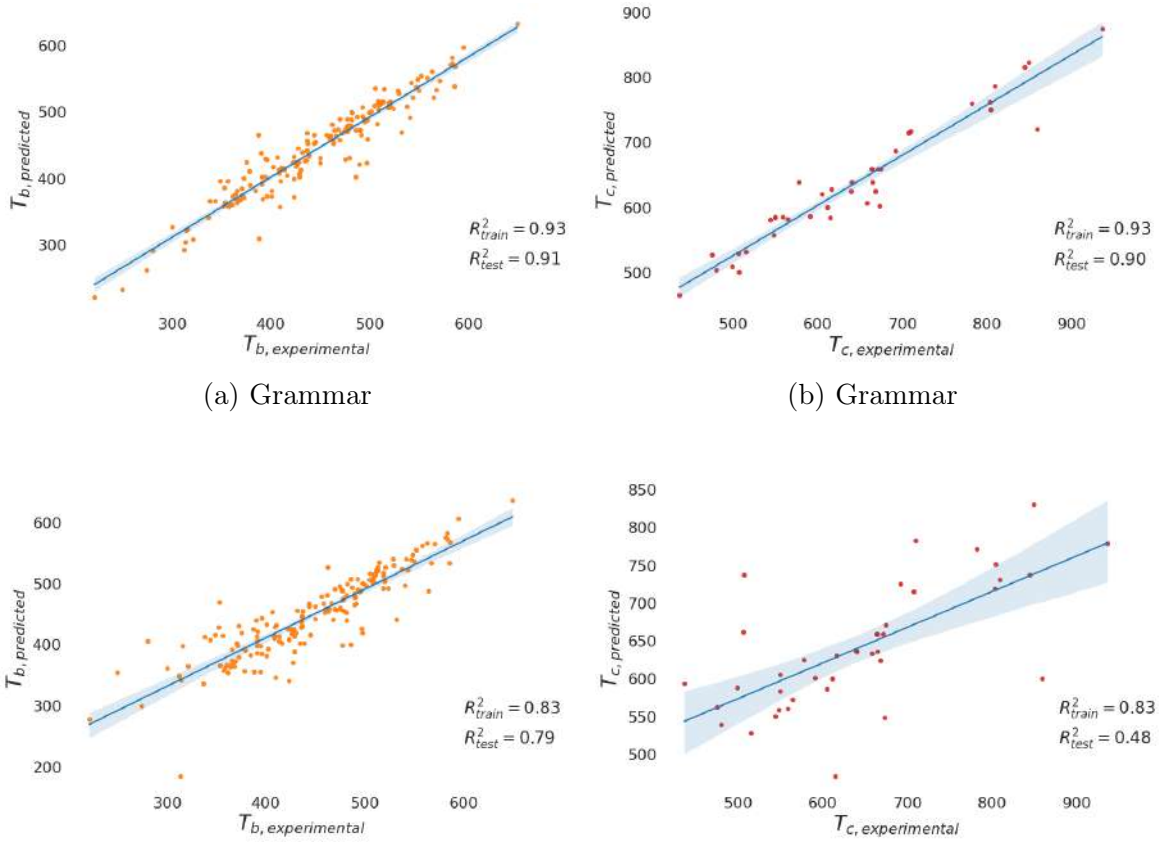
This property is extremely useful for methods like Word2vec (and consequently for Grammar2vec) because they attempt to encode a given word based on the contextual/neighboring words, and owing to the lower conditional information gain in the grammar representation, the contextual words are much more easier to infer. This property results in a lower reconstruction loss in the Word2vec model at the training stage and therefore, better representations. This richness and superiority of the grammar-based features should translate to better performance on the regression task.

2.4.2 Property model development and application

After comparing the representations and establishing their relative richness in an ML-independent manner, the results of the regression task for each of these representations are now presented based on the kernel-SVR models (utilizing Equation 2.7). Based on a random 95/5 split, the T_b dataset had 3313 data points in the training set and 175 data points in the test set. On the other hand, the T_c dataset had 760 data points in the training set and 40 data points in the test set. The scatter plots between the true values and the experimental values for normal boiling points (T_b) for molecules in the test set (unseen at the training stage) are presented in Figure 2.7. The R^2 values indicating the goodness of the predictions on the training and the test set are indicated in the inset of each sub-figure.

It is observed that the grammar representations-based models have the highest R^2 values on the test set (comprising molecules unseen during the training stage), pointing towards their better generalization capabilities. In addition, their predictive accuracy is higher even when the number of training samples is reduced by nearly an order of 4 (3488 for T_b vs 800 for T_c). The SMILES representations-based model is seen to perform poorly under small-sample conditions. The groups representations-based models though have a near-perfect accuracy on the training set, the performance on the test set in both the cases is significantly lower,

and therefore, the models overfit the training set.



(a) Grammar (b) Grammar

(c) SMILES (d) SMILES

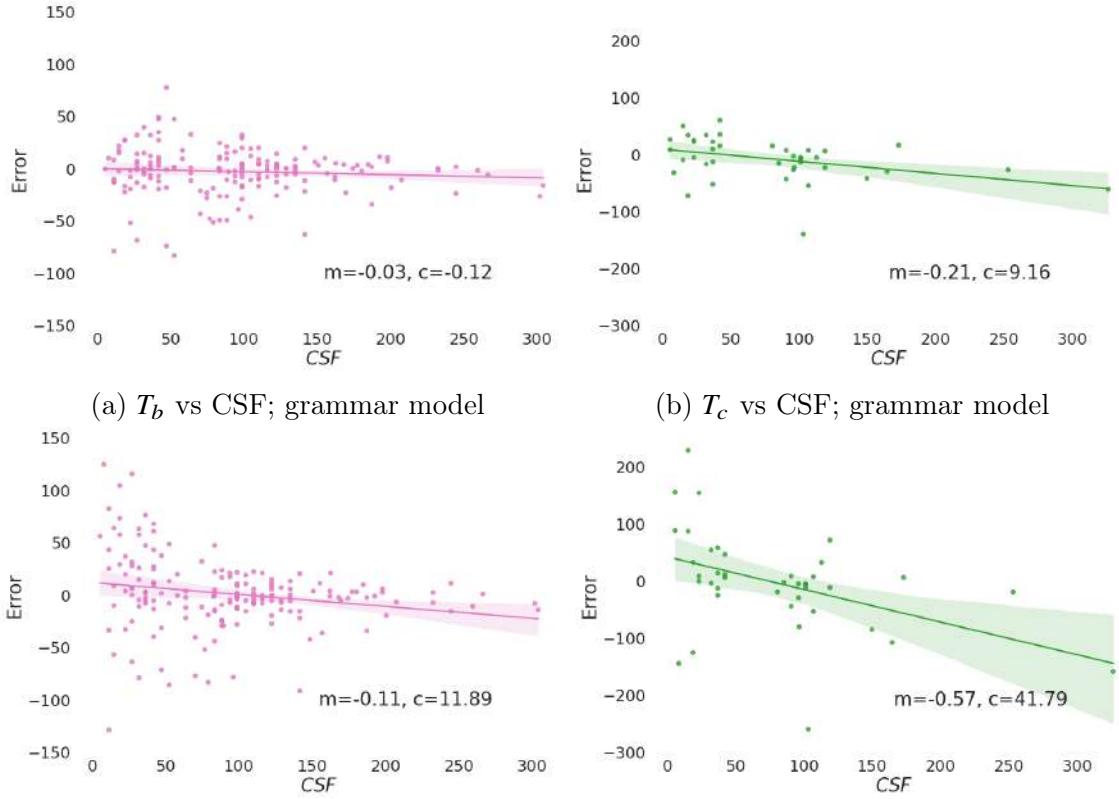
Figure 2.7: Regression results on the normal boiling point (T_b) and critical temperature (T_c) prediction tasks on the test set containing molecules unseen at the model training stage.

To gain a better understanding of the model performance for various molecules, the prediction errors ($\hat{y} - y$) are studied as a function of the complexity of the molecule. The complexity of molecules inspired by the chemical scoring functions (CSF) [39] for scoring the chemical and synthetic complexity of a reaction pathways is defined as,

$$CSF = \text{SMILES\_LEN}^{3/2} + \alpha\text{RINGS} + \beta\text{STEREO} \tag{2.10}$$

where SMILES_LEN is the length of the SMILES string of the molecule which is related to its mass and overall complexity, RINGS is the total number of rings in the molecule and STEREO is the number of stereocenters in the molecule. The parameters α and β could be

tuned but for simplicity, their values are fixed to the average value for the SMILES_LEN$^{3/2}$ across all molecules to ensure the three terms contribute roughly equally to the CSF score. The plot of the CSF and the prediction errors for T_b and T_c using different representations for molecules in the test set are presented in Figure 2.8.



(a) T_b vs CSF; grammar model

(b) T_c vs CSF; grammar model

(c) T_b vs CSF; SMILES model

(d) T_c vs CSF; SMILES model

Figure 2.8: Prediction error vs molecular complexity analysis using chemical scoring functions (CSF) for T_b and T_c on the test set containing molecules unseen at the model training stage.

Based on these results, the following inferences are drawn:

- For the grammar representations, both for T_b and T_c, the relationship between the prediction error and CSF is either non-existent or very weak, pointing towards the complexity-agnostic nature and hence, wider generalizability/applicability of the fitted model. This is established by the slope and the intercept of the regression line along with their confidence bounds indicated in Figure 2.8(a) and (b) for T_b and T_c, respectively.

- The models fitted using the SMILES representations seem to underpredict the proper-

ties (negative $\hat{y}-y$) for complex molecules. This points towards their inability to capture the complex interactions between various functional groups in complex molecules and erroneously treats them as relatively simpler molecules. The grammar-based models do not suffer from this bias, possibly due to their richer features.

- Across both the models, the absolute values of the prediction errors are relatively high for smaller or simpler molecules. However, even in this case, the errors for the grammar-based models are much smaller than the errors in the other model.

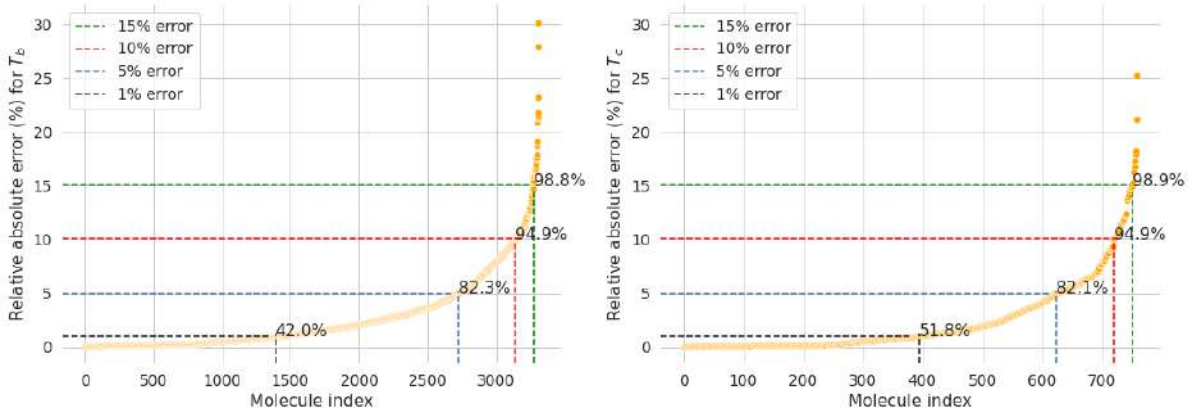
It is hypothesized that the nearly non-existent correlation between the prediction errors and the molecular complexity for the SMILES grammar model could be attributed to the higher structural information in-built in these representations. The sources of error in a machine learning models could be attributed to two factors – first, the structure-based errors and model inadequacies, and second, the errors in the parameter estimates. The latter is often due to limitations of the model in mapping the transformations between the features and the target variable including sub-optimal parameter estimates. The former, however, depends on the ability of the representation (features) in capturing the molecular chemistry. The grammar representations, therefore, do not seem to suffer from the structure-based limitations but only the model-based errors.

In order to asses the performance of our model better, the percentage deviation of the predicted values from the measured property values is computed based on their relative absolute percentage errors, for both T_b and T_c, defined as

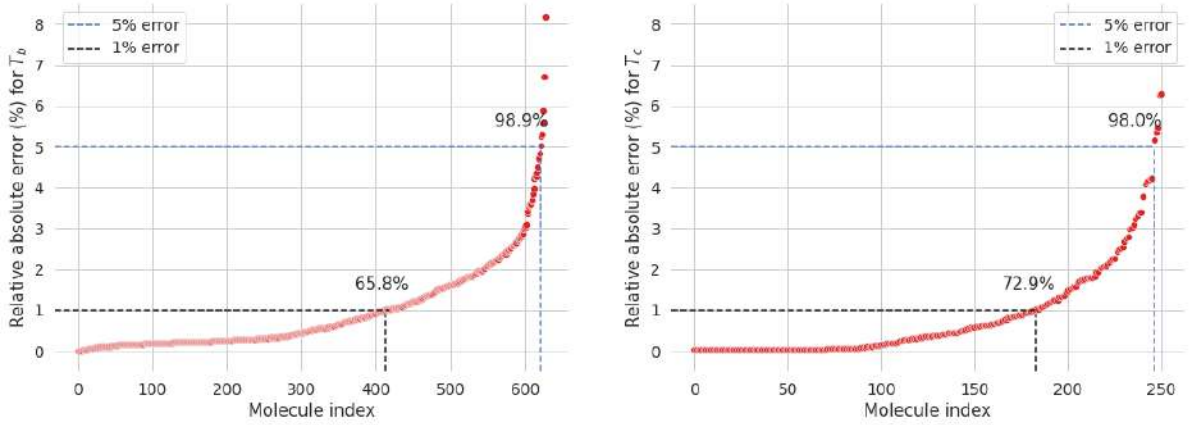
$$RAE\ (\%) = 100 \times (y_{pred} - y_{true})/y_{true} \tag{2.11}$$

In addition, the same exercise is also performed end-to-end (representation learning, model training, RAE computation) for a subset of the entire dataset containing only hydrocarbon molecules. This dataset contained 1134 molecules that were used for learning molecular representations, 628 and 34 data points respectively for training and testing the $T_{b,hydrocarbons}$

estimation model, and 251 and 14 data points respectively for training and testing the $T_{c,hydrocarbons}$ estimation model. The error distribution plots are presented for our models in Figure 2.9 and compare the results against those presented in Alshehri et al. [11] on various metrics in Table 2.2.



(a) Error distribution for the entire dataset



(b) Error distribution only for the hydrocarbons dataset

Figure 2.9: The relative absolute percentage error plots obtained using the grammar representation model for T_b and T_c on the entire dataset (training set + test set).

Table 2.2: Comparison with a previous work on property estimation that worked with the same dataset. The numbers indicate the percentage of molecules below the given percentage relative error threshold except the last column that provides values for the maximum percentage relative error observed in the predictions.

		1% error	5% error	10% error	15% error	Max. error
Alshehri et al. [11]	T_b	**79.1**	**87.8**	92.7	-	>50%
	T_c	**84.4**	**91.4**	94.2	-	>40%
Our model	T_b	42.0	82.3	**94.9**	**98.8**	**30.2%**
	T_c	51.8	82.1	**94.9**	**98.9**	**25.3%**
	$T_{b,hydrocarbons}$	65.8	**98.9**	**100**	**100**	**8.2%**
	$T_{c,hydrocarbons}$	72.9	**98.0**	**100**	**100**	**6.3%**

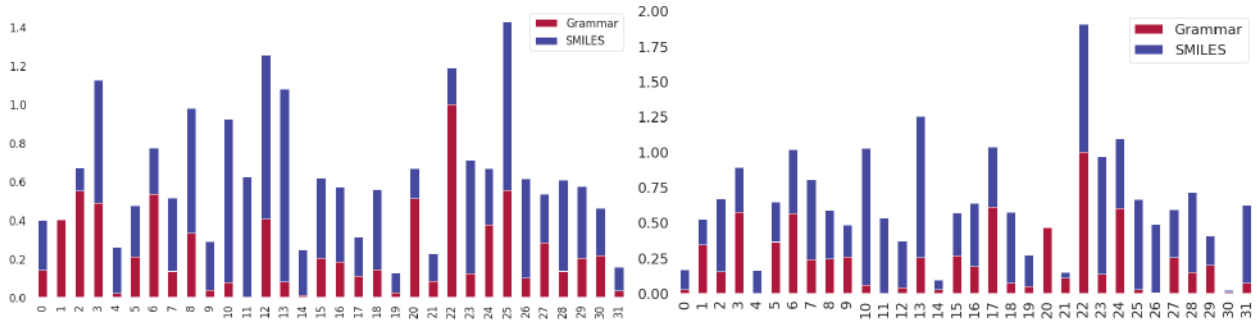
It is observed that even though our model predicts relatively fewer number of molecules within the 1% error threshold, the model quickly catches-up and outperforms the model by Alshehri et al. for an error threshold of 10% and above, with ~ 99% of all the predictions falling within a 15% error threshold. Moreover, the maximum prediction error for T_b and T_c at ~ 30% and ~ 25%, respectively, is much lower than the maximum error reported by Alshehri et al. for their best performing model. The performance on the hydrocarbons dataset is significantly better with nearly $98 - 99\%$ of all the molecules within an error-threshold of less than 5%, both for T_b and T_c estimation. The gradually increasing trend in the relative errors (as opposed to a steep increase beyond a certain number of molecules), highlights a smooth functional mapping from the features to the property values learned by the ML-based regression model. As a consequence of this, the model has better generalizability. Therefore, it is inferred that our model is robust towards encountering new molecules or the presence of outliers in the dataset. The latter is a major issue because new molecules are being synthesized at a much faster pace and the experimentally measured properties dataset often have errors and outliers.

2.4.3 Feature importance and model interpretability

To further bolster the interpretability of our model, an analysis using Shapley values is performed to understand the contribution of the individual features towards the model

predictions. The Shapley values were computed using a weighted k-means approach using 10 samples weighted by the number of points they represent. This analysis is performed the two models each for T_b and T_c properties, separately.

The feature contribution charts for the T_b and T_c predictions are presented as stacked bar plots in Figures 2.10 (a) and (b), respectively. In order to allow for an easy comparison on a relative scale, the feature contributions are scaled in the range 0 to 1 by performing a min-max scaling for each model separately. It is observed that the most important features (out of the 32 features) for the grammar model are T_b : $\{22, 25, 2, 6, 20, 3, 1, 12, 24, 8, 27, 5, 30, 15, 29, 16, 18, 28, 0, 7\}$, T_c : $\{22, 24, 17, 3, 6, 20, 5, 1, 15, 9, 7, 27, 13, 8, 29, 16, 2, 28, 23, 21\}$ and for the SMILES models are T_b : $\{13, 25, 12, 10, 3, 11, 8, 23, 26, 28, 18, 15, 7, 16, 29, 24, 0, 5, 27, 30\}$, T_c : $\{13, 10, 22, 23, 28, 25, 31, 24, 7, 2, 26, 11, 18, 17, 16, 6, 27, 3, 12, 15\}$.



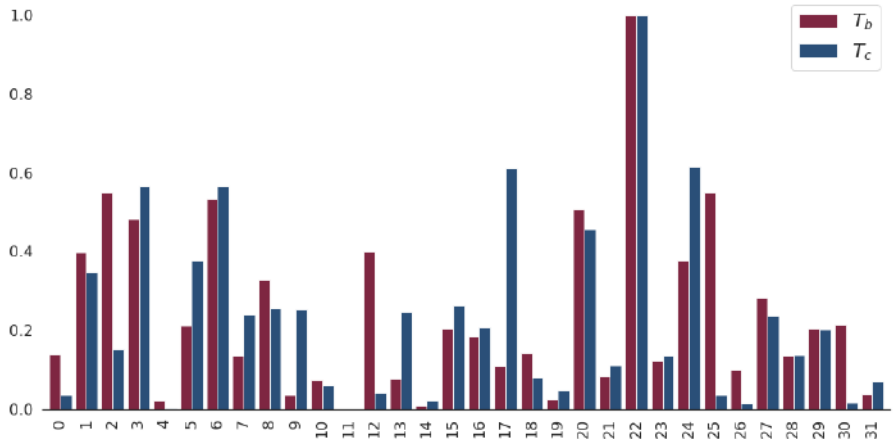
(a) T_b (b) T_c

Figure 2.10: Feature importance charts (scaled) for the T_b and T_c predictions for all the three representations. The feature importance are relative and indicate the contribution of each feature towards predicting the properties of interest. It is seen that not all features are equally important and the unimportant ones could be dropped without much impact on the model performance.

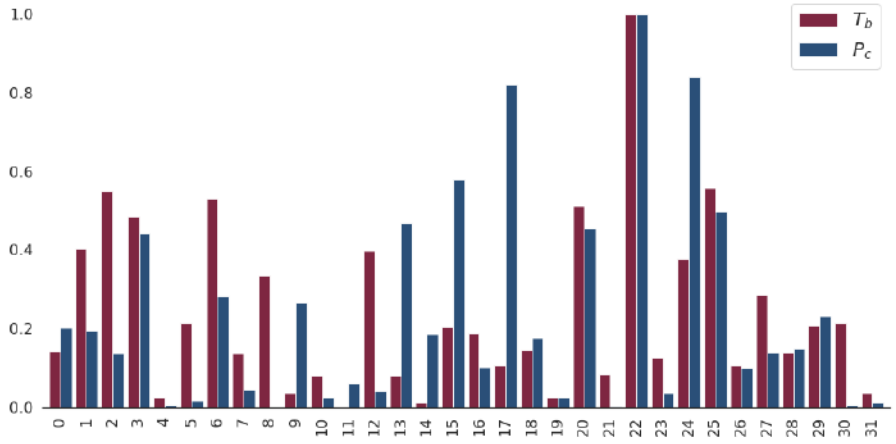
An interesting observation based on the above model is that the feature importance for a given molecular representation (grammar or SMILES) is very similar for both T_b and T_c predictions. For instance, of the 20 most important features for the grammar-based representations for T_b and T_c prediction, 15 out of 20 features are common. For the SMILES-based representations too 15 out of 20 features are common. This points towards a possible corre-

lation between the features and the underlying molecular chemistry (such as intermolecular forces and interactions between various molecular groups). This implies that the model has discovered an overlap between a majority of the features responsible for the underlying chemistry behind the two properties overlap. In fact, this observation is rooted in chemistry since it is known that the normal boiling point (T_b) and critical temperature (T_c) of a given molecule are correlated. A side-by-side plot of the relative feature contributions for T_b and T_c models for grammar-based representations is shown in Figure 2.11 (a).

In order to ensure this chemistry-correlation is indeed captured by the developed model and it is not just an artifact of the example considered, an additional grammar representation-based model is trained following the same approach for predicting critical pressure (P_c) which has nearly the same number of training data points a T_c. A side-by-side plot of the relative feature contributions for T_b and P_c is shown in Figure 2.11 (b). As expected from chemistry, it is seen that the correlation between feature importance is much weaker between these properties when compared to the feature correlation between T_b and T_c. Thus, it is conluded that the grammar-based features preserve the underlying chemistry correlations between molecules and consequently in the model analysis. However, the grammar2vec approach in its current form could only be used to infer chemistry correlations indirectly which could then be used to gain a deeper understanding of the system as shown using Shapley values. Developing a more detailed analysis framework to facilitate a deep understanding of the important features from a chemistry perspective is one of our future directions of research.



(a) T_b vs T_c



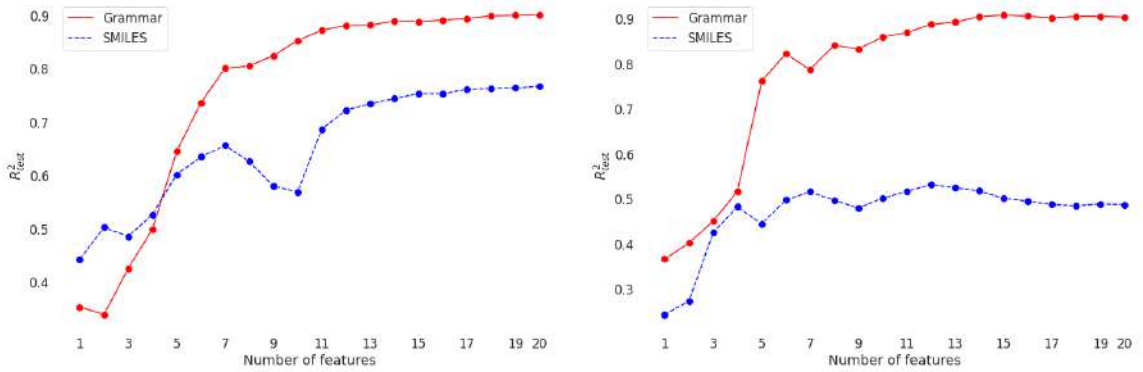
(b) T_b vs P_c

Figure 2.11: Comparison of feature importance (contributions) for regression models for T_b, T_c, and P_c. Based on the underlying chemistry-based correlations, it is expected that T_b and T_c would have similar feature importance, whereas T_b and P_c would have relatively higher differences in feature importance. This behavior is observed in the above comparison plots between T_b vs T_c and T_b vs P_c.

2.4.4 Model pruning

Having obtained information on the importance of various features for estimating molecular properties of interest, this information is leveraged to further simplify our model by pruning the features. To do this, for each representation, models are built starting with a model with just the top-most important feature and compute the R^2_{test} value, then build another model with this feature and the next most important feature and again compute

the R^2_{test} value, and do this sequentially until a sufficiently good performance is observed for the mode. The plot of the R^2_{test} values for sequentially developed models (from 1 to top-20 features) for both the representation vs the number of features is presented in Figures 2.12 (a) and (b) for T_b and T_c model.



(a) T_b (b) T_c

Figure 2.12: The R^2_{test} as a function of the number of features for the different models.

Table 2.3: Comparison of model performance (test-R^2) with different number of features.

	# Features	Grammar	SMILES
T_b	32	0.91	0.79
	20	0.90	0.77
T_c	32	0.90	0.48
	20	0.90	0.49

It is observed that both the models require just 20 features or fewer to achieve nearly the same level of performance as obtained by using all the 32 features. The comparison of the performance on the test set using 32 and 20 features for T_b and T_c predictions are presented in Table 2.3. Therefore, the final pruned model is much simpler and is characterized by just 20 features that are dense, rich, and capture the molecular structural information efficiently. Moreover, it is observed from the trend in Figure 2.12 that in both the cases, the grammar-based models achieve a significantly higher R^2 by using fewer features to attain the same accuracy both for T_b and T_c predictions. For example, the grammar model achieves an

R^2 of nearly 0.8 for both T_b and T_c using just 7 features whereas it requires at least 20 features for the SMILES-based models to achieve a similar R^2. These again point toward the richness of the grammar features and its better generalization, as demonstrated earlier in the section information-theoretic analysis and complexity-agnostic nature of the model shown through the error vs. CSF analysis. In addition, fewer features would translate to better computational efficiency and a smoother decision boundary – both of which result in better generalizability of the model on unseen data points. In other words, the model predictions would have smaller variance and hence lower possibility of overfitting (or memorizing) the training data.

2.5 Conclusions

Thermodynamic property prediction is an important problem that requires a systematic approach to ensure the data-driven models are rooted properly in fundamental principles of physics and chemistry. Engineering the molecular representations that are input to such models in a way that maximizes the amount of physics/chemistry captured is one possible approach. Here, the SMILES grammar-based, dense vector representations obtained using the Grammar2vec framework are identified to be one such promising possibility. These representations were obtained by invoking the natural language analogy to generate vector representations for 'molecular sentences' constructed by combining the SMILES strings with their underlying SMILES grammar production rules.

The results demonstrated the superiority of the Grammar2vec molecular representations. They were shown to be associated with richer features that capture different chemistry, lower conditional uncertainty, better regression statistics both for data-rich (T_b) and data-limited (T_c) scenarios, higher efficiency in capturing molecular complexity indicated by the error analysis, and relatively simpler models with fewer features when compared to models using other representations. It was established that data alone is not sufficient to make better predictions – as is often assumed in the current era of deep learning – at least in chemistry.

Instead, data along with richer representations that capture the intricacies of the underlying entities is of equal importance. The Grammar2vec framework could be used for estimating several other thermodynamic properties (in addition to T_b, T_c) using relatively simpler and more interpretable machine learning models.

It is envisioned that representations rooted in chemistry would hold significant value, not only for the thermodynamic property prediction task but also for other chemistry problems that demand data-driven modeling. Future extensions of this work involve applications such as incorporating additional molecular structural descriptors, employing the grammar2vec framework for performing comprehensive property estimation on other thermodynamic properties with larger datasets, and carrying out molecule design and optimization in conjunction with retrosynthesis planning.

Chapter 3: Forward and Retrosynthesis Reaction Prediction

Reaction prediction, both in the forward direction (from reactants to products) as well as in the retrosynthesis direction (from product to reactants), is a central problem in computational chemistry. Traditional reaction prediction and retrosynthesis planning relied heavily on the expertise of chemists, which is both time-consuming and resource-intensive. In contrast, data-driven methods offer automated strategies for predicting reaction outcomes fairly accurately, enabling the design of high-throughput systems, computer-aided retrosynthesis planning, and other such computational chemistry-focused applications. The immense interest in this problem over the recent years could be attributed to its practical applications across areas such as drug discovery, synthesis of novel organic compounds, and improvements in the reactions pathways from a commercial, social, or economic viability standpoint. The recent advances in artificial intelligence (AI), particularly the emergence of language models, have sparked further interest in the field.

The computer-aided reaction prediction problem has two distinct aspects: forward prediction, which involves predicting the reaction product from a set of reactant molecules, and retrosynthesis (or inverse reaction prediction), which involves identifying the necessary precursors for synthesizing a given target molecule. While both problems require understanding complex molecular interactions to correctly identify target molecule(s), retrosynthesis is notably more difficult due to its combinatorial nature. Multiple synthesis pathways are often feasible at each step, and the entire pathway sequence needs to be predicted correctly. However, the challenge of integrating explicit chemistry information with machine learning techniques remains unaddressed for both forward and retrosynthesis reaction prediction frameworks. It has been argued that the development of hybrid approaches that combine data-driven techniques with chemistry knowledge is required for more robust and practical

reaction prediction models [5, 40].

Our focus is the single-step reaction prediction and retrosynthesis prediction, which approximates multiple pathways as a 'single-step' leading from reactant to products, or vice versa. Single-step models can be broadly classified into two categories: template-based and template-free. Template-based models rely on predefined reaction templates to categorize and predict chemical pathways. These templates are derived from data-driven approaches or expert knowledge. One of the pioneering template-based retrosynthesis models, LHASA [41], incorporated chemical templates through a combination of reaction logic and heuristics. A more recent template-based approach, Synthia (formerly known as Chematica) [39], utilizes a decision tree to select a reaction template from over $70,000$ expert-created rules. Other template-based approaches which utilize various reaction heuristics and similarity scores are presented in [42, 43, 44].

On the other hand, recent advancements in computational power and machine learning have spurred a focus on template-free models. Template-free models use purely data-driven approaches to plan and evaluate reaction pathways and address some limitations of template-based methods such as lack of generality of prediction and dependence on template-quality. These methods typically represent molecules as strings in simplified molecular-input line-entry system (SMILES) format and model the reaction prediction problem as a sequence-to-sequence (seq2seq) problem. One of the earliest such works is by Liu et al. [45]. Recently, owing to the success of the state-of-the-art transformer architecture [46], a molecular transformer model was proposed [47] for forward prediction with significant performance improvement over other approaches [48, 49, 50]. Similarly, for the retrosynthesis problem, transformer-based approaches have been shown to be promising [51, 52, 53, 54]. The reader is referred to [5] for a detailed presentation and comparison of various template-based, template-free, and hybrid approaches for computer-aided reaction prediction and chemical synthesis.

While the vast majority of sequence-to-sequence models use the SMILES representation to represent input and target molecules, there are several limitations with using purely text-

based SMILES representations in sequence modeling frameworks. First, molecules are treated as merely character-based strings, disregarding the additional structural aspects. Second, the SMILES representation does not provide *explicit* structural information on the nature of chemical bonds, chains, atoms, and so on. Third, the underlying rules characterizing a SMILES string are disregarded and it is left upon the model to identify these patterns to ensure syntactically correct SMILES strings during prediction.

For the forward prediction problem, these issues are addressed by building a reaction prediction framework using a text-based representations generated from the underlying SMILES grammar (instead of just SMILES). Using SMILES grammar-based representations are akin to providing more syntactic as well as semantic information about a molecule's SMILES representation, resulting in chemistry-rich representations that explicitly contain additional structural information. It is shown that the SMILES grammar-based text representation leads to improved accuracy, fewer model parameters, and lower conditional entropy from an information-theoretic standpoint. Our proposed SMILES grammar-based molecular representation is a new and promising alternative to the SMILES representation where the molecules are initially represented as hierarchical tree.

For the retrosynthesis reaction prediction problem, the vanilla transformer architecture is adapted to handle SMILES grammar trees directly, formulate the retrosynthesis prediction problem as a tree-to-sequence problem, and perform tree convolution operations inspired by the group-contribution theory for property prediction. The proposed architecture explicitly incorporates the hierarchical structure of the molecular grammar tree. The resulting models achieve nearly state-of-the-art performance on a variety of metrics using relatively simpler model architectures that are more chemistry-aware.

3.1 Problem statement and objectives

3.1.1 Forward prediction

Given a set of reactants and the agents facilitating the chemical reaction, our objective is to predict the most likely major product of the reaction. This is formulated as a machine translation problem where the input sequences comprising the reactants and agents correspond to the source sentence and the output comprising the major product of the reaction corresponds to the target sentence (from a different language). The sentence analogues in this translation task are the set of SMILES strings whereas the characters in each SMILES string are their word analogues as in a natural language sentence. This analogy between chemical reactions transformation and natural language translation is exploited for predicting reaction outcomes.

The SMILES strings, however, are comprised of arbitrary characters that do not provide chemical or structural information crucial for modeling reaction chemistry systems. A SMILES grammar, analogous to context-free grammars in natural language [55], is therefore used to incorporate structural information for each molecule in our reaction prediction framework in a hierarchical manner. The sequence of grammar rules corresponding to each SMILES string therefore becomes their representation in this framework as shown in Figure 3.1. Such SMILES grammar-based representation are an important contribution of our work due to advantages such as explicit incorporation of chemical structure, reduction of strain on the model by letting it discover the transformations in a chemical reaction directly without a need to model the relationships between arbitrary characters. This overcomes overfitting in neural machine translation models often characterized by a large number of parameters, and increases the likelihood of predicting molecules with valid SMILES representations.

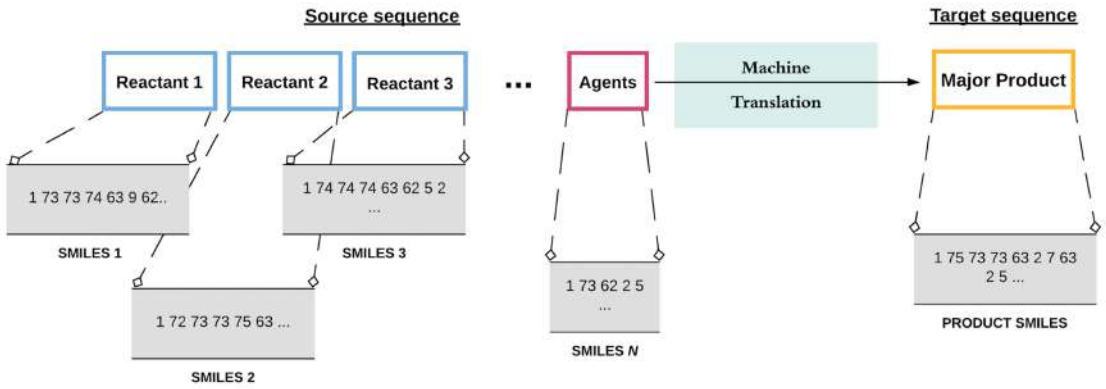


Figure 3.1: Modeling forward chemical reaction prediction task as a machine translation or sequence to sequence modeling problem with molecules represented using SMILES grammar-based representations.

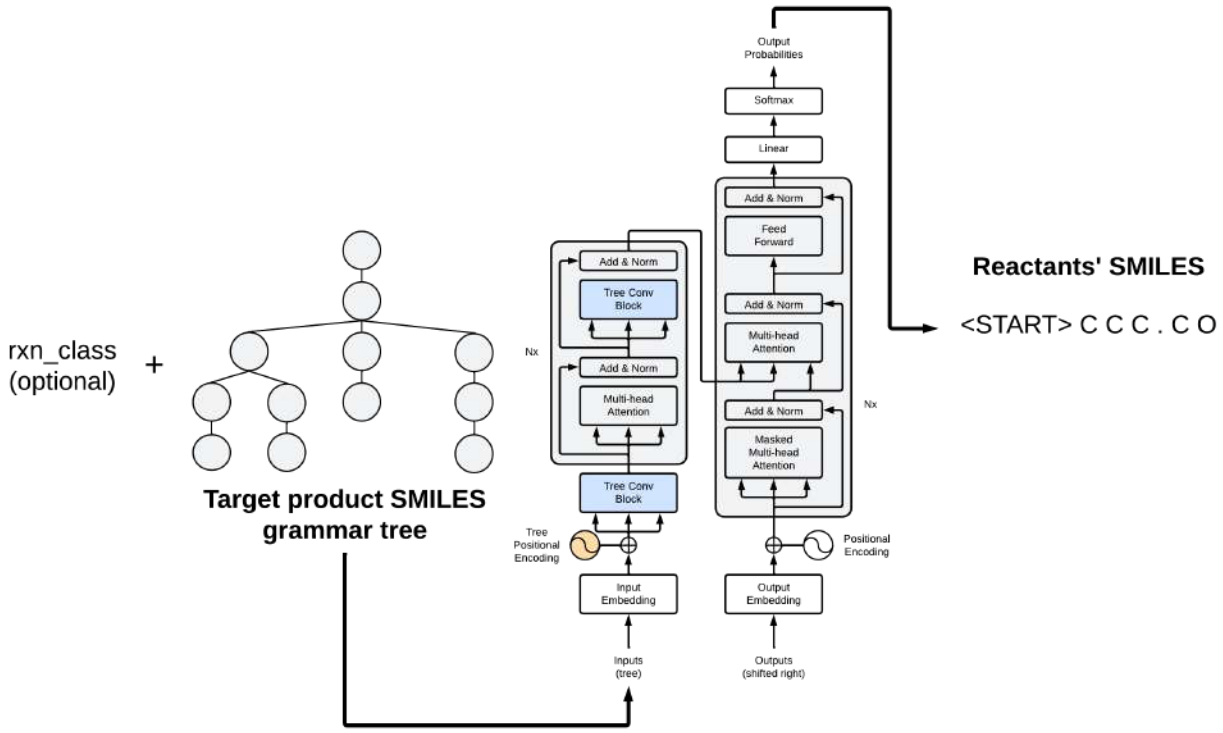


Figure 3.2: Modeling retrosynthesis chemical reaction prediction as a tree to sequence modeling problem with SMILES grammar tree of the target molecule as input to predict SMILES strings of the set of precursors.

3.1.2 Retrosynthesis prediction

A novel approach to the retrosynthesis prediction problem is presented by formulating it as a tree-to-sequence modeling task. Our input molecule(s) are represented as SMILES grammar-based trees, while the target molecule is represented as a canonical SMILES string. This representation choice naturally gives rise to a tree-to-sequence modeling problem, rather than the commonly used sequence-to-sequence framework. To the best of our knowledge, this is the first attempt to develop a tree-to-sequence approach for both retrosynthesis and forward reaction prediction.

The retrosynthesis prediction problem has two scenarios: known and unknown reaction class. In the known reaction class case, a class identifier is appended to the beginning of the target molecule, indicating the reaction class type. Conversely, for the unknown reaction class scenario, only the target molecule is provided as input. Since there may be multiple reactants in the predicted pathway, a special delimiter character `"."` is used to separate precursors. Additionally, a special `<START>` token to begin the translation process is used. The problem formulation diagram is illustrated in Figure 3.2.

3.2 Methods

In this section, the background methods for the sequence-to-sequence approach for forward prediction and tree-to-sequence approach for retrosynthesis are presented. This includes an overview of the SMILES grammar-based molecular tree representations, the vanilla transformer for the forward prediction model, the modifications made to the original transformer architecture to include tree positional encodings and convolutional blocks, and the beam search decoding procedure used to predict the most likely target sequences for a given input sequence.

3.2.1 Formal grammars and grammar for SMILES

Formal grammars have been the backbone of various language modeling tasks such as semantic interpretation of natural language, dialogue understanding, and machine translation. They are largely based on the idea that group of words belong to the same constituent units and that different constituents could be hierarchically grouped together to convey the given meaning [56].

Context-free grammar (CFG)

The most widely used formal grammar is the Context-Free Grammar (CFG) and was formalized in [55]. A context-free grammar consists of a set of productions (or rules) that express the way in which different words or symbols, comprising the lexicon in the language, can be grouped and ordered together. The symbols used in a CFG are grouped into two classes – symbols that correspond to the actual words with meaning in the language, called terminals, and the symbols that represent abstraction over a group of words and are used to represent a class of words or phrases in the language (terminals), called non-terminals.

Formally, a context-free grammar G is represented by four parameters – N, Σ, R, S where

- N: a set of non-terminal symbols
- Σ: a set of terminal symbols
- R : a set of production or rules of the form A $\longrightarrow \beta$, where A is non-terminal and β is a string of symbols from the set $(\Sigma \bigcup N)_*$
- S: a designated start symbol and a member of N

Typical English grammar rules comprise sentence level constructions (S $\longrightarrow$ NP VP, S $\longrightarrow$ VP), the noun phrase (Det $\longrightarrow$ NP), the verb phrase (VP $\longrightarrow$ Verb, VP $\longrightarrow$ Verb NP) and so on, where S, NP, VP, Det, and Verb are the sentence symbol, noun phrase, verb phrase, determiner, and verb, respectively.

A CFG can be thought of as a generator that could be used to generate sentences in a language by sequential application of productions, or as a tool for assigning structure to a given sentence [56]. In our work, the latter aspect of CFGs is primarily focused on, and they are used to incorporate structural information from a SMILES string.

Grammar for SMILES

Analogous to the context-free grammar for the English language, there exists a formal grammar for the string-based molecular representations used in chemistry such as the most commonly used representation – SMILES [14]. As described in the foregoing section, the set of productions (or rules), non-terminals, terminals, and a designated start symbol are the essential components of a context-free grammar. These components for the SMILES representations are presented in [1], that could be applied sequentially to generate the grammar-based parse trees representing the constituency of various components in a given SMILES string.

For instance, consider the simplified grammar in Table 3.1. Analogous to the notation for context-free grammar introduced in the section Context-free grammar (CFG), the following are their equivalents in this grammar.

- N: { `SMILES, CHAIN, BRANCHED_ATOM, BOND, ATOM, RINGBOND, BB, RB, BRANCH, AROMATIC_ORGANIC, ALIPHATIC_ORGANIC, DIGIT` }
- Σ: { `(, ), =, c, C, O, 1, 2` }
- R: productions (rules) 1 through 20 in Table 3.1
- S: `SMILES`

[1] http://opensmiles.org/spec/open-smiles-2-grammar.html

Table 3.1: Representative subset of the SMILES grammar

S.No	Production rules
1	SMILES ⟶ CHAIN
2	CHAIN ⟶ CHAIN BRANCHED_ATOM
3	CHAIN ⟶ CHAIN BOND BRANCHED_ATOM
4	CHAIN ⟶ BRANCHED_ATOM
5	BRANCHED_ATOM ⟶ ATOM RINGBOND
6	BRANCHED_ATOM ⟶ ATOM
7	BRANCHED_ATOM ⟶ ATOM BB
8	BRANCHED_ATOM ⟶ ATOM RB
9	BB ⟶ BRANCH
10	RB ⟶ RINGBOND
11	BRANCH ⟶ (CHAIN)
12	RINGBOND ⟶ DIGIT
13	BOND ⟶ =
14	ATOM ⟶ AROMATIC_ORGANIC
15	ATOM ⟶ ALIPHATIC_ORGANIC
16	AROMATIC_ORGANIC ⟶ c
17	ALIPHATIC_ORGANIC ⟶ C
18	ALIPHATIC_ORGANIC ⟶ O
19	DIGIT ⟶ 1
20	DIGIT ⟶ 2

In order to motivate the grammar representation used in our framework, consider methyl ethylene (propene) and cyclopropane with SMILES string representations as `CC=C` and `C1CC1`, respectively. The parse tree structures corresponding to the two strings are shown in Figures 3.3 and 3.4, respectively. The grammar representation for each of these molecules correspond to the sequence of production rules extracted when these structures are parsed in a bottom-up

left-corner strategy as highlighted in their respective schematics.

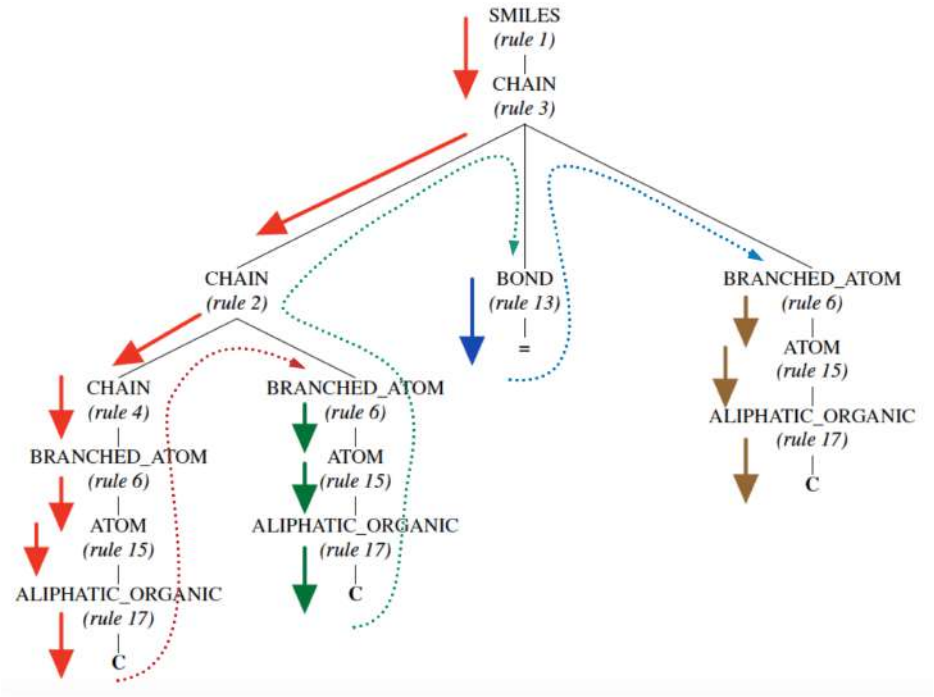


Figure 3.3: The parse-tree obtained for propene with the SMILES representation as `CC=C` using the representative grammar in Table 3.1. The sequence of production rule indices obtained while parsing the above tree corresponds to the grammar representation and is given as 1, 3, 2, 4, 6, 15, 17, 6, 15, 17, 13, 6, 15, 17

Consider the parse tree for propene given in Figure 3.3. This parse tree contains information about the various chemistry aspects of the given molecule. For instance, it contains information such as the number of aromatic carbon atoms, the presence of a ring-structure, the alternating double bonds in the ring, the presence of an aliphatic oxygen atom, and finally the active hydrogen atom attached to the oxygen atom. Moreover, this information is represented in a hierarchical manner, with the broadest class of rules at the top and increasingly more specific ones towards the bottom of the parse-tree. The parse-tree structure in Figure 3.3 is encoded using the sequence of productions used to generate the given structure as the *sentence* analogue in our language model with the individual rule indices as the equivalent *word* analogues.

Contrasting such a grammar-based representation with a purely string-based represen-

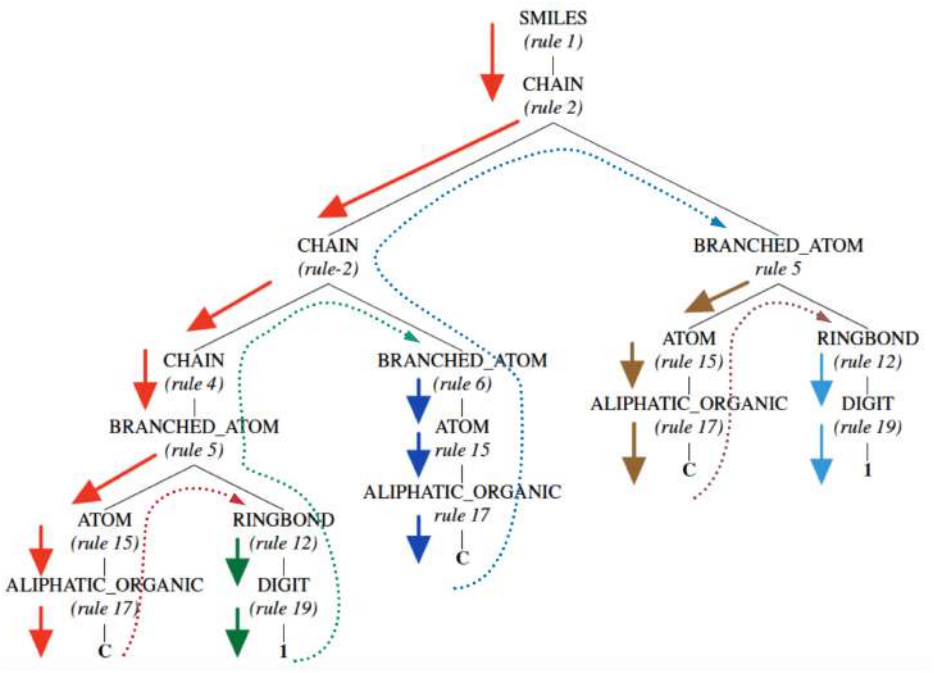


Figure 3.4: Another motivating example representing the the parse-tree structure obtained for cyclopropane with SMILES representation as `C1CC1`. The equivalent grammar representation is given as 1, 2, 2, 4, 5, 15, 17, 12, 19, 6, 15, 17, 5, 15, 17 12, 19

tation that treats each of the tokens comprising the SMILES string ('C', 'C', '=', 'C) as independent entities, the differences between the two are evident. A model trained on a purely character or string-based representation would require the model to first understand the structural relationships between the different tokens comprising the SMILES string which is not a trivial task for any neural language model architecture, and only then model the transformation between the reaction space to the product space.

Remark 1: Although the grammatical validity of a SMILES string does not necessarily mean that the corresponding compound is chemically feasibly, it is a step closer towards ensuring synthesizable molecules are predicted as the output.

Remark 2: In contrast to the English language, the proposed SMILES-grammar based molecular representation does not suffer from ambiguity with respect to its constituency parsing structure since a given (canonicalized) SMILES string cannot correspond to two completely different molecules under different contexts.

3.2.2 Sequence-to-sequence transformer

The transformer architecture was proposed recently in [46] for machine translation tasks and comprises an encoder-decoder architecture that is more parallelizable and superior to other seq2seq architectures. Transformers replaced the complex recurrent (or convolutional) neural network layers with simpler attention based mechanisms proposed in [57] combined with positional embedding for encoding sequential information. An overview of the transformer architecture as proposed in [46] is presented in Figure 3.5. In the following sections, the concepts of the encoder-decoder architecture, positional encoding, and the attention-mechanism that comprise the building blocks of a transformer are briefly described.

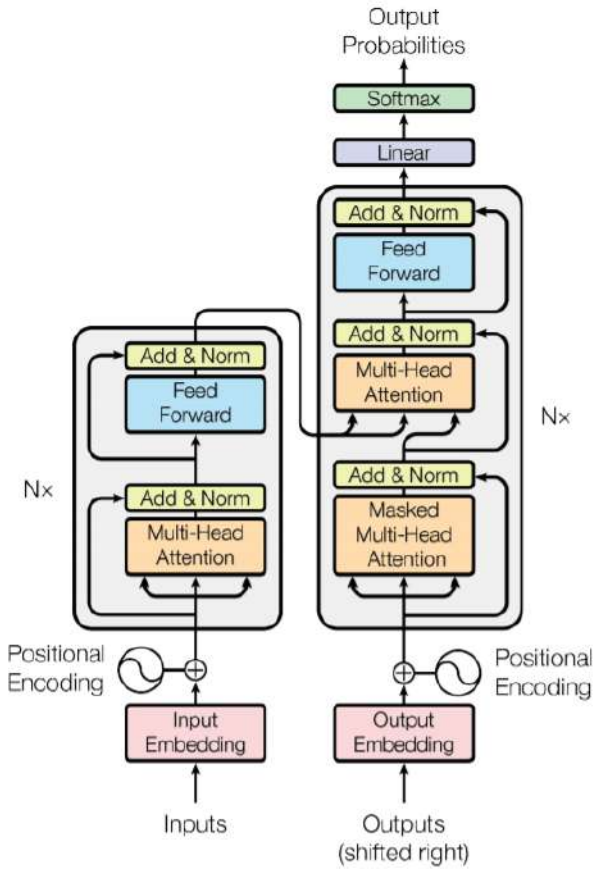


Figure 3.5: The encoder-decoder model architecture of a transformer. The left-half corresponds to the encoder whereas the right-half corresponds to the decoder, the positional information is encoded using the positional embedding, and multi-head attention mechanism aids the model in discovering relationships between groups of tokens at different stages.

Encoder-Decoder architecture

The transformer architecture primarily consists of an encoder-decoder structure, wherein the encoder maps an input sequence $x_1, x_2, \ldots, x_n$ to a continuous latent-space representation $z_1, z_2, \ldots, z_n$. Given $\mathbf{z}$, the decoder generates the output sequence $y_1, y_2, \ldots, y_n$ one element at a time, in an autoregressive manner, consuming the previously generated tokens as additional input while generating the next.

The encoder and decoder consist of stacks of identical layers, each of which are comprised of two sublayers – a multi-head attention mechanism, and a fully connected feed-forward neural network. There are residual connections around each of the sublayers along with a batch normalization. The decoder, in addition, consists of an additional layer which performs a multi-head attention over the output of the encoder.

Positional encoding

Since the transformer architecture does not contain recurrent or convolution layers, the sequential information of the tokens in a sequence is fed to the model through these embeddings. Mathematically, positional encoding is a mapping of the position of a given word (pos, an integer) in the sequence to a d-dimensional vector space ($\vec{p}_{pos}$). These mappings are characterized by sines and cosines of different frequencies, given by

$$\vec{p}_{pos,i} = \begin{cases} sin(pos/10000^{2k/d}), & \text{if } i = 2k \\ cos(pos/10000^{2k/d}), & \text{if } i = 2k+1 \end{cases} \tag{3.1}$$

The positional encodings are added to the word embeddings representing the individual tokens in a sentence, thus, the dimensions of the two embeddings, d_{word} and d_{pos}, must be the same so that the two can be summed, i.e.

$$\vec{w}(t)' = \vec{w}(t) + \vec{p}_{pos^t} \tag{3.2}$$

where $\vec{w}(t)'$ represents the word embedding with encoded position information, $\vec{w}(t)$ represents the word embedding, and $\vec{p}_{pos^t}$ represent the positional encoding.

Attention mechanism

The attention mechanism lies at the heart of the transformer architecture and allows the model to focus on different tokens in the sequence at different stages of the network, enabling it to discover multiple relationships between groups of tokens.

The attention-mechanism used in [46] is the 'Scaled-Dot Product Attention', characterized by a set of queries, keys, and values vectors. The query and key vectors are of dimensions d_k and the value vector is of dimension d_v. The attention-score then, is computed as softmax function applied over the dot-products of the queries and key vectors, scaled down by a factor of $\sqrt{d_k}$, given by

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.3}$$

where Q, K, and V are the matrices of query, key, and values vectors, respectively. The attention score computed above determines the importance that should be given to different parts of an input sequence in the current context. In order to allow the model to jointly factor in information from different representation subspaces at different positions, multi-headed attention is computed which involves computing multiple attention scores, in parallel, which are then concatenated and projected using a linear transformation to compute the multi-head attention scores as,

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \ldots, head_h)W^O \tag{3.4}$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, and $W_i^Q \in \mathbb{R}^{d_{pos} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{pos} \times d_k}$, and $W_i^V \in \mathbb{R}^{d_{pos} \times d_v}$ are the projection matrices for Q, K, and V, respectively.

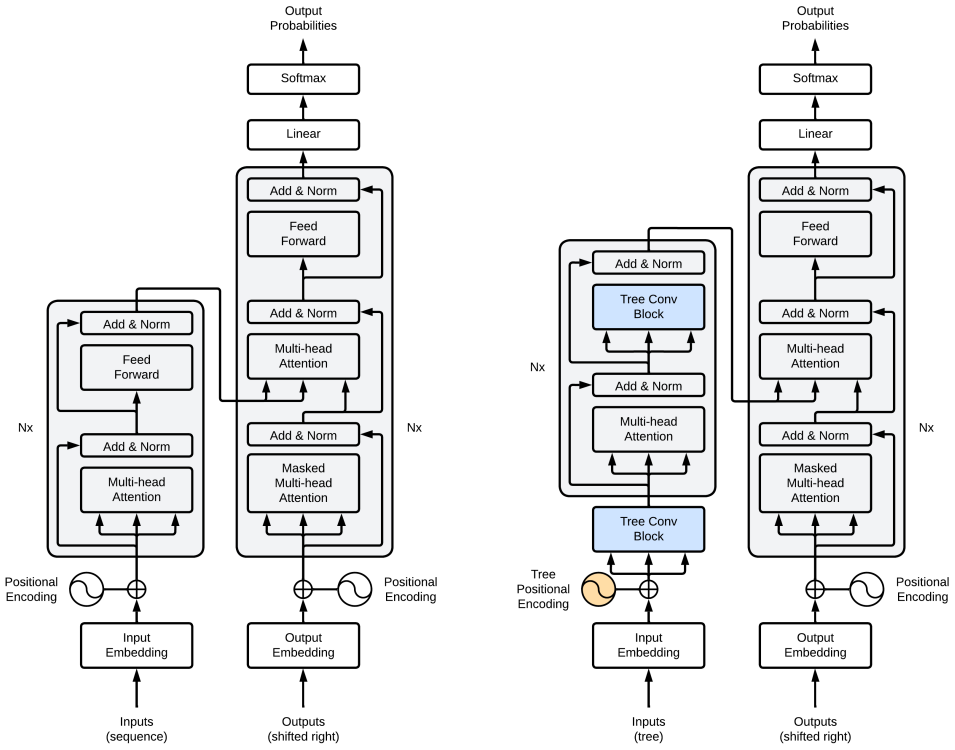
3.2.3 Tree-to-sequence transformer

Since the retrosynthesis problem is formulated as a tree to sequence modeling problem, The state-of-the-art transformer is modified so that it is capable of handling tree-based input data, as illustrated in Figure 3.6. Both the vanilla and tree-to-sequence transformer adopt an encoder-decoder architecture. The encoder maps the input to a latent space, while the decoder autoregressively decodes this latent representation to generate the output sequence. The key adaptations to the original transformer architecture involve integrating tree positional encodings and tree convolutional block sublayers, which effectively handle the hierarchical nature of the input data. In the tree-to-sequence transformer, the attention mechanism, which enables the transformer to determine relationships between tokens in the input and output, is retained. Specifically, the “Scaled-Dot Product Attention" introduced in [46], which uses query, key, and value vectors is used.

Tree positional encodings

The original transformer paper [46] employs sequential positional encodings using sinusoidal functions with varying frequencies. These encodings are added to the embeddings in the encoder and decoder to provide the model with information about the order of the input sequence. However, when working with tree-structured data, flattening into a linear sequence loses important hierarchical information. Parents and children may end up far apart in the sequential representation. Conversely, consecutive tokens in the flattened sequence may not accurately represent their true relationships in the grammar tree. To overcome these challenges and better preserve the hierarchical structure, tree positional encodings introduced in [58] are adopted. These tree positional encodings are based on the sinusoidal encodings used in the original transformer, but they are designed to explicitly encode the positions of nodes in the grammar tree. By incorporating tree positional encodings, G-MATT better utilizes the inherent tree hierarchy for more accurate predictions.

The input grammar tree T is defined as a tuple $T = (V, E, r)$, where V is the set of nodes,



(a) (b)

Figure 3.6: A schematic comparison of the vanilla sequence-to-sequence transformer in (a) and our tree-to-sequence transformer in (b). The two new components in the tree2seq transformer are the tree positional encoding (orange) and tree convolution block (blue). The tree positional encoding replaces the sequential encoding in the encoder and the TCB replaces the feed forward network in the encoder only.

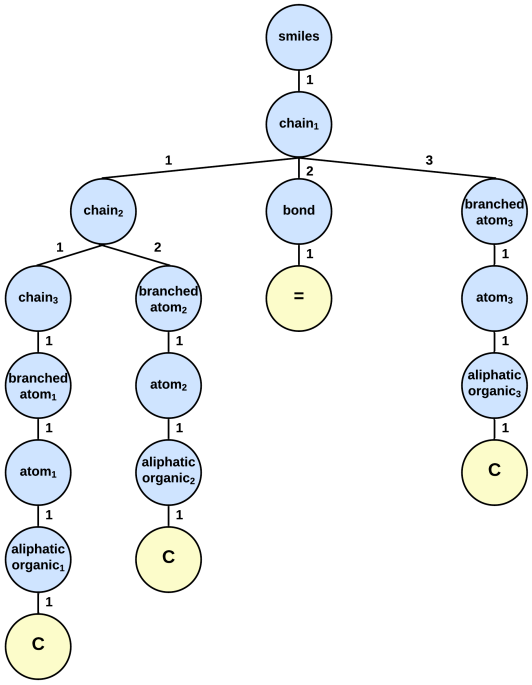
E is the set of edges, and r is the root node. For any $v \in V$ with $v \neq r$, let v^* be the parent of v. Additionally, i_v is defined as the index of v among its siblings (i.e. the children of v^*). For example, if a node has three children a, b, c (in that order), then $i_a = 1$, $i_b = 2$, and $i_c = 3$. The path from the root r to any node $v \in V$ can be uniquely identified by an edge path $p_v \in \mathbb{Z}_{\geq 0}^L$ of length L according to the following definition:

$$p_v = \begin{cases} \vec{0}_L & \text{if } v = r \\ i_v \parallel \pi_{L-1}(p_{v^*}) & \text{otherwise,} \end{cases}$$

where $\vec{0}_L$ denotes the zero vector of length L and $\pi_{L-1}(p_{v^*})$ denotes the first $L-1$ elements

of p_{v^*}.

In Figure 3.7b, the edge paths for each node in the example grammar tree are illustrated. For clarity, the edges corresponding to each child are labeled with their respective index i_v. Note that the root node `smiles` edge path is a vector of all zeros. Additionally, each child shares its parent's node path shifted to the right by one.



(a)

Node label	Path
smiles	[0, 0, 0, 0, 0, 0, 0, 0]
chain_1	[1, 0, 0, 0, 0, 0, 0, 0]
chain_2	[1, 1, 0, 0, 0, 0, 0, 0]
chain_3	[1, 1, 1, 0, 0, 0, 0, 0]
branched_atom_1	[1, 1, 1, 1, 0, 0, 0, 0]
atom_1	[1, 1, 1, 1, 1, 0, 0, 0]
$\text{aliphatic\_organic}_1$	[1, 1, 1, 1, 1, 1, 0, 0]
branched_atom_2	[2, 1, 1, 0, 0, 0, 0, 0]
atom_2	[1, 2, 1, 1, 0, 0, 0, 0]
$\text{aliphatic\_organic}_2$	[1, 1, 2, 1, 1, 0, 0, 0]
bond	[2, 1, 0, 0, 0, 0, 0, 0]
branched_atom_3	[3, 1, 0, 0, 0, 0, 0, 0]
atom_3	[1, 3, 1, 0, 0, 0, 0, 0]
$\text{aliphatic\_organic}_3$	[1, 1, 3, 1, 0, 0, 0, 0]

(b)

Figure 3.7: The tree paths for the example grammar tree for propene `CC=C`. The grammar tree with edge labels is shown in (a) and the corresponding edge paths with $L = 8$ for each node in (b). The subscripts in the node labels are solely to distinguish between different nodes with same labels.

To represent the position of a node v, sinusoidal positional encodings similar to the approach in [46] are utilized. These encodings are applied individually to each element in the edge path p_v, which results in an edge encoding $EE_{v,\ell}$. For every node $v \in V$ and index

in the edge path $\ell \in \{1, \ldots, L\}$, the edge encoding components are defined as follows:

$$(EE_{v,\ell})_{2i} = \sin(\omega_i \cdot (p_v)_\ell)$$

$$(EE_{v,\ell})_{2i+1} = \cos(\omega_i \cdot (p_v)_\ell)$$

with $w_i = 1/10000^{2i/d}$ for $i \in \{1, \ldots, \frac{d}{2}\}$. The parameter d is the dimension of the edge encoding i.e. $EE_{v,\ell} \in \mathbb{R}^d$. The tree positional encoding TE_v is obtained by concatenating the edge encodings in the edge path p_v.

$$TE_v = EE_{v,1} \parallel \ldots \parallel EE_{v,L}.$$

The resulting positional encoding has a dimension of $d \cdot L$, equal to the model size d_{model} or the node embedding dimension. In Figure 3.7a, the edge and tree positional encodings are illustrated for the `bond` node in the example grammar tree for propene with $d = 4$. In our model, L is set to 64, d is set to 4, and d_{model} is set to 256. The choice of L is based on the height of the largest SMILES molecular grammar tree in the dataset, ensuring that all paths are well-defined.

The proposed tree positional encodings offer several desirable properties, as demonstrated in [58]. Like the original sequential encodings, tree encodings satisfy uniqueness. This is because the edge path from the root r to a node v is unique, and the sinusoidal encodings are injective for each frequency ω_i due to the transcendality of π. Furthermore, the encoding of a child node shares the first $d \cdot (L-1)$ dimensions with its parent's encoding, shifted right by d dimensions. This property, combined with the uniqueness property, enables the model to distinguish whether a node is an ancestor or sibling of another node. Finally, the mathematical properties of sine and cosine allow the encoding for a node to be efficiently computed by a linear combination of its sibling's encodings. These characteristics of tree positional encodings provide valuable hierarchical information, allowing the model to effectively capture the structural relationships within the grammar tree.

Tree convolutional block

Inspired by [59], a modified Tree Convolutional Block (TCB) sublayer is incorporated into the tree-to-sequence transformer architecture, enabling it to effectively handle tree-structured data. This block provides each node access to information about its parent and children, thereby explicitly incorporating hierarchical information into the model. The TCB replaces the feed-forward network following the attention sublayer in the vanilla transformer encoder [46]. Additionally, a TCB is introduced after the positional encoding, which combines the parent, children, and current node embedding before any self-attention is applied. The TCB is not included in the decoder since it predicts a sequence of SMILES tokens and does not handle tree information.

For a given node x, the current node is represented as x_t, its parent as x_p, and the average of its children nodes as x_c. Specifically, if node x has children $x_{c_1}, x_{c_2}, \ldots, x_{c_N}$, then $x_c = \frac{1}{N}\sum_{i=1}^{N} x_{c_i}$. The single-layer tree convolution block for node x is computed as follows:

$$TCB_1(x_t, x_p, x_c) = \mathrm{ReLU}(x_t W_t + x_p W_p + x_c W_c) W_2 + b_2.$$

In cases where the current node does not have a parent or children nodes, x_p or x_c are replaced with a learned embedding v_p or v_c respectively.

The TCB can be generalized to multiple layers, where an L-layer TCB consists of L consecutively stacked single-layer tree convolution blocks. The single-layer TCB captures information from nodes that are one step away from the current node. In contrast, the L-layer TCB combines information from nodes up to a maximum distance of L away. Consequently, increasing the depth of the TCB allows the model to access more surrounding context. For our model, L is set to 2 for all TCBs.

Since each encoder layer contains a tree convolution, the internal representation of each node will progressively include more information about the neighboring nodes. This enables the attention sublayer to attend to entire components within the tree rather than individual

nodes. By incorporating multi-layer TCBs, the model gains a deeper understanding of the hierarchical relationships within the molecular grammar tree, thus increasing its capacity to learn structural relationships relevant to the underlying chemistry.

It is important to note that a modification has been made to the original TCB architecture proposed in [59], where the children nodes are used instead of the left-sibling in the convolution process. Improved performance was observed in our empirical experiments with this modification, and two possible explanations for this behavior are offered. Firstly, by utilizing children nodes in the convolution, information can flow both up and down the tree, rather than being limited solely to the upward direction when using left-siblings. This is useful because nodes lower in the tree, which typically describe the molecular composition or bonding, will have access to important local context. Secondly, most production rules in the grammar involve a limited number of children, typically two. As a result, the order information of siblings is of relatively low importance, making the use of children nodes a viable and efficient approach. Moreover, information about the children is effectively retained by averaging their representations.

To visualize the behavior of the TCB, an example of its operation on the grammar tree for propene is provided in Figure 3.8. Each colored region represents the convolution window centered around the corresponding node. It is evident from this illustration that the TCB enables the model to explicitly incorporate information about the surrounding local structures. This prior chemistry-aware knowledge allows the architecture to more effectively model the relationships between molecules, as it possesses richer structural details.

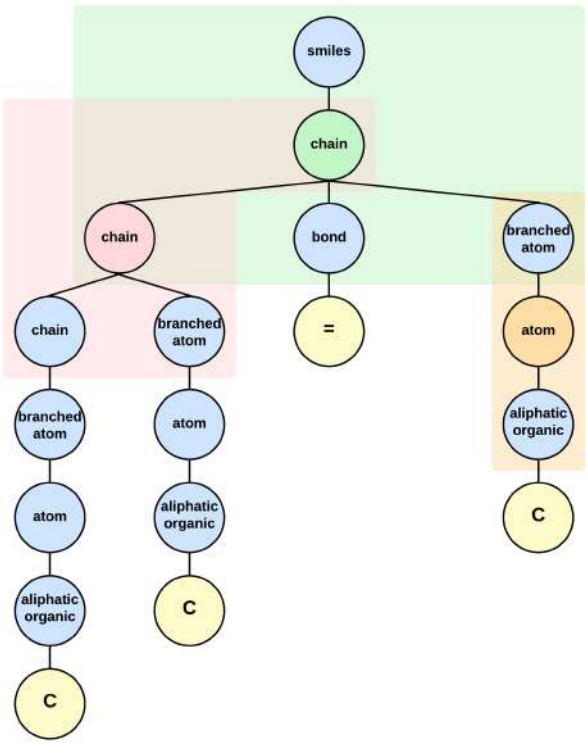


Figure 3.8: The grammar tree for propene `CC=C` and three example convolution windows in red, green, and orange. Each convolution window operates on the respectively colored node, thereby providing the node with information about its parent and children.

3.2.4 Beam search

During inference, the model's performance is evaluated using a beam search decoding strategy to autoregressively predict the output. The decoding process begins with a `<START>` token, and the transformer generates the next token from the latent space using the current predicted sequence as input. The beam search procedure, with a specified beam size B, determines the top-B tokens at each step based on their likelihood and retains the top-B sequences as the search frontier for the next step. If $B = 1$, the transformer employs a greedy strategy, selecting the maximum likelihood token at each decoding step. By using beam search, the model performance could be thoroughly evaluated and compared with the top-k accuracy reported in other similar works. A beam search size of $B = 10$ is used for all evaluation metrics outlined in Section "Results". A schematic of the beam-search decoding

procedure with $B = 3$ is depicted in Figure 3.9.

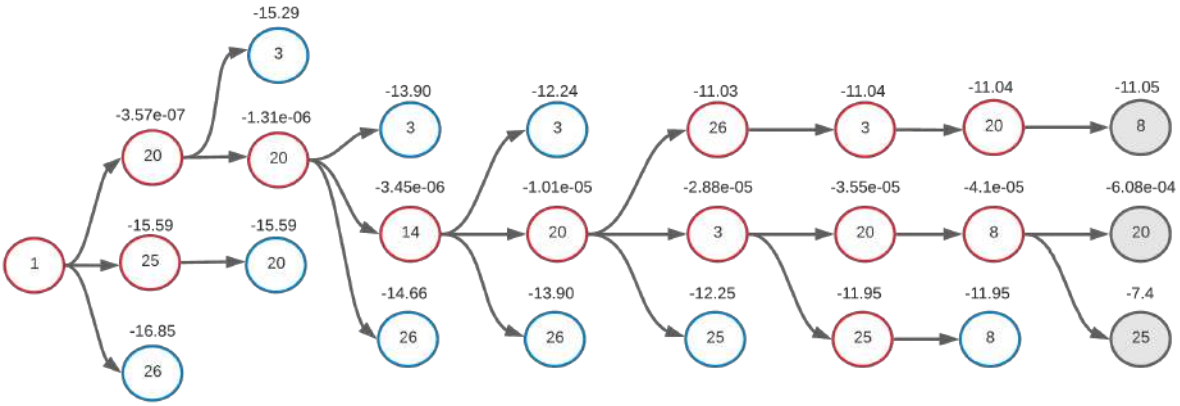


Figure 3.9: A schematic for a partially completed beam search procedure with beam size $B = 3$. At each decoding step, the 3 most likely sequences are preserved and used as the search frontier. The B most likely tokens for each sequence is computed and the top-B sequences are used for the next step. The completed sequences are then used to reconstruct the corresponding SMILES string. The log-likelihood values are indicated above each node in the schematic.

3.3 Dataset and model training

3.3.1 Dataset

For the forward reaction prediction model, two standard reaction datasets – Jin's USPTO dataset [49] and its subset of 80 reactions for benchmarking reaction prediction against human organic chemists in [50] were used. Lowe's grants database [60], based on the text mining work done on US reactions patents granted between 1976 and 2013, has now become one of the standard datasets for demonstrating quantitative approaches for reaction prediction. Since this dataset contains erroneous and duplicate reactions, there are several datasets derived from this mining work that address such issues – excluding stereochemical information, retaining only single product reactions, and removing certain class of reactions. Jin's USPTO dataset is one such derived datasets that only includes single-product reactions. Therefore, this dataset is primarily used to evaluate the performance of the proposed approach, and a

comparative analysis is performed using the latter dataset. Table 3.2 summarizes the two datasets.

For the retrosynthesis problem, the USPTO-50K retrosynthesis prediction dataset, which is a curated subset of the US Patents and Trademark Office's (USPTO) database [61], was used. This dataset has been further classified into ten distinct reaction classes [62], and the number of reactions in each class is detailed in Table 6.11. The filtered dataset exclusively contains reactants and products, with the reagent information removed, and the SMILES strings are canonicalized. It is a widely used benchmark dataset in the literature for evaluating retrosynthesis model performance, offering the flexibility to train models for both known and unknown reaction class scenarios.

Table 3.2: A summary of the datasets used for training the forward and retrosynthesis prediction models

Dataset	**train**	**valid**	**test**	**total**
Jin's USPTO (forward)				
with (sanitized) single product				479,035
in grammar	385,429	28,269	37,676	**451,374**
Human dataset (forward				
with (sanitized) single product	-	-	80	80
in grammar	-	-	78	78
USPTO-50K (retrosynthesis)				
with (sanitized) molecules				50,037
in grammar	38,899	4,849	4,846	**48,594**

Since our models use the SMILES grammar-based representations, the input molecules' SMILES strings from the database were parsed using the grammar outlined in Section "**??**". As a result, certain molecules may not be *in grammar* are thus are not included in the model training process. However, it is worth noting that expanding the grammar to include these

molecules is straightforward by adding corresponding rules for additional elements such as Si, Pt, Zn, Mg, and others. Table 3.2 provides a summary of the reaction database, indicating the number of reactions that are in grammar, along with the train, validation, and test-set splits.

3.3.2 Model architecture and training

The transformer architecture, like any other machine learning architecture, consists of several hyperparameters that need to be tuned for achieving the desired performance. Therefore, the best hyperparameter values are searched by evaluating various model architectures on the validation set of Jin's USPTO dataset. Table 3.3 describes the possible hyperparameters along with their tuned values characterizing the forward prediction model. The forward prediction model is characterized by $\sim$ 5M training parameters. The model was trained using the Adam optimizer [63] with beta $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$, and a learning rate that is characterized by a fixed number of warmup steps and given by

$$lr = d_{model}^{-0.5}.min(step\_num^{-0.5}, step\_num * warmup\_steps^{-0.5}) \tag{3.5}$$

where d_model is the embedding dimension (positional). At the training stage, in order to avoid overfitting, a dropout layer is used for both the feed-forward networks and the attention-mechanism, for the encoder as well as the decoder. A loss function based on sparse categorical cross entropy between the predicted and actual target sequences is minimized. The model was trained using TensorFlow 2.1 and python 3.7 for 60 epochs. For generating the parse-trees and extracting grammar-based features, the Natural Language ToolKit (NLTK) 3.4.5 library was used. The molecular datasets were processed using the 2019 release of RDKit library.

Table 3.3: Possible and best hyperparameter values for the model architecture described in Figure 3.5

Hyperparameter	**Possible values**	**Final model**
Embedding dimensions	128, 256, 512	256
Attention heads	4, 8, 16	8
Feedforward network units	512, 1024	512
Number of layers	4, 6	4
Dropout	0.1, 0.2	0.1
Warmup steps	1k, 4k, 8k, 12k	12k

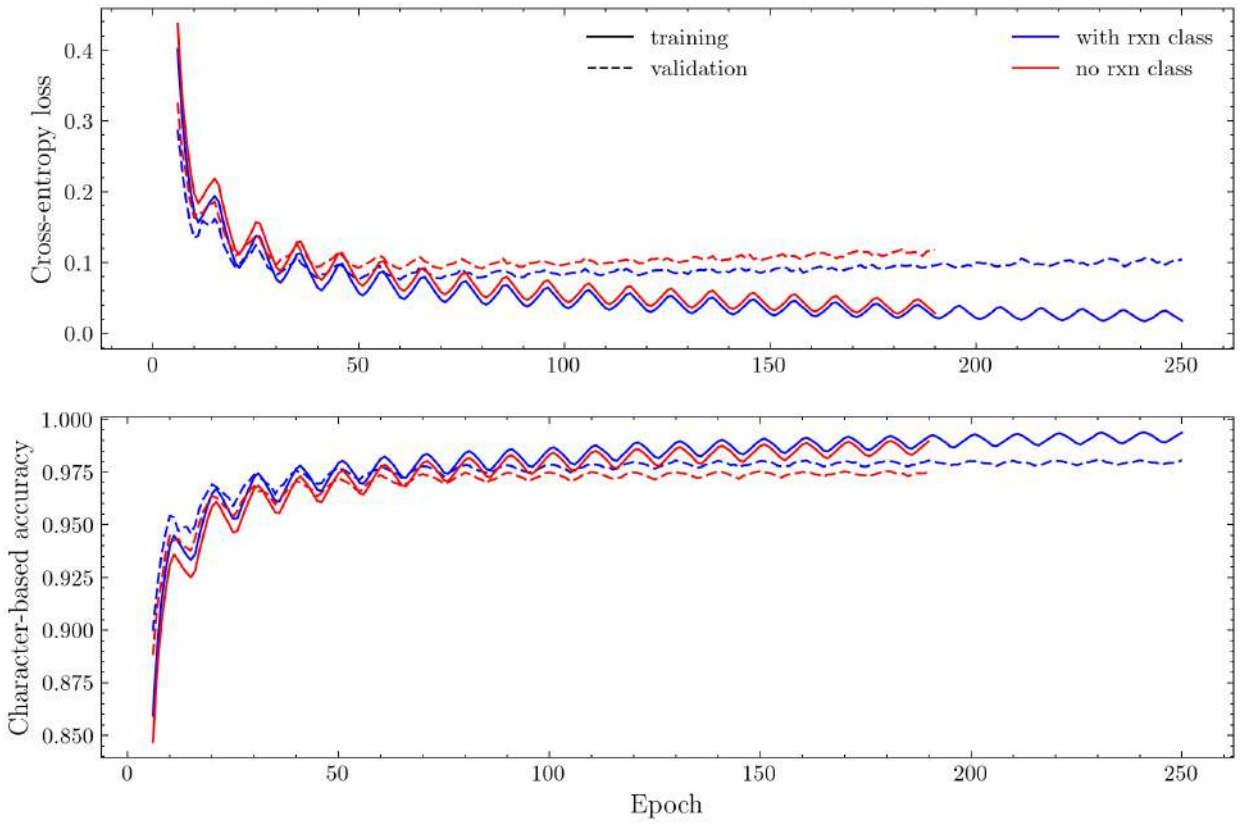


Figure 3.10: The cross-entropy loss and character-based accuracy for the training and validation set for the retrosynthesis models. The 'with reaction class model' was trained for 250 epochs and the 'without reaction class model' for 190 epochs. The oscillatory behavior of the training loss and accuracy is due to the cyclic learning rate scheduler, which has a cycle length of $n = 10$.

Similarly, for the retrosynthesis model, the optimization objective was to minimize the loss function based on sparse categorical cross-entropy between the predicted and actual target sequences. The Adam optimizer [63] with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$ was used. Additionally, a triangular cyclic learning rate schedule proposed in [64] was used, with a maximum learning rate $\eta_{max} = 5 \times 10^{-4}$ and a minimum learning rate $\eta_{min} = 1 \times 10^{-4}$. The number of epochs per cycle was set to $n = 10$ and learning rate decay per cycle was $\gamma = 0.98$. During the training stage, dropout layers with probability $p = 0.2$ were incorporated in both the attention sublayer and TCBs to prevent overfitting. The lengths of the input and output representations to the model were fixed at 350 and 121, respectively, based on the longest representation lengths observed in the training set. The retrosynthesis model for known reaction class was trained for 250 epochs, while the model for the unknown reaction class was trained for 190 epochs. These choices ensured adequate training convergence and achieved desirable model performance. In Figure 3.10, the cross-entropy loss and character-based accuracy are plotted for both the training and validation sets throughout the entire training phase. Notably, early stopping based on validation loss was not employed since previous works [51] have indicated that such an approach potentially undermines model performance.

3.4 Results: Forward prediction

3.4.1 Performance metrics

In order to evaluate the performance of our approach, four different measures were considered that capture different aspects of the model performance, namely – the BLEU (Bilingual Evaluation Understudy) score [65] which is a standard metric used for the evaluation of a given translation against the reference translation; the top-1, top-2, and top-3 accuracies computed by identifying perfect matches between the predictions and the actual product; syntactic validity of the predicted outputs by determining if the predicted molecules SMILES string is *in grammar* i.e. could be parsed by the given grammar; and character-based sim-

ilarity[2] between the actual and the predicted SMILES strings that measures the similarity between substructures within the given set of strings.

The above four measures capturing the performance of GO-PRO on the test-set of Jin's UPSTO dataset is summarized in Table 3.4.

Table 3.4: Results on the test set for Jin's USPTO dataset computed using the top-1 predictions

BLEU	top-1 accuracy	valid fraction	similarity
93.2%	80.1%	99.0%	95.8%

In order to further understand the model performance on this dataset, the split of similarity scores were computed for reactions in the dataset across three bins with similarity scores of more than 0.95, 0.85, and 0.75, as presented in Table 3.5.

Table 3.5: Similarity scores on the test set for Jin's USPTO dataset corresponding to the top-1 predictions

similarity ≥ 0.95	similarity ≥ 0.85	similarity ≥ 0.75
84.4%	90.3%	93.4%

Based on the above results, a few conclusions about the efficiency of the proposed approach could be drawn. First, only 1% of the predicted reaction outcomes resulted in invalid SMILES strings that could not be parsed by the given grammar. This indicates that the transformer-model has learnt the underlying SMILES grammar almost with perfection from the reaction encoding strategy. Second, the BLEU score and similarity values suggest that the predicted products are very similar to the actual products of the reaction. This is not trivial since the reactants, especially in organic chemistry reactions, often give rise to products that are significantly different from each one of them after only a few elementary transfor-

[2]computed using the SequenceMatcher routine in python that matches the longest contiguous matching sub-sequence that does not contain any unwanted (or junk) elements

mations involving addition, substitution, and elimination reactions between different groups. This is further established through Table 3.5 where the splits indicate that over 90% of the predicted products share a similarity of more than 0.85 with the actual product. Third, the top-1 accuracy of over 80% on the test set is indicative of the fact that the model has really discovered the complex transformations occurring in chemical reactions subject to the reactions conditions. A comparison of the model performance with human chemists presented in the following section validates this claim.

3.4.2 Comparison with human organic chemists

In this section, the performance of our approach on the dataset of 80 reactions used for benchmarking against human organic chemists used in [66] is reported. The test set contains 80 randomly chosen reactions from Jin's USPTO dataset, 10 from each of the 8 category of reaction templates, categorized based on their frequency of occurrence. The comparison of the model accuracy with the average performance of the human chemists across various reaction template bins is presented in Figure 3.11. The performance measures for the model on this dataset are summarized in Table 3.6.

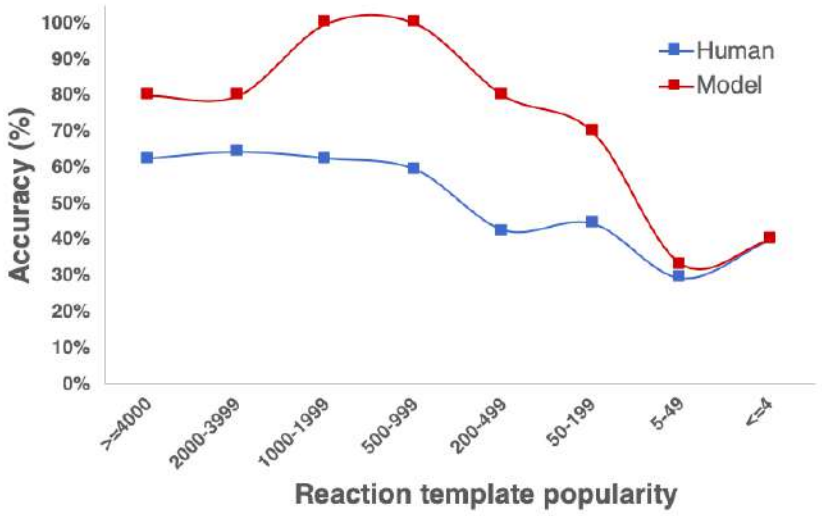


Figure 3.11: Prediction accuracy of the model and the average accuracy of human chemists versus reaction template popularity

Performance measure	(in %)
BLEU	93.2
top-1 accuracy	72.9
valid fraction	100.0
similarity score	94.4

Table 3.6: Performance measures for model on the human chemists dataset

Clearly, the model outperforms the chemists for each of the reaction template bins with 100% prediction accuracies for the second and third categories of reaction templates. Even for the increasingly rare reactions represented by the following bins, the model achieves an accuracy of over 70% except for the last two bins where the performance is comparable to the human chemists. Moreover, as is evident from Table 3.6, 100% of the predictions made by GO-PRO on this dataset correspond to valid SMILES strings, again reinforcing the advantages of a grammar ontology-based encoding strategy for reactions.

Figure 3.12 visualizes some of the incorrect predictions made by our model on this dataset. It is observed that even when the predictions were inaccurate, the predicted products were very similar to the actual product of the reaction – based on their structural forms in Figure 3.12 and also based on the BLEU and similarity scores from Table 3.6.

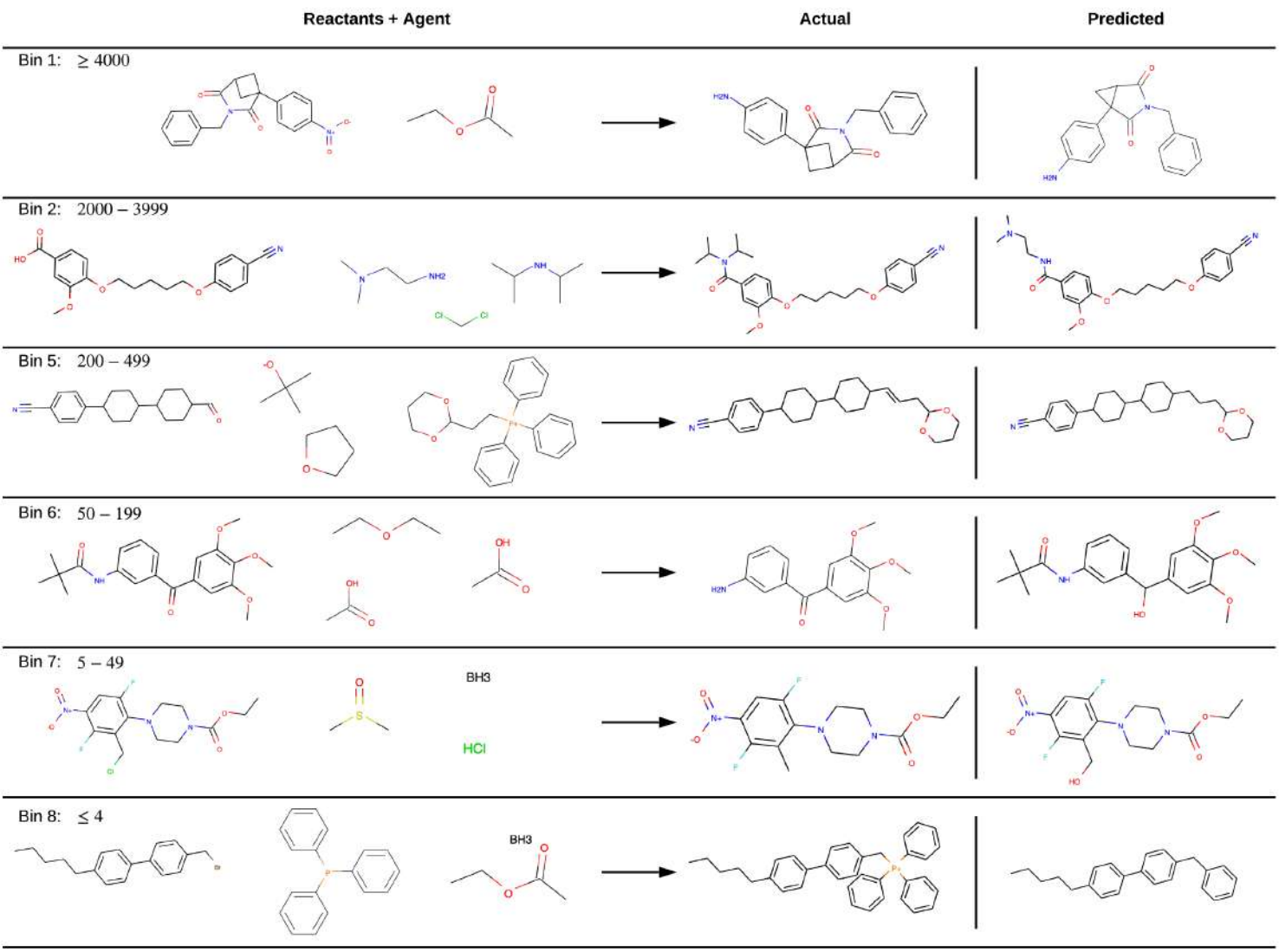


Figure 3.12: Some of the reactions in the human chemists dataset that were predicted incorrectly by our model. Even the incorrect predictions share a structure very similar to the actual product of the reaction. The bin popularity along with the their frequency of appearance in the database is indicated for each reaction.

3.4.3 Comparison with other works

The current state of the art model in the reaction prediction literature is the molecular transformer model [47]. Though the overall accuracy obtained using our approach does not outperform the best model, our model achieves an accuracy of over 80% just by using a fraction of the training parameters characterizing the model used in other works. A comparison

of the accuracies and the number of parameters characterizing the seq2seq model used in other works is presented in Table 3.7.

Table 3.7: Comparison of accuracies (in %) reported in other works involving seq2seq models using Jin's USPTO dataset

Model	# parameters	top-1	top-2	top-3
Molecular transformer[47]	12 M	88.6	92.4	93.5
S2S [67]	30 M	80.3	84.7	86.2
GO-PRO[3]	**5 M**	**80.1**	**86.3**	**88.7**

Based on this, it is claimed that our grammar-based approach significantly aids the model in learning transformations that occur in a chemical reaction by explicit incorporation of the relationships between the constituent tokens in a SMILES string. A significantly fewer number of parameters also inherently implies that the model does not have the capability to *memorize* the entire training set and therefore, overcomes overfitting during the model training stage – making it more robust and generalizable in practice. The following are the advantages of using the proposed grammar-based representation for reaction prediction

- the grammar-based representation explicitly encodes the structural information for the given molecule in a hierarchical manner with constituency relationships mapped between different components in the SMILES string. This is evident by the contrasting parse-tree based grammar-representations (Figures 3.3 & 3.4) and their equivalent character-based SMILES string representations described in the section Grammar for SMILES.

- the proposed (more systematic) representation incurs less strain on the model (in terms of number of training parameters) and consequently requires a significantly less complex model architecture for modeling the underlying transformations in a chemical reaction,

[3]using only 3 reactants and 1 agent for predicting the major product of the reaction

as seen through the results in Table 3.7 where the proposed model is characterized by only a fraction of the training parameters in other works with comparable accuracies.

- owing to the grammar-based representation, the model learns to predict the product based on the same grammar, and hence, the output SMILES strings are more likely to be syntactically valid. This is validated by the results in Tables 4 & 6 which demonstrate that 99% and 100% of the predictions made by our model (on the test set) are syntactically valid which would have been unlikely without the model learning the underlying grammar production rules.

It is imperative to note here that a relatively lower accuracy in our model could be attributed to certain assumptions and approximations made during the model building stage. First, the model was trained on only three reactants and one agent, discarding all the other molecules involved in the reaction. Moreover, even among these four molecules, those that were not *in-grammar* were dropped while encoding the reaction. Second, the SMILES grammar that used here does not include metallic ions, certain metallic catalysts, and inorganic elements, limiting the coverage of the training dataset. Third, molecules with representations of over 300 were discarded and reactions were truncated if the left-hand side representation was longer than 600. Owing to these limitations/approximations, the model is constrained and has restrictive predictive capabilities that affects the prediction accuracies. However, these could be overcome by relaxing these constraints which although does not result in an increase in the model parameters, increases the model training time significantly due to longer sequences.

3.5 Results: Retrosynthesis prediction

In this section, the performance of the trained models on the test set is presented. To assess retrosynthesis prediction, the model evaluation metrics are first defined. Additionally, the model performance is compared with other similar works in this field to benchmark

its effectiveness. Finally, the advantages of our framework along with its limitations are discussed.

Considering that a product may have multiple correct retrosynthesis pathways, not only exact matches between predicted reactions and ground truth are considered, but also the chemical similarity of precursor molecules. This evaluation is important because chemically similar molecules are likely to produce feasible and correct reactions in practice. Therefore, similarity-based metrics are incorporated to provide a more comprehensive and practical assessment of the model's performance.

To determine chemical similarity, the Tanimoto index, one of the most widely used and reliable metrics for computing the structural similarity between molecules [68], is used. The Tanimoto coefficient T_c is a value ranging from 0 to 1, representing the fraction of common molecular fingerprints shared between two molecules. For two molecules X and Y with fingerprint sets A and B respectively, the Tanimoto coefficient is computed as:

$$T_c(X, Y) = \frac{A \cap B}{A + B - A \cap B}.$$

A coefficient of zero implies that the molecules have no common fingerprints, while a coefficient of one indicates two identical molecules. Although no specific coefficient distinguishes similar and dissimilar molecules, the commonly used threshold of $T_c = 0.85$ for defining bioactive similarity is adopted. In other words, two molecules are considered bioactively similar if their Tanimoto score satisfies $0.85 \leq T_c \leq 1$.

3.5.1 Evaluation metrics

The following performance metrics to evaluate our models: accuracy, fractional accuracy, MaxFrag accuracy, MaxFrag bioactive similarity rate (BASR), and invalid rate. Accuracy measures the fraction of reactions in which the predicted precursor molecules perfectly match the ground truth. Fractional accuracy measures the proportion of correctly predicted precursor molecules relative to the total number of ground truth precursors. MaxFrag accuracy

represents the prediction accuracy of the maximal fragment, which corresponds to the longest precursor molecule based on SMILES length. MaxFrag BASR quantifies the fraction of reactions where the predicted maximal fragment is bioactively similar (with a Tanimoto score between 0.85 and 1) to the ground truth maximal fragment. Lastly, the invalid rate calculates the percentage of syntactically or grammatically invalid SMILES predictions. A schematic illustrating these metrics for an example molecule in the test-set is presented in Figure 3.13.

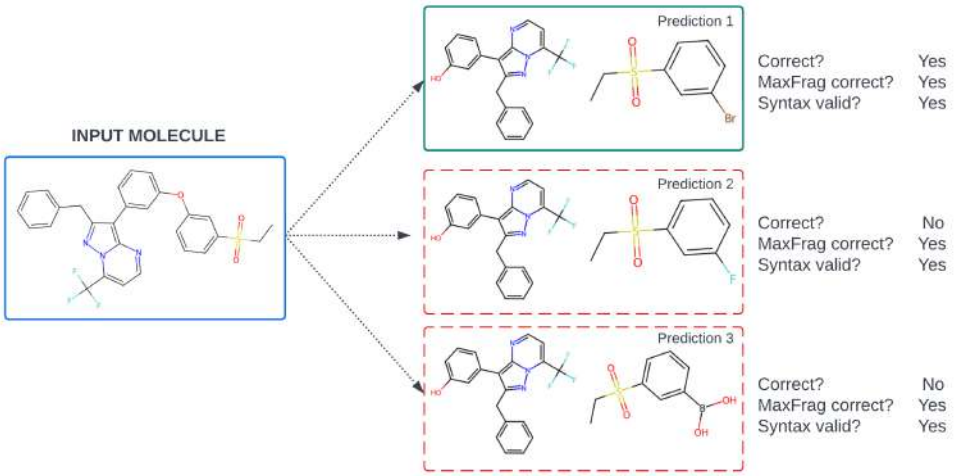


Figure 3.13: Top-3 predictions generated by our model for a given input molecule in the test-set. The first (most likely) prediction matches the ground truth, the second is chemically feasible but incorrect since it doesn't exactly match the ground truth, and the third prediction is chemically incorrect. All predictions are syntactically valid and have correctly predicted the maximal fragment.

Based on the observations from the figure, it is noticed that a prediction is considered incorrect if it does not exactly match the set of precursors in the ground truth, even if it is chemically feasible. For example, the most likely prediction (prediction 1) matches the ground truth precursors and is considered correct. However, the second most likely prediction (prediction 2) contains a fluorine (F) instead of bromine (Br) for one of the precursors. Although both elements are halogens and are chemically correct, the prediction is still deemed incorrect when computing accuracy. As a result, the reported exact-match accuracy serves as a conservative lower bound on the model's performance. In practical scenarios, the actual model performance is likely to be better than expected based solely on exact-match accuracy.

3.5.2 Model performance

Table 3.8 presents the performance evaluation measures calculated on the test set of the USPTO-50K dataset for both the known and unknown reaction class scenarios. Additionally, the class-wise prediction accuracy and invalid rate for both known and unknown reaction class scenarios are computed and shown in Figure 3.14 and Figure 3.15, respectively. This granular evaluation provides insights into how the model performs in distinct reaction categories and in different reaction contexts. Note that the retrosynthesis prediction accuracies for certain reaction classes – reaction class 3 (corresponding to C–C bond formation) and class 4 (corresponding to heterocycle formation) are lower compared to other reaction classes. This could possibly be attributed to the inherent complexity of such reactions in terms of dependence on key reagents that determine reaction outcomes (and hence retrosynthesis) and a wide array of reactions (for reaction class 3) that are grouped together in the same reaction class – such as aldol reactions, Diels–Alder reaction, Grignard reaction, cross-coupling reactions, the Michael reaction, the Wittig reaction, and so on. The combined effect of these two factors could possibly be the reason behind relatively lower accuracy for reaction class 3. Similarly, for reaction class 4, lower prediction accuracy could be attributed to a lack of enough training examples. The complete list of reaction classes and corresponding number of examples in the train, validation, and test set are listed in Table 6.11 and corresponding class-wise performance metrics in the supplementary information.

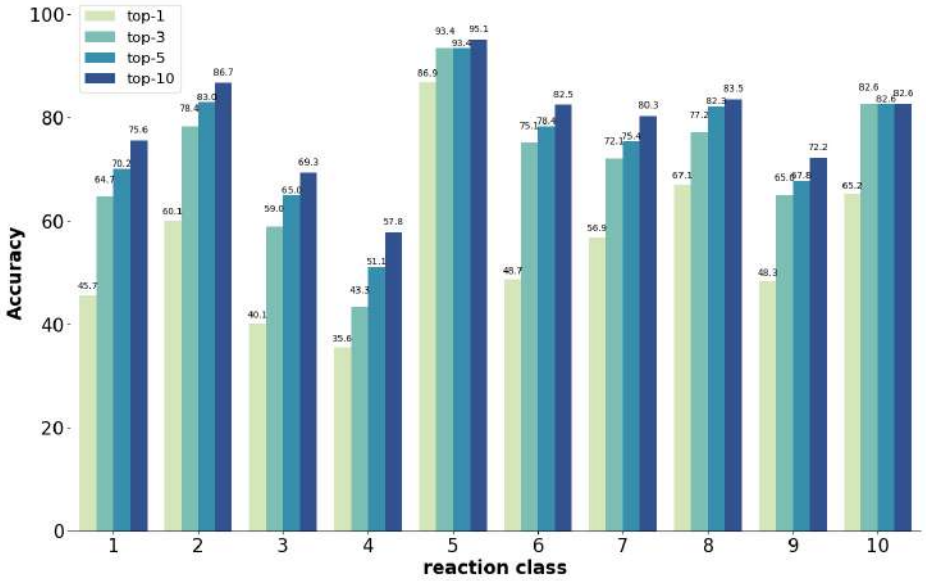
Table 3.8: Retroynthesis models' performance metrics on the test set

Performance measure	top-k measure (%)				
	1	2	3	5	10
Known reaction class					
Accuracy	51.0	64.3	70.0	74.6	79.1
Fractional accuracy	64.7	74.9	79.4	83.2	86.8
MaxFrag accuracy	60.6	71.6	76.5	80.4	84.1
MaxFrag BASR	74.8	82.2	85.4	88.3	92.3
Invalid rate	1.5	2.7	4.0	6.0	9.8
Unknown reaction class					
Accuracy	41.6	54.0	60.4	67.6	73.1
Fractional accuracy	52.0	63.7	69.8	76.3	81.3
MaxFrag accuracy	49.0	60.5	66.5	72.7	77.8
MaxFrag BASR	60.0	70.8	76.4	82.2	86.8
Invalid rate	1.3	2.0	2.6	3.8	6.

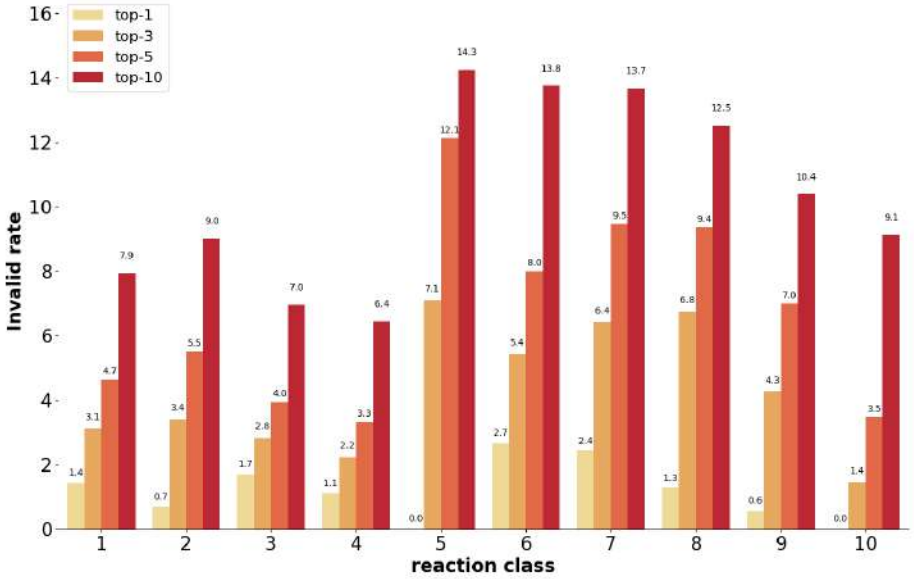
3.5.3 Comparison with other works

In this section, the performance of our G-MATT model is benchmarked against other transformer-based approaches reported in the literature. All models in the comparison are trained and evaluated on the same USPTO-50K dataset used for training and evaluating our model. To provide a comprehensive comparison, the top-k prediction accuracy and top-k invalid rate are evaluated and presented in Table 3.9. Our findings demonstrate that for both the known and unknown reaction class scenarios, the proposed G-MATT model achieves nearly state-of-the-art performance in terms of prediction accuracy and invalid rate.

In the case of the known reaction class scenario, G-MATT's top-k prediction accuracy is, on average, only 4.3% lower than that of Lin et al.'s state-of-the-art model [71]. However, G-MATT has an 1.25% improvement in invalid rate over Lin et al.'s model. Similarly, for the unknown reaction class scenario, G-MATT's top-k prediction accuracy is within 3.1% of Lin et al.'s model, while outperforming it by 1.13% on the invalid rate. Additionally, when compared to SCROP, which utilizes an additional transformer model to correct incorrect



(a)

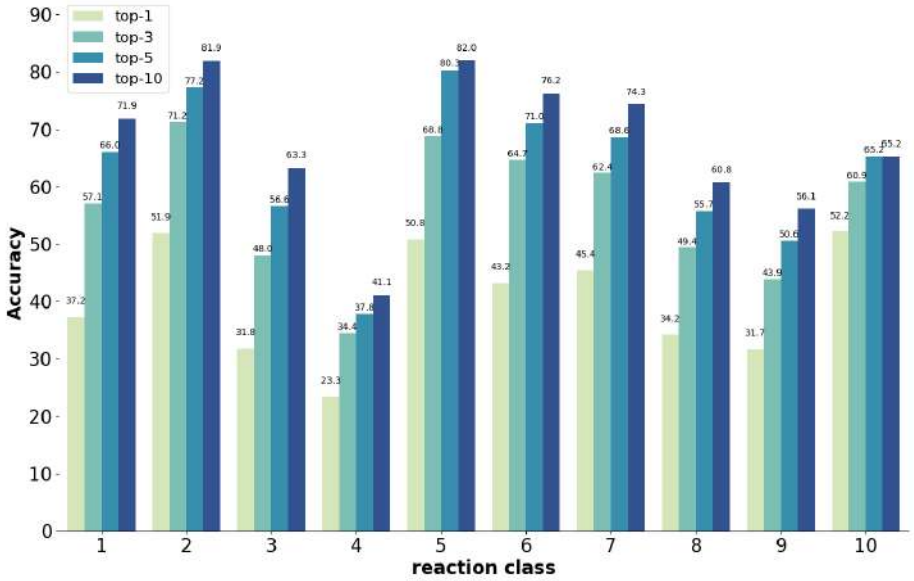


(b)

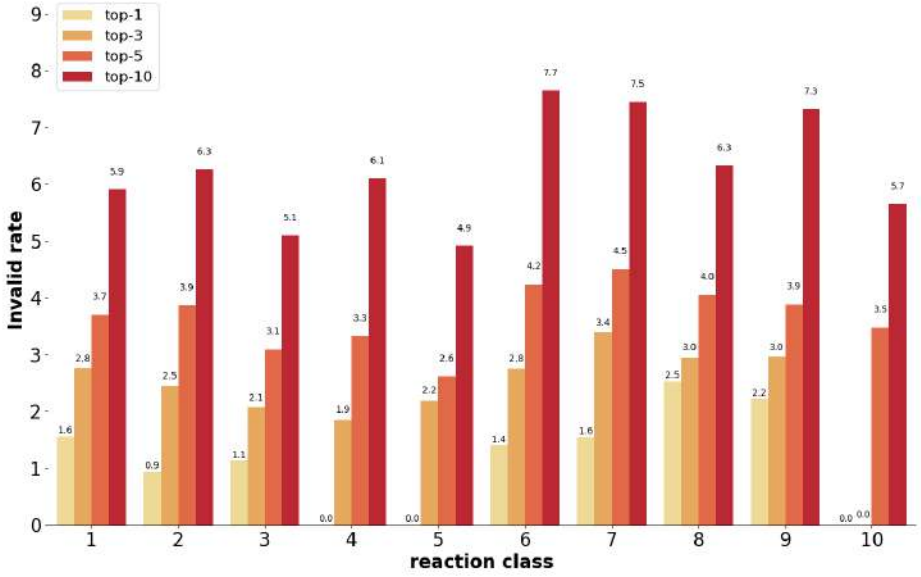
Figure 3.14: The class-wise top-1, top-3, top-5, and top-10 prediction accuracy (a) and invalid SMILES string rate (b) for retrosynthesis prediction with known reaction class.

predictions made by a primary retrosynthesis transformer model, G-MATT, with its single transformer architecture, outperforms the prediction accuracy for the unknown reaction class scenario across several top-k values.

It is worth noting that G-MATT, despite its relative simplicity compared to other ap-



(a)



(b)

Figure 3.15: The class-wise top-1, top-3, top-5, and top-10 prediction accuracy (a) and invalid SMILES string rate (b) for retrosynthesis prediction with unknown reaction class.

proaches in the literature, achieves nearly state-of-the-art performance in terms of prediction accuracy and invalid prediction rate. This can be attributed to several key characteristics that distinguish G-MATT. Firstly, G-MATT benefits from richer input representations based on SMILES grammar trees, as opposed to solely relying on linear SMILES strings. Secondly,

Table 3.9: Comparison with other transformer-based retrosynthesis models trained on USPTO-50K dataset

model	top-k accuracy (%)				top-k invalid rate (%)			
	1	3	5	10	1	3	5	10
Known reaction class								
Liu et al. [69]	37.4	52.4	57.0	61.7	12.2	15.3	18.4	22.0
Grammar seq2seq [29]	43.8	57.2	61.4	66.6	4.4	7.2	8.4	9.6
SCROP [70]	59.0	74.8	78.1	81.1				
Lin et al. [71]	54.3	74.1	79.2	84.4	2.3	4.9	7.0	12.1
T2T [72]	54.1	68.0	69.0	70.1				
G-MATT (proposed)	51.0	70.0	74.6	79.1	1.5	4.0	6.0	9.8
Unknown reaction class								
Liu et al. [69]	28.3	42.8	47.3	52.8				
Grammar seq2seq [29]	32.1	44.3	48.9	54.0	5.1	7.4	8.4	9.7
Chen et al. [73]	39.1	62.5	69.1	74.5				
Karpov et al. [51]	40.6	42.7	63.9	69.8				
SCROP [70]	43.7	60.0	65.2	68.7	0.7	1.4	1.8	2.3
Lin et al. [71]	42.0	64.0	71.3	77.6	2.2	3.7	4.8	7.8
Tied two-way TF [54]	47.1	67.1	73.1	76.3	0.1	0.2	0.6	10.2
G-MATT (proposed)	41.6	60.4	67.6	73.1	1.3	2.6	3.8	6.3

the inclusion of tree positional encodings allows the model to accurately capture the hierarchical structure present in the trees. Lastly, the use of tree convolution blocks within the transformer architecture enables the model to attend to local molecular structures, such as functional groups, thereby providing valuable contextual information. These factors collectively contribute to G-MATT's impressive performance.

While G-MATT already demonstrates competitive results, it is acknowledged that further improvements are possible by incorporating additional model training strategies from prior works in the literature. Our primary objective is not to pursue the state-of-the-art performance, but rather to showcase the value of utilizing alternative SMILES grammar-based tree representations for reaction prediction tasks. The effectiveness of G-MATT highlights the potential benefits of considering underlying molecular structures and hierarchical information, leading to improved performance in retrosynthesis prediction.

3.5.4 Near-miss predictions

To further analyze the model's performance, an analysis of incorrect predictions from a chemistry standpoint is conducted. Specifically, the Tanimoto similarity of incorrectly predicted maximal fragment molecules with the ground truth is computed. Only the maximal fragment is focused on since there are multiple molecules in the target reaction pathway. This analysis provides us with insights into the degree of similarity between the incorrectly predicted precursors and the ground truth, which is important as chemically similar precursors are likely to lead to feasible and accurate reactions in practice. Table 3.10 below presents the percentile splits of incorrect predictions, categorized based on their Tanimoto similarities.

Model	Tanimoto similarity (%) (incorrect predictions only)		
	$0.5 \leq T_c$	$0.7 \leq T_c$	$0.85 \leq T_c$
Known reaction class	66.6	46.8	31.5
Unknown reaction class	78.8	63.7	48.7

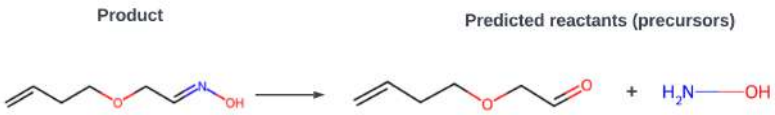
Table 3.10: Distribution of Tanimoto coefficient scores across *incorrect* top-1 predictions for both the models

3.5.5 Attention-map for molecules

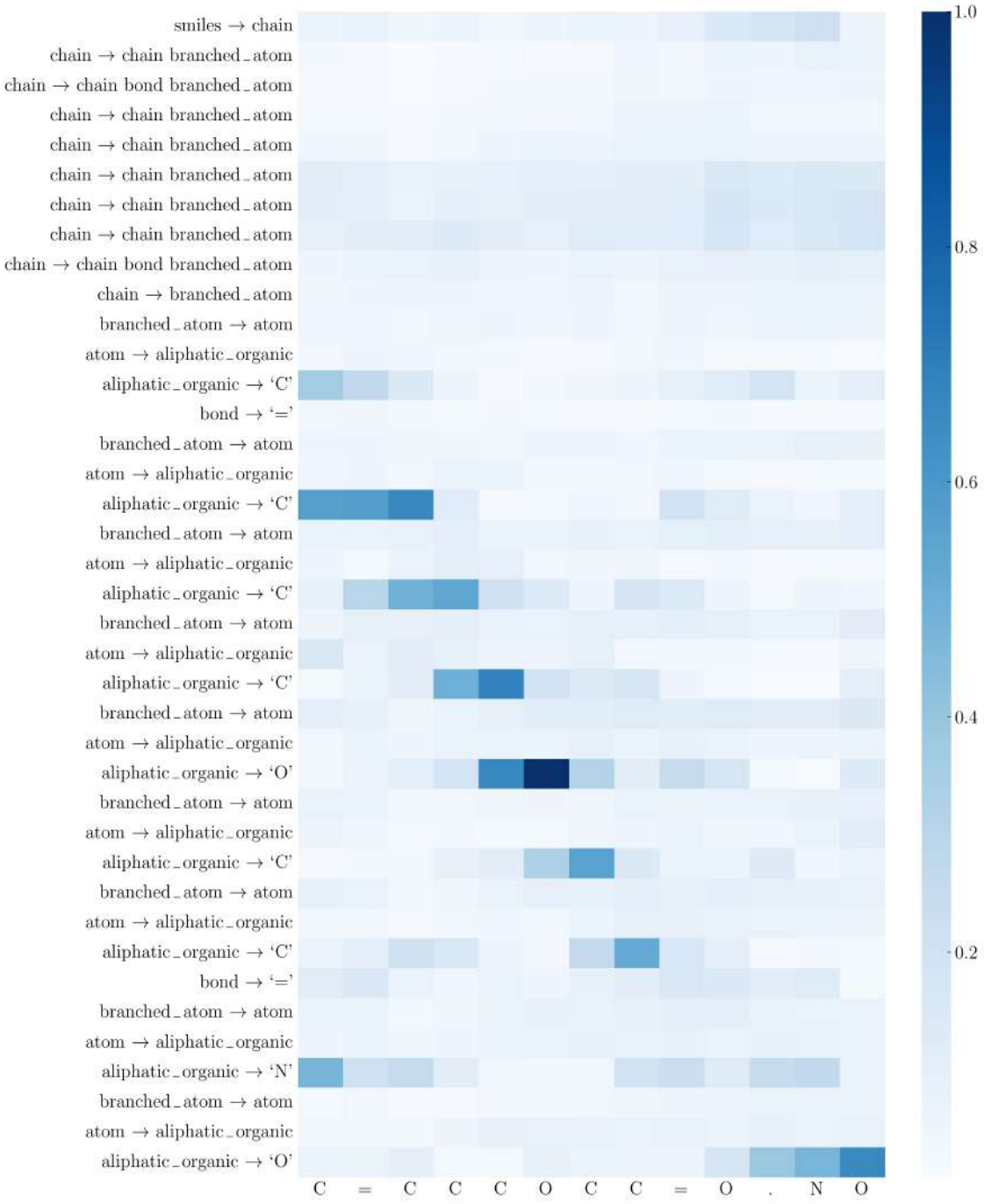
To gain deeper insights into the grammar-based molecular representation, an analysis of the attention weights computed by the attention sublayers is conducted. The cross-attention modules in the decoder are used to calculate the average attention scores across all of them. The resulting attention heatmap is showcased in Figure 3.16. Higher attention scores indicate a more pronounced correlation between the corresponding tokens, signifying their greater relevance in the translation process. This investigation demonstrates how the transformer utilizes the grammar-based representation to identify molecular structures.

The attention mechanism exhibits high attention scores along the diagonal, indicating its ability to correctly associate atoms and bonds in the input molecule with those in the precursor molecules during translation. For instance, Figure 3.16 illustrates an example reaction

where the bond disconnection site is the `N` atom in the `-N=O` functional group. From the attention map, it is observed that both `aliphatic_organic` → `O` and `aliphatic_organic` → `N` have the highest attention scores for the `'.'` token, which serves as the separator between the precursor molecules. This confirms that the model has accurately identified the bond disconnection site. This pattern is further validated through additional examples provided in Appendix B where the bond disconnection sites – `aliphatic_organic` → `S` in Figure B.1 and `aromatic_organic` → `c` in Figure B.2 – have the highest attention scores with molecule separate token. Furthermore, it is observed that atoms in the precursor molecules (horizontal axis) often have high attention scores with multiple adjacent atoms or bonds in the corresponding product (vertical axis), and vice versa. These findings strongly suggest that due to the convolutional operations on the hierarchical tree representation, G-MATT considers the context of neighboring atoms while making predictions and identifies the importance of local structures within the molecules. This capability captures more comprehensive relationships between atoms and bonds, improving the accuracy of predicted pathways.



(a) An example top-1 prediction to study the transformer attention map



(b) Average attention scores for top-1 prediction on the retrosynthesis reaction `C=CCCOCC=NO` ⟶ `C=CCCOCC=O.NO` extracted from the transformer decoder cross-attention sublayer

Figure 3.16: Molecular attention map for the SMILES grammar tree of the product and SMILES strings of the reactants (precursors) for an example reaction

Another noteworthy observation is that G-MATT's attention maps are generally sparse, with several input tokens having relatively lower cross-attention scores. This sparsity indicates the efficiency of the tree representation, which inherently imposes structure on the tokens. Consequently, the model captures both the semantics and syntactical structure of tokens, allowing it to focus on the most crucial tokens while resolving uncertainty during predictions. The model can infer the remaining tokens from the structural constraints it has learned. This ability to attend selectively to important tokens contributes to the model's performance despite its relatively low complexity. Additional cross-attention maps for the G-MATT model are provided in the Appendix B.

3.6 Conclusions

For the forward prediction problem, an approach for exploiting the grammar underlying the SMILES representation for molecules was presented. It was shown that grammar-based representations offer several advantages inherent to the reaction prediction task. First, they reduce the strain on the model training stage incurred while modeling relationships between individual tokens in a text-based molecular representation (such as SMILES) by encoding such relationships explicitly in the input and target sequences using the underlying grammar. Second, they overcome over-parameterization in complex machine learning architectures typically used in the reaction prediction tasks as observed through the significantly reduced number of training parameters in our proposed architecture. Third, such representations ensure syntactic validity of the molecular representations predicted as outcomes of chemical reactions, taking us a step closer towards constraining the model to predict synthesizable molecules. The proposed approach resulted in 99.0% syntactic validity of the prediction with an overall accuracy of 80.1% using a model characterized by 5M parameters. In contrast, the current state of the art in reaction prediction achieves an accuracy of 88.6% using a model characterized by 12M parameters, significantly higher than the number of parameters used in our model. It was concluded that context-free grammars (CFGs) could be exploited

to develop efficient representations for reaction prediction frameworks that encode reactions hierarchically, reflecting the peculiar characteristics of molecular transformations inherent to a chemical reaction. Moreover, the accuray could be further improved by implementing a beam-search approach so that the ground truth could be compared against a set of possible reaction outcomes. We therefore claim that ontologies that incorporate prior structured-information about the constituents would significantly aid the machine learning models used in such applications.

For the retrosynthesis prediction problem, a novel tree-based transformer architecture was proposed, which represented molecular inputs as hierarchical trees instead of text-based string representations. The developed tree transformer significantly improved upon previous approaches in three key aspects. Firstly, the SMILES grammar was leveraged to represent molecules, which lead to incorporation of explicit chemistry information to make the model more domain-aware and robust. Secondly, tree positional encodings were introduced to indicate the position of nodes in the grammar tree, thereby fully utilizing the tree hierarchy structure and inherent structural relationships. Lastly, tree convolutions were employed, providing nodes with progressively more contextual information. This context is important in identifying molecular characteristics, such as functional groups or other local structures, which might be ambiguous in the traditional SMILES string representation. Throughout our experiments, it was demonstrated that explicitly incorporating the tree structure in the transformer model takes full advantage of the benefits provided by a grammar-based tree representation. This resulted in enhanced performance and a deeper understanding of the underlying chemistry, making our tree-based transformer a promising approach for retrosynthesis prediction.

In the known reaction class scenario, a top-1 prediction accuracy of 51.0% (top-10 accuracy of 79.1%), fractional accuracy of 64.7%, and a syntactic invalid rate of 1.5%, was achieved. The model performed similarly for the unknown reaction class case, achieving a top-1 prediction accuracy of 41.6% (top-10 accuracy of 73.1%), fractional accuracy of 52.0%,

and a syntactic invalid rate of 1.3%. The attention score visualizations provided insights into the model's robustness, as it accurately identified reaction centers, considered the surrounding context of atoms and bonds, and effectively learned structural constraints. Remarkably, the G-MATT framework, despite its relative simplicity, attained nearly state-of-the-art performance in terms of prediction accuracy and invalid rate.

Going forward, our focus for future work would be to incorporate additional reaction conditions in the reaction prediction framework, address issues related to underrepresented reaction classes during model training, and further bifurcate the reaction classes into more specific reaction types would help in improving the model performance further. It is envisioned that additional model training strategies, such as model weights averaging, customized learning schedules, and exhaustive hyperparameter search, would lead to further improvements in G-MATT's performance. Moreover, performing multi-step retrosynthesis by combining our single-step model with techniques such as Monte Carlo tree search would be another future direction of research.

Chapter 4: Chemical Flowsheet Representation and Generation

The increased availability of large data sets combined with relatively easy-to-use machine-learning software has led to a plethora of purely machine-learning approaches that seem not to exploit the vast domain knowledge that is already there [2, 5]. Such approaches are appropriate for certain applications, such as vision, natural language processing (NLP), and game-playing, where there are no conservation laws and first-principles-based knowledge to leverage. Everything is in the data. Therefore, purely probability-based autocomplete techniques are appropriate and successful in these applications. However, chemical engineering processes, systems, and products are governed by fundamental laws, principles, and constitutive relations, which are mostly known. Not using such a trove of information, particularly for applications such as process flowsheet synthesis and design, seems not only inefficient but can also lead to designs that are incorrect and/or unsafe. Purely data-driven techniques mimicked from primarily big-data domains are not appropriate. On the other hand, our first-principles knowledge can be leveraged and exploited to reduce the need for large amounts of data. Thus, building hybrid AI models is more appropriate for many chemical engineering applications.

Motivated by such considerations, a hybrid AI framework is developed for process flowsheet synthesis and design. The overall goal is to develop a flowsheet that enables the conversion of input materials to desired output products efficiently, taking into account energy consumption, environmental impact, safety, operability, and many more. This involves solving the flowsheet synthesis and design problems, either sequentially or simultaneously. The synthesis problem involves identifying the underlying tasks, associated operations for each task, and their sequence in the flowsheet. On the other hand, the design problem involves determining the feasible and/or optimized operational and equipment details for the process

to be built and operated. Given the complexity of the problem, defined by the tasks and the search space, efficient and intelligent approaches are required to generate sustainable optimal solutions [3]. An essential step towards achieving this is an appropriate method for flowsheet representation that contains the necessary and sufficient information.

There is a long and successful tradition in process engineering on the use of domain knowledge for synthesis and design [74, 75, 76, 77, 78, 79]. This is built upon by augmenting such techniques with AI. However, as argued by Mann et al. [40], such hybrid approaches, irrespective of the applications, require efficient blending of domain concepts and theory with machine learning to arrive at optimal solutions. Moreover, these hybrid AI-based models need to further incorporate issues such as economics, environmental impact, operability, safety, and sustainability in the early stages of process development [79].

Facilitating the development of a hybrid AI-based approach requires an appropriate flowsheet representation. The representation needs to be concise, complete, and accurate[40]. This further needs to be integrated with automated generation methods that are flexible to incorporate domain knowledge and to allow the formulation of innovative process flowsheets for a wide range of applications. The SFILES representations developed by [80, 81, 82] was the first step in this direction and have been shown to have various applications such as flowsheet autocompletion [83], piping and instrumentation diagram generation [84], and flowsheet pattern mining [85]. However, these works exclude important flowsheet information by building purely data-driven models based only on the information in the SFILES strings. While such approaches using the text-based SMILES representation [14] for molecular generation [86, 87], reaction prediction [28, 47, 29], and property prediction [88, 89] have been successful, it must be remembered that in these cases other background information is also provided.

Just as in molecular representation, such as SMILES, where information on the number and types of atoms and their bonds are needed, the relevant details to be considered for flowsheets are the number of chemicals and their paths within the system boundary, the process

objectives and desired end goals of the flowsheet, constraints on process equipment, and so on. Thus, the context of flowsheets constrained by mass and energy conservation, equipment design factors, agreement with process design principles, energy efficiency, environmental impact, recycling of unused reactants, and so on are essential to consider. Moreover, flowsheet synthesis is different from process flow-diagram generation, where the latter only involves depicting a configuration of operations and their connectivity without the details that are necessary for the generation of flowsheet alternatives in process simulation, design, and analysis. Generating a process flowsheet (or even process flow diagrams) without taking into account the flowsheet context is akin to automatically generating molecules starting with a 'C-atom' (carbon-atom) without specifying desired properties, structural constraints (valency rules), desired molecular types, etc., leading to a practically infinite number of molecules, without knowing if any of the molecules would match the desired set of constraints. To address such drawbacks, here the computational benefits of AI/ML methods are combined with process engineering domain knowledge to develop an intelligent, hybrid AI-based representation for process synthesis and design.

In particular, a multi-level framework is presented for flowsheet representation and its use for fast, efficient, and reliable flowsheet synthesis, design, and simulation. The framework leverages established concepts and theories on process development and data-driven strategies that could be deployed on any process flowsheet to extract likely processing route patterns. The *eSFILES* representation is an *extended* SFILES representation that incorporates text-based SFILES of process flowsheets [80, 81, 82], symbolic AI-based syntax rules (formal grammar) similar to the SMILES grammar [28, 29], annotated hypergraph representation [90] for representing flowsheet connectivity, and a process ontology for efficient knowledge representation and its use.

The proposed representation comprises a base level (level 0) followed by three other levels. The base level involves representing the process flow-diagram as purely text-based representations. Level 1 involves flowsheet connectivity information, which is represented

as flowsheet hypergraphs. Level 2 incorporates specifications related to the conservation of mass and energy such as separation factors, reaction stoichiometry, and conversion rate, needed for simple model-based process simulations. Finally, Level 3 contains additional process operational specifications in terms of design parameters needed for rigorous process simulation as well as options for process innovation through optimization and intensification. Note that the data available through the hypergraphs at levels 1,2 & 3 also serve as input to tools for process simulation, synthesis, and design. After the solution of any problem, the results are added to the corresponding hypergraph level so they can serve as a repository of all information of a specific process flowsheet. That is, flowsheet alternatives are generated for a given process objective by combining available or generated process knowledge and appropriate computational techniques [79, 82].

4.1 Motivation and framework

4.1.1 Motivation

Process creation involving simulation-based synthesis, design, and analysis could be performed efficiently, reliably, and rapidly through a decomposition of the problem into three stages and a hierarchical work-flow of the calculation steps supported by their corresponding data flow (Tula et al [79], Seider et al. [91]). The three stages are synthesis (including the generation of process flow diagrams), design (including analysis to verify that the base case design delivers the desired products), and innovation (including generation and analysis of alternatives that are better than the base case design). The final process flowsheet, after creation, contains vast amounts of information related to process streams; equipment design and operation parameters; connectivity between operations through streams; various input, output, recycle, and linked streams; efficiency of process equipment; and so on. However, only a subset of this information is required for the tasks related to the synthesis stage, while tasks related to the design stage require additional data for tasks such as fast mass and energy balance calculations with simple models or detailed equipment and operational

parameters for rigorous simulation, or equipment sizing and cost data for economic analysis. An important question is how to provide the necessary data for all stages and also when and how the necessary data could be obtained.

Based on the above discussion, it is desirable to have a universal flowsheet representation that is both efficient in organizing flowsheet information and also mathematically tractable for use in conjunction with existing methods and tools for process synthesis, design, and simulation through consistent and efficient data transfer between system components. That is, a computer-aided system (framework) is needed that combines AI techniques (such as flowsheet hypergraph enumeration, formal flowsheet grammar representation, and process ontologies-based inference) with domain knowledge related to process synthesis ([74, 77]), process design ([75]) and process simulation (Seider et al. [91]) into an intelligent, flexible and consistent framework for process flowsheet representation that can be used for a wide range of problem solutions. The flowsheet representation system should be able to handle different contexts for tasks such as flow-diagram generation, process design, or process simulation. Available domain knowledge and details of the models used at each stage provide the list of data that needs to be included in flowsheet representations for each stage.

4.1.2 Framework

The basic structure of a proposed intelligent and versatile framework for multi-level flowsheet representation and its links to tools for the solution of various flowsheet synthesis, design, and simulation problems is presented in Figure 4.1. This framework meets the requirements of process creation as described above in section 2.1. The objective of this framework is to efficiently organize the flowsheet information at three distinct levels plus a base level 0, each capturing increasingly more context about the process. As shown in Figure 1, the flowsheet representation at each level is linked to a set of tools that use the information as input to solve the associated problem. The problem solution results are also added to the corresponding level flowsheet representation.

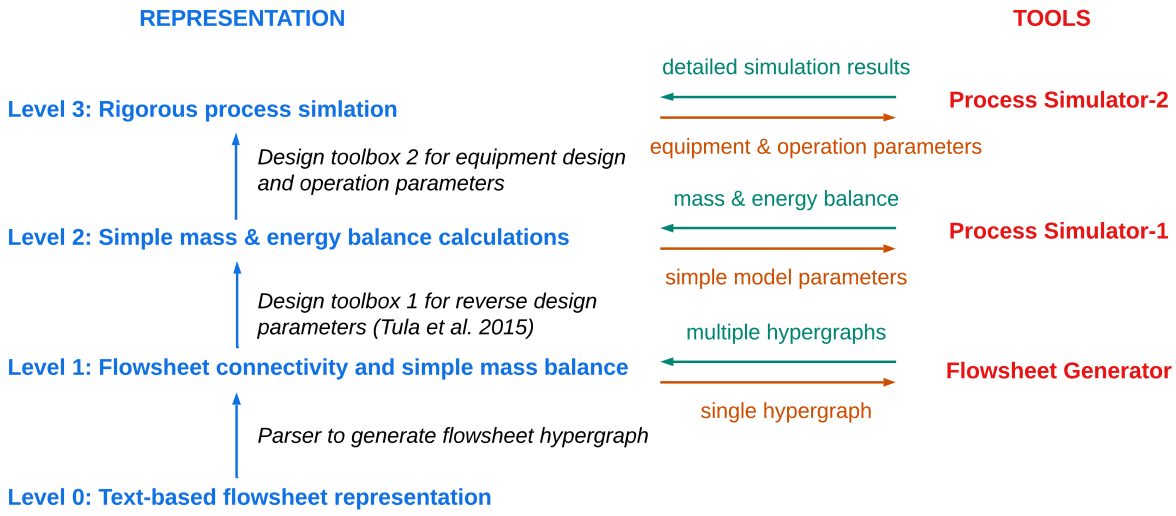


Figure 4.1: An overview of the developed multi-level flowsheet representation and generation framework [92]

The framework highlighted in Figure 1 is based on symbolic formalism (SFILES grammar, ontology) and mathematical objects (hypergraphs) that allow for flowsheet representation at a given level to be integrated directly with existing methods for process synthesis and/or design, thus aiding in the development of hybrid AI models rooted in domain knowledge as listed in Table 4.1. This multi-level framework is called *eSFILES* because it is based on the SFILES representation of the original process flowsheet.

The base level (level 0) requires a concise, text-based flowsheet representation capturing information from process flow diagrams and should be useful for efficient flowsheet storage, sharing, and search. It is proposed to use the concept of SFILES [80, 81] for this level. The goal of level 1 representation is to explicitly capture the flowsheet connectivity information so that options for flowsheet generation can be selected. It is proposed to use graph-theoretic objects such as annotated hypergraphs ([90]) where process streams are represented as nodes and process operations as hyperedges. The flowsheet hypergraphs, owing to their graph-theoretic nature, allow for applications of numeric techniques for model-based simulations, superstructure enumeration, as well as visualization. The objective of level 2 is to include additional information on process operational parameters such as reaction stoichiometry, con-

version, separation factors, and purge factors that are typically used in simple model-based process simulation. The goal of level 3 is to provide design and operation parameters such as reactor operating conditions, distillation column number of stages, and feed stage location; heat exchanger driving forces that are typically used in rigorous model-based process simulation. Using hypergraphs with node-edge annotations allow for a two-way transfer of data between the representation and the corresponding linked tools, as highlighted in Figure 1. The large amount of data in level 3 representation is handled through process ontology that hierarchically organizes design and operations parameters associated with each process operation using class/subclass/properties. Note that as described above, the *eSFILES* framework integrates concepts from graph theory, ontologies, and natural language grammar with well-known methods for process synthesis, design, and simulation.

Table 4.1: Various hybrid AI applications enabled by the developed multi-level flowsheet representation and generation framework

Level	Applications	Numeric ML techniques
Level 0: SFILES strings	text-based flowsheet representation; flowsheet storage, sharing, and search	sequence modeling frameworks; text-mining; pattern identification
Level 1: flowsheet hypergraph	process alternatives	hypergraph-based flowsheet enumeration; validation checks; superstructure generation
Level 2: annotated flowsheet hypergraph	process intensification; flowsheet property prediction	heuristics-based graph traversal; ML and GC-based property prediction
Level 3: hypergraph connected with process ontology	flowsheet simulation and optimization	integration with commercial simulation tools; graphical methods for optimization

4.2 Basic concepts

In this section, an overview of the background methods required for constructing the *eSFILES*-based representation of process flowsheets at various levels of complexity is provided. For consistency and easier comprehension, without loss of generality, the methods and the *eSFILES*-based representation (*eSFILES* framework) are explained using the well-known

hydrodealkylation (HDA) process [75] used for the production of benzene and biphenyl from toluene and hydrogen. The HDA process is used since it has complexity, involving simultaneous reactions, multiple separation tasks and their corresponding separation techniques, and the presence of an inert compound that needs to be purged and recycled of unreacted materials.

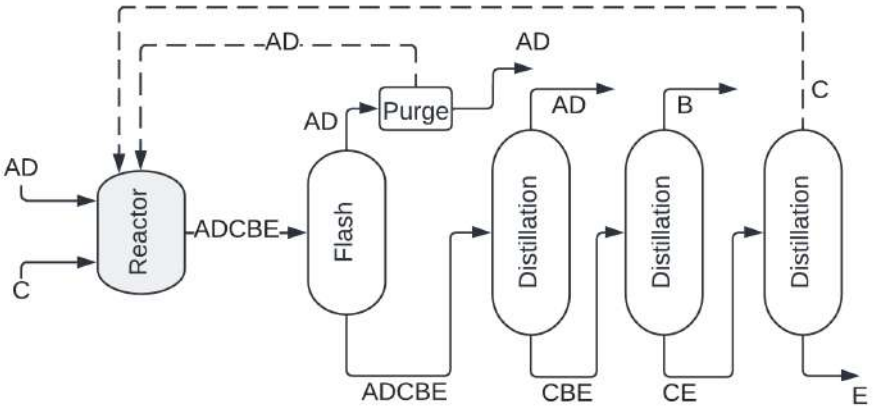


Figure 4.2: The HDA process flow diagram [75]

In the HDA process shown in Figure 4.2, a stream [AD] of hydrogen (A) and methane (D) and a stream [C] of toluene (C) is fed to the reactor where toluene reacts with hydrogen to produce benzene (B) and methane (D). A side reaction results in benzene converting to biphenyl (E) and hydrogen. Throughout this paper, a '[ ]' will represent a stream, and a '( )' will represent one or more chemicals.

$$Toluene\ (C) + H_2\ (A) \longrightarrow Benzene\ (B) + Methane\ (D)$$

$$2\ Benzene\ (B) \rightleftharpoons Biphenyl\ (E) + Hydrogen\ (A)$$

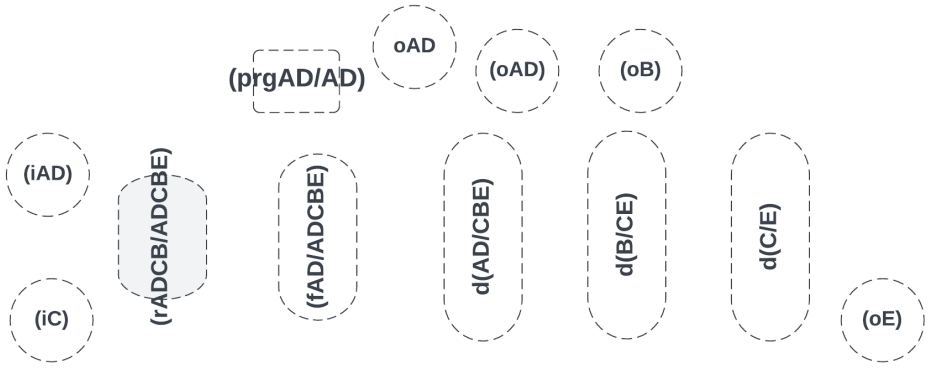
The reactor outlet stream [ADCBE] containing the products, as well as the unconverted reactants and the inert compound, is fed to a flash unit that separates the feed stream into a top product stream [AD] and a bottom product stream [ADCBE]. The product stream [AD] is sent to a purge unit that splits the feed into a purge stream [AD] and a recycled stream [AD].

The bottom product stream of the flash unit is sent to the first unit of a three-distillation column separation train from which the main product stream [B], the side product [E], and the unconverted reactants (AD) and (C) are obtained. The following sections explain the underlying methods associated with the *eSFILES*-based process flowsheet representation.

Remark 1: For the HDA process shown in Figure 4.2, the following two assumptions are made– first, the product streams are represented as a single chemical product. It may include impurities, which are added at the design and simulation levels. Second, in the case of reactants with inert chemicals, the corresponding streams are represented as mixed streams (such as AD, hydrogen, and methane), which are purged and also recycled.

4.2.1 *eSFILES*-based flowsheet representation

Inspired by the simplified molecular input line entry scheme (or SMILES) representation [14], an analogous text-based representation for flowsheets was developed by [80, 82, 81]. The simplified flowsheet-input line-entry scheme (or SFILES) represents flow diagrams containing streams and unit operations as a single-line text representation. This representation involves representing the various unit operations as 'process-atoms' analogous to atoms in a molecular representation, the linked streams as 'process-bonds' analogous to bonds linking atoms in molecular representations, and the various input, output, and recycle streams as process-bonds with special symbols. The same version of SMILES, as proposed by [80, 82], is primarily worked with, but it is extended to include additional process-atoms such as dividers, purge, and recycle.For the HDA process shown in Figure 4.2, the corresponding process-atoms signifying different unit operations in the process flow diagrams are shown in Figure 4.3a and the corresponding text-based SFILES representation is shown in Figure 4.3b.



(a) Illustration of the HDA process flowsheet represented by process-atoms and process-bonds

$$(iAD)(rADCB/ADCBE) < 1 < 2[(< iC)](fAD/ADCBE)[(prgAD/AD)1(oAD)]- \\ -(dAD/CBE)[(oAD)](dB/CE)[(oB)](dC/E)2(oE)$$

(b) Text-based SFILES representation for the HDA process

The various aspects of the SFILES notation and associated symbols are explained below:

- Similar to SMILES, the SFILES strings are read from left to right
- Process-atoms representing unit operations for specific tasks (reactor, distillation, divider, etc.) in the flowsheet are delimited by parenthesis. Hence, the process-atoms in Figure 4.3a are represented as
  - reactor (rADC/ADCBE): 'r' indicates the process-atom type, 'ADC' indicates the inlet stream containing the compounds, A, D (inert) and C, '/' symbol to separate reactants from products, 'ADCBE' indicates the reactor effluent stream. Note the reactor effluent 'ADCBE' is a linked process-bond that connects the reactor to its first downstream process-atom.
  - flash (fAD/ADCBE): 'f' indicates flash, 'AD' before the symbol '/' indicates the top product, and 'ADCBE' indicates the bottom product. Note that the input

stream for the flash is a linked process-bond, for example, the reactor effluent connected to the flash unit.

  - distillation (dAD/CBE): 'd' indicates distillation, 'AD' before the symbol '/' indicates the top product, and 'CBE' indicates the bottom product. Note that, as in the flash unit, the input stream for the distillation is also a linked process-bond, for example, the bottom product stream linking the flash and the distillation.

  - purge or divider (prgAD/AD): 'prg' indicates a purge unit, and the two divided streams are [AD] with different symbols, 'o' indicating a product (purge) stream and a '1' indicating recycle stream 1.

- Two consecutive process-atoms represent a connection (process-bond) from the first process-atom to the second process-group. For instance, (iAD)(rADCB/ADCBE) represents an inlet process-atom connected to a reactor process-atom.

- Branches are represented by square brackets ('[', ']') and the '<' symbol is used to specify a recycle stream. Thus, [(<iC)] indicates that the process-atom (iC) is connected as *inlet* to the reactor process-atom (instead of the outlet).

- Recycles in the flowsheet are represented by numbers, one for each recycle loop present. For example, consider (prgAD/AD)1(oAD) in the above example. The number 1 here indicates that an outlet of the purge process-atom (prgAD/AD) is connected to the inlet of the reactor process-atom (rADCB/ADCBE) (due to the presence of '<1' after the reactor), whereas the other outlet is connected to (oAD) process-atom indicating it exits the flowsheet. Intelligence is added to this level so that reactants after a reaction is identified and marked for possible recycling.

Additional details and examples are given in references [80, 82], while Section 4.2.2 gives the specially developed SFILES grammar that formalizes the notation using formal syntax rules.

4.2.2 *SFILES* grammar syntax rules

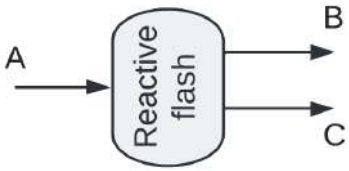
Since the SFILES representation is a novel and alternate way of representing process flow diagrams, this representation system is formalized with a specially developed SFILES grammar (similar to SMILES [14]) based on grammar formally defined by Chomsky [55], which consists of the following elements:

- S, a designated start symbol
- Σ, the set of terminal symbols
- N, the set of non-terminal (intermediate) symbols
- and R, the set of syntax rules of the form $A \longrightarrow \beta$ where $A \in N$ is non-terminal and $\beta \in \Sigma$ is a terminal symbol

The SFILES grammar syntax rules are, therefore, a formal description of the SFILES-based flow-diagram representation system and could be used for generating hierarchical trees that represent the grammatical structure of SFILES strings. These SFILES grammar trees could then be used to automatically infer process context such as – process-atoms (unit operations), process-bonds (input, output, recycle, intermediate), and associated connectivity between process-atoms and process-bonds. Moreover, the SFILES grammar syntax rules are also useful for identifying syntactically invalid SFILES strings – for a given SFILES string, if the corresponding grammar tree could not be generated, then the given SFILES string has incorrect syntax (just like invalid English sentences do not have a correct grammatical structure). Therefore, the SFILES grammar is developed with the primary purpose of automatically converting SFILES string to their connected hypergraph representation (see Section 4.2.3) in a process context-aware manner.

For instance, consider a simplified process flow diagram (see Figure 4.4a) with an inlet stream, a reactive flash unit where complete conversion of reactant A to products B and C takes place, and two outlet streams (with products B and C), represented by the SFILES

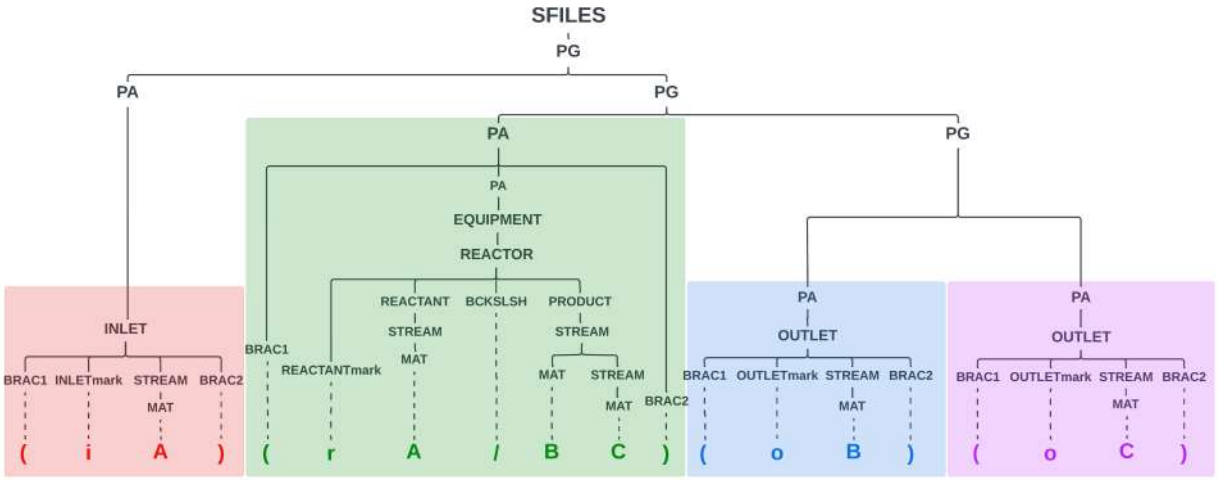
notation in Figure 4.4b, and the corresponding SFILES grammar tree shown in Figure 4.4c. Additional examples of grammar trees of process flow diagrams are given in Appendix D.



(a) Illustration of a reactive flash process flow diagram

(iA)(rA/BC)(oB)(oC)

(b) Text-based SFILES representation for the reactive flash process shown above



(c) Hierarchical grammar tree corresponding to the SFILES string obtained using grammar syntax rules in Table 4.2 (different process-atoms are highlighted in different colors)

This top-down, hierarchical grammar tree shows the grammatical structure for representing the SFILES string characters `{ (, i, A, ), r, A, /, B, C, ), (, o, B, ), (, o, C, )}` as the leaf nodes (or terminal nodes) of the grammar tree. The top node (`SFILES`) indicates that the grammar tree corresponds to the SFILES representation of the given process flow diagram. The intermediate (nonterminal) nodes `{ PG, PA, INLET, EQUIPMENT, REACTOR, OUTLET, ...}` represent the broad category of symbols used to represent various sub-components of the SFILES string (just like in English language noun phrase, verb phrase, adjective, etc.

represent various sub-components of a sentence). For instance, PG represents a process group that consists of one or more process-atoms. A process-atom PA could be an INLET representing flowsheet input streams, OUTLET representing flowsheet output streams, EQUIPMENT representing process equipment such as a REACTOR, and so on for other components in a process flow diagram. The grammar syntax rules listed in Table 4.2 define how they could be applied to an intermediate symbol (left-hand side of the rule) to subdivide (or transform) it into a new set of symbols (intermediate or terminal) on the right-hand side. The equivalent symbols characterizing the grammar syntax rules, along with their meaning, are listed in Table 4.3. A more comprehensive list of developed SFILES grammar syntax rules, but not a complete set, is given in Appendix C.

In the grammar tree, for example, the one shown in Figure 4.4c, the top node SFILES is subdivided using grammar syntax rule R_1 : SFILES $\longrightarrow$ PG, into PG. On PG, the grammar rule R_2 : PG $\longrightarrow$ PA PG is applied to create two horizontal branches with PA as the left intermediate node and PG as the right intermediate node. On the left non-terminal node PA, grammar rule R_6 : PA $\longrightarrow$ INLET is applied, and further, on INLET grammar rule R_8 : INLET $\longrightarrow$ BRAC1 INLETmark STREAM BRAC2 is applied to generate four intermediate nodes that will be used for representing a flowsheet inlet stream. Finally, on INLETmark, grammar rule R_{24} : INLETmark $\longrightarrow$ 'i' is applied to get the terminal symbol 'i'; on BRAC1 rule R_{20} : BRAC1 $\longrightarrow$ '(' is applied to get the terminal symbol '('; on STREAM rules R_{15} : STREAM $\longrightarrow$ MAT and R_{17} : MAT $\longrightarrow$ 'A' are applied to get the terminal symbol 'A' representing the material in the input stream; and on BRAC2 rule R_{21} : BRAC2 $\longrightarrow$ ')' is applied to get the terminal symbol ')'. The end-point (leaf or terminal node at the bottom level of the tree) is reached when on the right-hand side, only terminal nodes are encountered. In this way, the SFILES grammar syntax rules listed in Table 4.2 could be applied to each of the intermediate symbols until terminal symbols representing characters in the SFILES string are encountered. If there are multiple nodes (intermediate or terminal) that could be used on the right-hand side of a rule, then alternative flow diagrams are generated.

Table 4.2: Example of SFILES grammar. The complete set of rules is listed in Appendix C.

Rule	Grammar rules	Meaning/explanation
R_1	SFILES ⟶ PG	top node transforming into an intermediate node (process group PG)
R_2	PG ⟶ PA PG	process group PG transforming into one or more process-atoms (PA) and more PG
R_3	PG ⟶ PA	PG transforming only to one PA
R_4	PA ⟶ BRAC1 PA BRAC2	PA enclosed in parentheses
R_5	PA ⟶ EQUIPMENT	PA that is a process equipment
R_6	PA ⟶ INLET	PA that is an inlet stream
R_7	PA ⟶ OUTLET	PA that is an outlet stream
R_8	INLET ⟶ BRAC1 INLETmark STREAM BRAC2	flowsheet inlet stream represented as inlet-mark followed by stream process-atom enclosed in parentheses
R_9	OUTLET ⟶ BRAC1 OUTLETmark STREAM BRAC2	flowsheet outlet stream as outlet-mark followed by stream process-atom in parentheses
R_{10}	EQUIPMENT ⟶ REACTOR	equipment that is a reactor
R_{11}	EQUIPMENT ⟶ SEP	equipment that is a separator
R_{12}	REACTOR ⟶ REACTANTmark REACTANT BCKSLSH PRODUCT	reactor as reactant-mark, reactant, backslash, product
R_{13}	REACTANT ⟶ STREAM	reactant stream
R_{14}	PRODUCT ⟶ STREAM	product stream
R_{15}	STREAM ⟶ MAT	stream with a single material
R_{16}	STREAM ⟶ MAT STREAM	a recursive rule that allows a process stream to contain more than one material
R_{17}	MAT ⟶ 'A'	material containing A
R_{18}	MAT ⟶ 'B'	material containing B

Table 4.2: Example of SFILES grammar. The complete set of rules is listed in Appendix C.

Rule	Grammar rules	Meaning/explanation
R_{19}	`MAT` $\longrightarrow$ `'C'`	material containing C
R_{20}	`BRAC1` $\longrightarrow$ `'('`	opening parentheses
R_{21}	`BRAC2` $\longrightarrow$ `')'`	closing parentheses
R_{22}	`BCKSLSH` $\longrightarrow$ `'/'`	backslash symbol
R_{23}	`REACTANTmark` $\longrightarrow$ `'r'`	reactant-mark
R_{24}	`INLETmark` $\longrightarrow$ `'i'`	inlet-mark
R_{25}	`OUTLETmark` $\longrightarrow$ `'o'`	outlet-mark

NOTE: PG here refers to process-groups, a combination of one or more process-atoms (PAs) and has been introduced to enforce a hierarchy in the grammar rules.

Table 4.3: List of symbols in the grammar rules listed in Table 4.2

Type	Grammar symbols	Description
S	SFILES	the top node in the SFILES grammar tree representing SFILES flowsheet representation.
Σ	A, B, C, <, r, /, (,), i, o	Leaf (or terminal) nodes in the SFILES grammar tree representing the list of valid symbols in the SFILES vocabulary.
N	PG, INLET, OUTLET, EQUIPMENT, REACTOR, REACTANT, PRODUCT, STREAM, MAT	The intermediate (or non-terminal) nodes in the SFILES grammar tree that represent a broad categorization of subcomponents in the SFILES strings. For instance, `'PA'` represents all the process-atoms whereas '`REACTOR`' represents all reactor process-atoms.
R	$\{R_i\}_{i=1}^{25}$	set of grammar rules $\{R_i\}_{i=1}^{25}$ listed in each row in Table 4.2. In total, there are 25 distinct grammar syntax rules in the subset of SFILES grammar listed in Table 4.2.

The proposed SFILES grammar allows automated parsing of SFILES strings in databases (or as input flowsheet) to check for syntax validity, inferring structural information embedded in the SFILES representation, inferring connectivity information between process-atoms and various process-bonds (input, output, recycle, intermediate), and to generate alternative but valid process flowsheets by enforcing syntactic constraints. Note here that the grammar syntax rules are developed based on process knowledge underlying process flowsheets factoring in flowsheet context and not merely extracted from purely data-driven pattern extraction methods.

4.2.3 Flowsheet hypergraph

Flowsheet synthesis and design typically involves identifying a set of design variables for which optimal values are determined through, for example, simulation-based optimization techniques [78]. That is, the locations of the design variables on the flowsheet and their values are needed for the process simulation step. The *eSFILES* representation can easily add the necessary information data through the concept of the flowsheet hypergraph. A combination of a process ontology (see Section 4.2.4) for structuring the knowledge related to the design-decision variables in process flowsheets and a method for their contextual mathematical representation is applied. Such a representation should have the ability to capture the connectivity information as well as the associated contextual information between various process-atom-stream pairs. Thus, a process flowsheet is represented as a hypergraph with node-edge annotations where the nodes represent streams, edges (or hyperedges) represent process-atoms, and the node-edge annotations capture relevant contextual information. This is similar to the reaction hypergraph representation proposed by Mann and Venkatasubramanian [90] for studying reaction network statistics. Note that a combination of process-atoms (hyperedges) and associated process-bonds(streams) is referred to as a process-group containing multiple process-atoms. While it is possible to create flowsheet hypergraphs the other way around – nodes representing process-atoms and hyperedges representing process streams

– such a representation would not allow for leveraging the benefits of hypergraphs (hyperedges connecting multiple nodes together) to the fullest extent. This is because flowsheets where the exact same stream (hyperedge) enters multiple process-atoms (nodes) at the same time are not usually encountered, whereas it is often seen that multiple streams (nodes) enter the same process-atom (hyperedge) and hence, the latter representation is chosen for generating flowsheet hypergraphs. Nevertheless, the two representations could be converted to each other using the duality property of hypergraphs [90].

Though flowsheets have been represented as di-graphs in previous works, our work is a new attempt to represent process flowsheets using hypergraphs with annotations. Hypergraphs are more concise representations since a single hyperedge could connect multiple nodes as opposed to a graph that introduces additional edges for connectivity between more than two nodes [90]. The node-edge pair annotations, in addition, offer flexibility in terms of capturing additional information associated with process-atoms-streams pairs.

Mathematically, a hypergraph is a pair $H = (V, E)$ where V is a set of vertices and E is the set of edges (or *hyperedges*) where each edge contains a non-empty subset of V. A hypergraph with annotations is defined as $H = (V, E, X, l)$ where V is the set of vertices, E is a set of edges, X is a finite label set containing the possible set of labels (or annotations/roles), and l is a role labeling function for assigning roles to each hyperedge-node pair as $l(v, e) = x$. For representing directionality, X = {'in,' 'out'} labels could be used since the direction of the flow of materials is important for a chemical flowsheet. For the HDA process and its corresponding hypergraph containing the following

- streams represented as vertices, V = {C, AD, ADCBE, AD, CBE, CE, C, E, B}
- process-atoms represented as hyperedges, E, connecting input and output streams, E = {R1, F1, D1, D2, D3}
- node-edge role labels, {'in', 'out'} shown as directed arrows for brevity

is shown in Figure 4.5. The hypergraph representation is a mathematical representation and

allows for each manipulation using numeric methods as opposed to a simple process flow diagram shown in Figure 4.2.

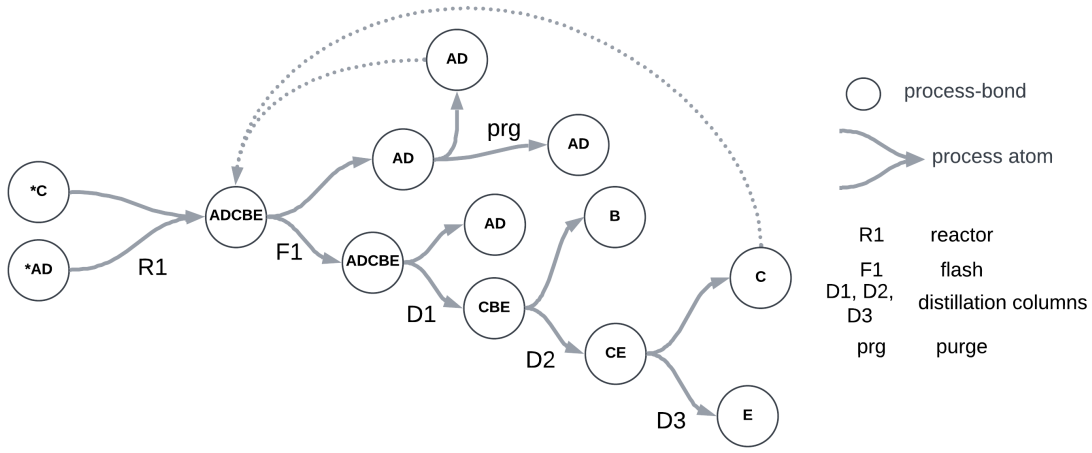


Figure 4.5: HDA process hypergraph with streams as vertices and process-atoms as hyperedges. The asterisk symbol (*) for streams 'AD' and 'C' indicates that they are raw input streams to the process.

4.2.4 Process ontology

To perform a simulation of a process flowsheet, much more information is needed about the process-atoms and streams, such as design and operation parameters, than those given in a typical process flow diagram or its corresponding SFILES string. This information is necessary for representing flowsheets at levels 2 (for simple model-based simulation) and level 3 (for rigorous model-based simulation) within the *eSFILES* framework, whose applications are highlighted in Section 4.3. However, an ontology is required since the design and operation parameters required for process simulation vary across operation and stream types within process flowsheets. Note that an ontology is a formal description of the knowledge and concepts of a domain and is a hierarchical organization of the concepts as class-subclass relationships. That is, it organizes the information hierarchically and semantically, capturing the class-subclass relationships that often characterize process design and operation information.

A snapshot of the developed process ontology is shown in Figure 4.6.

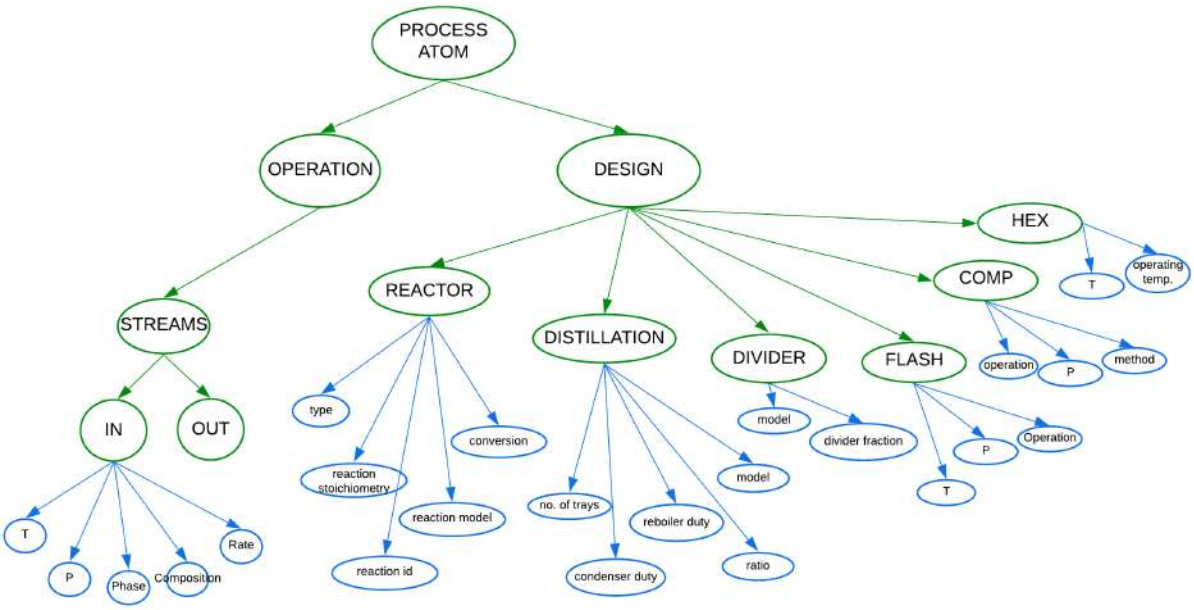


Figure 4.6: Developed process ontology to represent operation and design parameters required for rigorous flowsheet simulation.

In the developed ontology, at the highest level (main class) resides the 'process-atom', which maps uniquely to one of the edges in the flowsheet hypergraph shown in Figure 4.5.

- The 'process-atom' class has two sub-classes, 'operation' and 'design,' corresponding to operation and design information, respectively, for the given process-atom.
- The 'operation' has 'stream' as its subclass which further has all the incoming and outgoing streams as sub-classes along with their material and energy balance parameters stored as properties (indicated with blue arrows in Figure 4.6).
- On the other hand, the 'design' class has 'reactor', 'separator' (represents flash, distillation, pervaporation, and so on), 'divider', and 'comp/pump/hex' as sub-classes for storing design information corresponding to the relevant equipment corresponding to 'process-atom'.

- The associated design parameters are linked to these classes through properties (indicated by blue arrows) similar to the 'stream' class under 'operation'.

- Note that after populating the ontology with relevant data (called instances or frames) for each class, the population ontology could be connected with a flowsheet hypergraph by mapping the hyperedges to their respective process-atom classes in the ontology.

For a given chemical process flowsheet, various instances (or frames) of this ontology could be used to represent information required for storing information relevant to the specific process simulation performed. The instantiated ontology is shown for the HDA process in Figure 4.10 in Section 4.3.

4.3 *eSFILES* hierarchical framework: Description and Application

The *eSFILES* representation incorporated into an *eSFILES* hierarchical framework together with its application in flowsheet generation, simulation, and design is presented in this section. An overview of the *eSFILES* hierarchical framework is shown in Figure 4.7 and a schematic showing all levels in detail is shown in Figure **??**. At the bottom level or the base of the *eSFILES* representation resides the text-based representation of flowsheets which is the most concise representation of information in a process flow diagram. The SFILES representation is similar to the SMILES representation for molecules and requires identification of 'process-atoms' and their connectivity information for writing SFILES strings for a given process flowsheet. At level 1, the flowsheet is represented as a hypergraph that shows connectivity between streams and processes explicitly, where the 'process-atoms' are represented as hyperedges and streams are represented as nodes. This hypergraph is generated automatically by first parsing the SFILES string for a process using the SFILES grammar and then performing inference on the SFILES grammar parse tree to identify connectivity patterns and generate a process flowsheet hypergraph. At level 2, additional information on material and energy balance is added to the hypergraph as node-edge annotations, indicat-

ing the appropriate material/energy balance data for each node (stream) in a process (edge), thus making it complete for performing simple process calculations. At the highest level at level 3, all design and operation information necessary for the simulation of the process flowsheet using commercial process simulators like AVEVA Pro II and AspenPlus is added to the hypergraph using a process ontology. Each hyperedge (process-atom) is connected to a corresponding instantiated process ontology containing the necessary information for simulation.

While describing the framework, at each level, descriptions of what constitutes the framework, the relevant problem applications, and information required for traversing across levels is provided. The reader is suggested to refer to the methods and background information in Section 4.2 to gain a clear understanding of the *eSFILES* hierarchical framework and the mathematical techniques underlying the framework.

4.3.1 Flowsheet representation

A brief overview of the *eSFILES* hierarchical framework for flowsheet representation is shown in Figure 4.7.

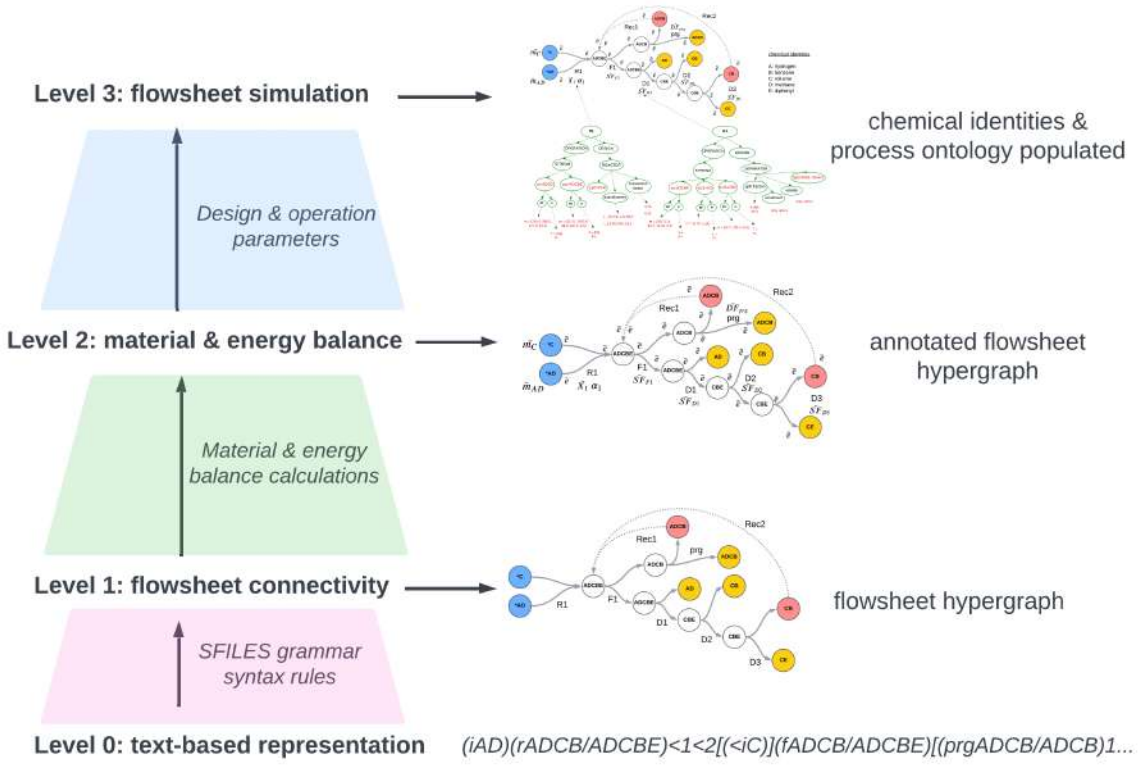


Figure 4.7: The multi-level *eSFILES* representation framework for the HDA process [92]

eSFILES representation at Level 0

At the lowest level, the *eSFILES* representation includes the text-based representation of flow diagrams that contains information on the various process-atoms, streams, materials, and connectivity information embedded in the single-line representation. For the HDA process in Figure 4.2, the *eSFILES* L0 representation is given in Figure 4.3b.

The input and output associated with *eSFILES* level 0 representation are listed in Table 4.4.

Table 4.4: *eSFILES* level 0 representation input and output

Level 0		
Input	process-atoms	connectivity through process-bonds
Output	SFILES strings	

Input The input required for generating *eSFILES* level 0 representation of flowsheets is the identification of process-atoms along with their associated process-bonds and connectivity patterns.

Output The output, which is a purely text-based representation of flowsheets has several uses including efficient information storage, sharing, and retrieval; deep learning frameworks for manipulation of flowsheet strings just like SMILES manipulation for molecular generation; flowsheet property prediction purely based on the underlying process-atoms and process-bonds, and so on.

eSFILES representation at Level 1

Level 1 of the *eSFILES* hierarchical framework includes the flowsheet hypergraph where vertices represent process-bonds (representing process streams), hyperedges represent process-atoms, and the directed connections represent the flowsheet connectivity explicitly. The *eSFILES* level 1 representation performs an explicit grammar check to identify syntactically incorrect process flowsheets; the hypergraph provides a flexible mathematical hypergraph representation that could be manipulated using numerical methods to perform flowsheet enumeration, and superstructure generation, and used in conjunction with machine learning models. In addition, to allow for schematic visualization of the process using the hypergraph, and hence expanding the use cases of *eSFILES* level 1, streams are color-coded to differentiate between various stream types – raw input streams are shown in blue, linked stream are shown without color, recycle streams are shown in yellow, and output streams are shown in red. Moreover, the node and hyperedges are named to indicate the underlying streams (process-bonds) and unit operations (process-atoms), respectively. The *eSFILES* level 1 representation for the HDA process is shown in Figure 4.8 below.

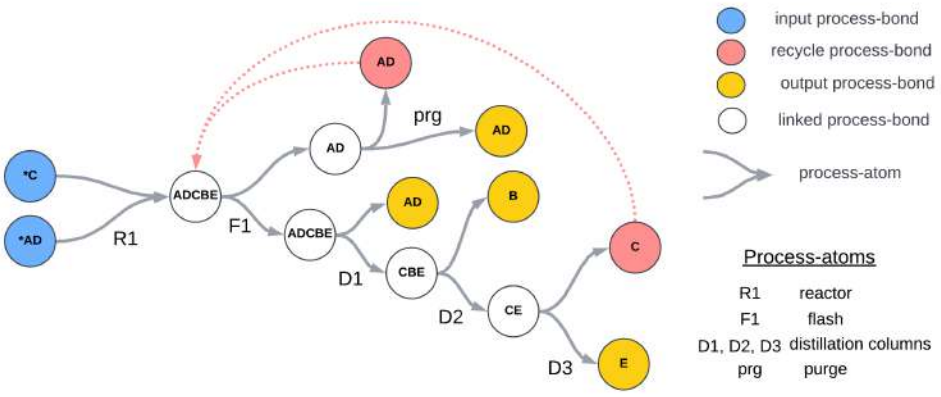


Figure 4.8: *eSFILES* level 1 representation for the HDA process

The input and output associated with *eSFILES* level 1 representation are listed in Table 4.5.

Table 4.5: *eSFILES* level 1 representation input and output

Level 1			
Input	SFILES string	SFILES grammar	Inference algorithms
Task	represent connectivity information		
Output	flowsheet hypergraph		

Input The input required for generating eSFILES level 1 is the eSFILES level 0, inferencing algorithms, and the SFILES grammar syntax rules described in Section 4.2.2 (in Table 4.2).

Output The SFILES grammar rules are used to parse the *eSFILES* level 0 representation to generate a hierarchical organization of the meta information about process-atoms and streams comprising the flowsheet. This hierarchical tree structure combined with connectivity-inferencing algorithms (using grammar parse trees) are then used to generate a flowsheet hypergraph depicting the underlying chemical process.

eSFILES representation at Level 2

At level 2 in the *eSFILES* hierarchical framework resides an annotated hypergraph representation with information necessary for performing mass and energy balance calculations. Recall from Section 4.2.3 that the hypergraph representation allows for node-edge specific labels (or annotations) that allow for storing additional contextual information specific to the process. This flexibility of the developed hypergraph representation is leveraged to store information on material compositions, stream temperature and pressures, reactor conversion, separation factors, divider fraction, and so on. The *eSFILES* level 2 representation for the HDA process in shown in Figure 4.9 below.

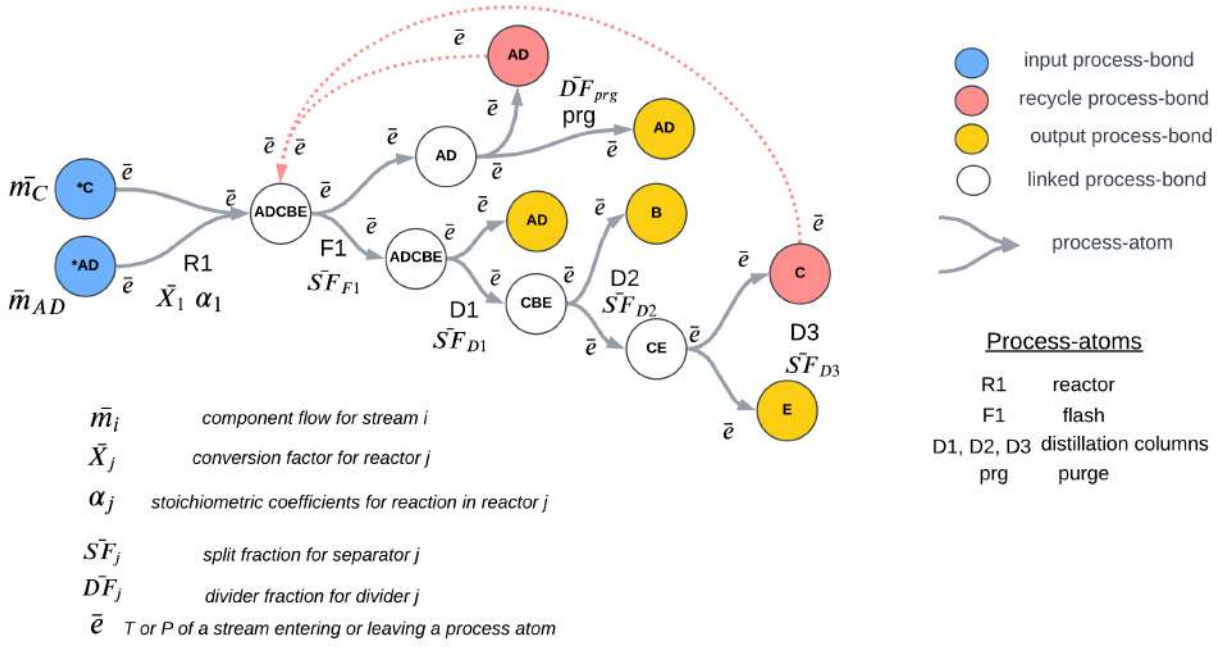


Figure 4.9: *eSFILES* level 2 representation for the HDA process

In the above representation, the symbols $\bar{m}_i, \bar{X}_j, \bar{\alpha}_j, \bar{SF}_j, \bar{DF}_j, \bar{e}$ are vectors of appropriate dimensions based on the number of components in the stream that need to be defined. To perform mass balance calculations using simple models [91, 79], $\bar{m}_i, \bar{X}_j, \bar{\alpha}_j, \bar{SF}_j, \bar{DF}_j$ are the set of variables that are needed since they could be used to compute component flow for each stream. For energy balance, $\bar{e}$ could be tracked along a given hyperedge and if its value

changes along the same hyperedge or vertex, it indicates the presence of a heat exchanger, pump, or compressor based on the change in temperature or pressure variables.

The input and output associated with *eSFILES* level 2 representation are listed in Table 4.6.

Table 4.6: *eSFILES* level 2 representation input and output

Level 2			
Input	flowsheet hypergraph	equipment parameters for simple models	chemical identities
Task	perform mass balance calculations with simple models; fix Temperatures and Pressures; compute energy demand or release through simple models		
Output	annotated flowsheet hypergraph		

Input The input required for generating *eSFILES* level 2 representation is the information from *eSFILES* level 1 representation, raw input stream compositions, conversion factors for all reactors, stoichiometric coefficient for all reactions, split fraction for all separators, divider fraction for all divider, and finally, the temperature/pressure of streams entering or leaving processes.

Output The output of *eSFILES* level 2 representation is a hypergraph containing data for the operations (process-bonds) so that a quick mass and energy balance can be performed, without requiring detailed design of the process-bonds. The data generated with the additional information at this level is used for the design of the operations in the process flowsheet. It also serves as input for process simulators employing simple models and not requiring rigorous model data.

eSFILES representation at Level 3

Level 3 of the *eSFILES* hierarchical framework involves the addition of necessary information required to perform rigorous simulations with any process simulator, such as Pro II and Aspen. The operational and design parameters associated with process-atom along with

identities of chemicals in each process-bond. The list of variables is taken from a combined in-house table of variables related to operations corresponding to rigorous models found in process simulators [91]. As described in Section 4.2.4, this information is stored for each process-atom using an instantiated ontology where the appropriate classes are instantiated with their values. The *eSFILES* level 3 representation for the HDA process is shown in Figure 4.10 below.

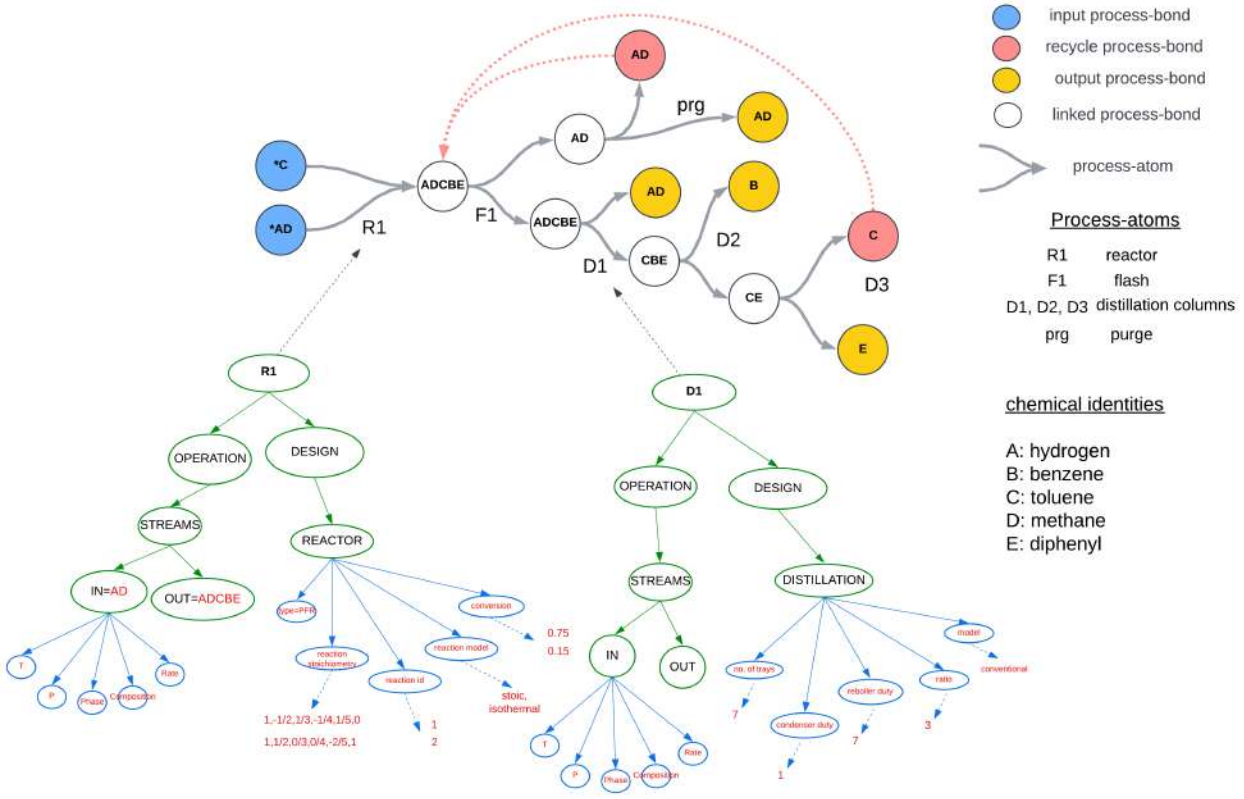


Figure 4.10: *eSFILES* level 3 representation for the HDA process

While the process ontology and the instantiated values are shown with a reasonable degree of detail, the ontology could be easily expanded to include additional details from process simulator keyword input files, mapping of the process-atoms and associated variables across various different simulators, or further categorization of the classes to capture detailed information. Given the flexibility and modular nature of the ontology, new classes could be easily added and linked to process-atoms in the hypergraph to depict any information necessary for any given application of the chemical process flowsheet.

Also, note that the hypergraph does not explicitly show heat exchangers and pumps explicitly since their location in the flowsheet could be inferred based on the simple energy balance information from *eSFILES* level 2 in Figure 4.9. Since they do not affect material balance calculations, if the energy parameters of a stream change, it automatically implies the presence of a heat exchanger (if enthalpy changed) or a pump (if pressure changed).

The input and output associated with *eSFILES* level 3 representation are listed in Table 4.7.

Table 4.7: *eSFILES* level 3 representation input and output

Level 3			
Input	annotated flowsheet hypergraph	design and operation parameters	process ontology
Task	perform rigorous flowsheet simulations		
Output	annotated flowsheet hypergraph connected to process ontology		

Input The input required for generating *eSFILES* level 3 are the information from *eSFILES* level 2 combined with a process ontology providing skeleton knowledge organization, design, and operational parameters for each process-atom, and the chemical identities indicating chemicals underlying each stream.

Output The output of *eSFILES* level 3 includes data needed to perform rigorous process simulation, perform rigorous process intensification, flowsheet ranking, flowsheet enumeration, generation of alternative candidates, and many more using appropriate models of the process. Such problems could be solved efficiently using a combination of process knowledge and machine learning or AI-based numerical techniques.

4.4 Conclusions

To address the drawbacks of purely data-driven approaches for process flowsheet synthesis and design, a novel hybrid AI framework called an extended, simplified, flowsheet-input

line-entry scheme (*eSFILES*) is proposed. This representation involves organizing flowsheet information using a hierarchical framework consisting of four levels, with Level 0 representing the flowsheet as purely text-based strings. Level 1 represents flowsheet connectivity as hypergraphs with process streams represented as nodes and unit operations represented as hyperedges. Level 2 includes process parameters required for performing simple mass balance calculations using node-edge annotations in the hypergraph. Finally, Level 3 uses a process ontology to capture detailed design and operation parameters required for rigorous flowsheet simulation using commercial software.

To facilitate the correct processing of text-based SFILES representation (i.e., Level 0 of the *eSFILES* representation), a flowsheet grammar is also developed for extracting relevant process context and checking for syntax validity of the SFILES strings. The *eSFILES* framework for flowsheet representation is also suitable for performing flowsheet synthesis and design using hybrid AI methods by combining domain knowledge and machine learning. The demonstrated case studies show the framework's ability to efficiently capture process context in the *eSFILES* representation, which is then used to generate flowsheets and perform rigorous simulations using the multi-level *eSFILES* flowsheet generation framework.

The *eSFILES* framework for process synthesis and design incorporates a combination of artificial intelligence-based methods and well-known chemical engineering knowledge incorporated through an intelligent system facilitating fast, correct, and consistent decision-making related to process synthesis and design. It is envisioned that this representation would aid in the wider adoption of hybrid AI-based models due to its efficient organization of relevant process information, an underlying mathematical framework allowing easy integration with existing methods, diverse flowsheet representation formats (text-based, graph-based, ontology-based), and the ability to allow for process constraints. The *eSFILES* framework enables the development of hybrid AI-based systems by combining the multi-level hypergraph representation with already known flowsheet synthesis, design and analysis methods to enable reliable, consistent and efficient solution of various process engineering problems

involving process simulation based synthesis, design and analysis along with other machine learning based specific applications listed in Table 4.1.

In future, we plan to extend this approach to other applications such as generating process alternatives, flowsheet property prediction, process intensification, process control specifications, process safety evaluation, and many more. Going forward, one of our goals is to extend this framework to higher levels where level 4 would include the process control aspects and level 5 would include process safety, sustainability, and environmental impact factors. A chemical safety analysis tool (ChemSub) has been already developed for inclusion within the framework at level 4 to provide the information on the hazardous effects of dangerous chemicals.

Chapter 5: Ontology-based Pharmaceutical Information Extraction

As the biomedical literature is growing by over a million publications each year [93], developing an efficient, automated information extraction framework is required. An automated information extraction framework that uncovers rich information buried in thousands of unstructured documents sitting idle in data repositories would accelerate the drug development cycle. This would lead to quicker, efficient, and objective analysis of new drug applications, better historical data search capabilities, efficient trend extraction from unstructured pharmaceutical text, and easier regulatory monitoring by integrating compliance documents in FDA's risk-based selection model for prioritizing inspections [94]. Such a framework is inline with the U.S. Food and Drug Administration's (FDA) new pharmaceutical quality initiative, knowledge-aided assessment and structured application (KASA) [95], based on formalizing drug product knowledge-base and knowledge-assessment using a structured approach.

Information extraction primarily involves automatically identifying specific domain-related information using the semantics and grammatical structure of text. This broadly comprises of two steps – identifying entities signifying specific information of interest (known as named entity recognition or NER), and inferring the relationships between identified entities (known as relation extraction or RE). Approaches underlying these tasks utilize grammatical structures to extract semantic frames [96, 97], utilizing knowledge bases for relation inference [98], syntactic dependency parsing-based information extraction [99], semantic role labeling [100], coreference resolution [101], and so on. There have been prior attempts in the pharmaceutical domain to use natural language-based methods for information extraction with various applications. Viswanath et al. [102] developed a data-driven tool called IMDP for rapidly developing a new drug briefing document. The methods underlying their approach are calibrated quantum mesh [103], image processing, and search capability com-

bined together to address various needs required for efficient document creation. Kang et al. [104] developed an open-source information extraction framework called ElilE for parsing and formalizing free-text clinical research eligibility criteria. Yuan et a. [105] proposed an information extraction pipeline to transform unstructured text-based eligibility criteria into structured representation as shareable clinical data queries. Xu et al. [106] developed MedEx, a system for medication information extraction from clinical notes and utilized NLP-tools like semantic tagger and parser for generating structured information/data. Harmata et al. [107] reported a semi-automatic approach for information extraction for legal and regulatory applications. Gentile et al. [108] developed a knowledge graph-based pipeline for semantic information extraction from medical package inserts. Such graph-based representations capture semantic information and are ubiquitous for representing proteins [109], chemical reactions [110, 90], process flowsheets [92, 84], and many more applications of graph-based data mining as presented in [111]. Skeppstedt et al. [112] reported a conditional random field to recognize clinical entities in health records. Moreover, given the challenges posed by biomedical text containing chemical names, symbols, tables, equations, technical terms, and so on, several adapted named entity recognition approaches have been reported for identifying biomedical terms including proteins, genes, and disease names [113, 114, 115, 116, 117]. A detailed description of prior works across various applications are provided in the excellent review by Bhatnagar et al. [118].

Despite previous works, the challenges in extracting end-to-end relevant important information from pharmaceutical documents still remain largely unaddressed. First, there is a need to systematically define the *important information* from pharmaceutical documents before building a system that recognizes and extracts it automatically. For instance, the crucial information would not just be all the named entities (drugs, chemicals, etc.) but also the necessary conditions, dosage, level of impurities, packaging type, risk factors involved, manufacturing processes, additional contextual information, and so on. Moreover, this information should be custom-defined by the user based on their downstream applications that

might vary significantly at various stages of the drug development process. Due to this, the standard annotated datasets with entities of a given type (disease names, chemicals, phenotypes) and standard defined relations [119, 120, 121] could not be used for such generalized information extractions. Second, as a results of these, general-purpose labeled datasets are not available for a customized and flexible information extraction problem that could be used for training ML-based classifiers for identifying important information and benchmark their performance. Hence, an unsupervised learning framework is necessary to mitigate these issues and extract the relevant information from pharmaceutical documents automatically in a dataset-agnostic and domain-specific manner. Moreover, due to the lack of availability of such labeled datasets and the need for a flexible, customizable information extraction framework, large language models (LLMs) like GPT-3 [122] are not suitable for a domain-informed pharmaceutical information extraction. LLMs are primarily trained on non-technical English language text, are characterized by a lack of transparency, and are not flexible in capturing domain knowledge. Further, the associated challenges with generative AI-based models such as 'hallucinations' due to their auto-regressive nature along with other limitations highlighted in [123] make them unsuitable for pharmaceutical information extraction.

Here, a Schema (i.e., ontology)-based Unsupervised Semantic Information Extraction (or SUSIE) framework is proposed which is a custom pharmaceutical ontology-based weak supervision framework for generalized, end-to-end information extraction from unstructured pharmaceutical documents. The underlying framework for SUSIE is based on (i) a custom-built pharmaceutical drug development and manufacturing ontology that organizes information underlying the drug development process; (ii) a weak supervision framework with labeling functions and unified medical language system (UMLS) database [124] inspired by clinical entity classification work of Fries et al. [125]; (iii) natural language dependency structure-based contextual information extraction combined with custom rules for processing various data formats; (iv) a fine-tuned BioBERT language model [126] trained on relevant pharmaceutical documents; and (v) a relation extraction module based on linguistic structure [127] to extract

semantic triples (subject, relation, object) from text. These triples are combined with extracted information to generate knowledge graphs representation of important information. The developed framework could be used for performing automated relevant information extraction from pharmaceutical documents in a domain-specific and context-aware manner and is adaptable to other domains.

5.1 Problem statement and objectives

Given an unstructured text document which possibly contains information relevant for drug discovery, development, and manufacturing at any stage – from research on drug molecule to the final drug product packaging – the objective is to automatically identify important information across the document, identify associated relationships between them, and represent them in a structured manner as a knowledge graph. Thus, the input is a document, D_i containing a sequence of sentences, $\{S_j\}$ where each sentence is a sequence of words, $(w_{j,t})$, and the objective is to identify collection of information chunks relevant for drug development from each sentence in the document, ϵ_j, and their associated relations. Formally, consider a document D_i comprising n sentences,

$$D_i = \{S_j\}_{j=1}^{n}; \qquad S_j = (w_{j,1}, w_{j,2}, \ldots, w_{j,t}) \tag{5.1}$$

where S_j is the j^{th} sentence comprising a sequence of t words. The relevant information contained in the document $\mathbf{\Gamma}(D_i)$ is given as,

$$\mathbf{\Gamma}(D_i) = \{\epsilon_j\}_{j=1}^{n} = \{\{\delta_{j,k}\}_{k=1}^{p}\}_{j=1}^{n}; \qquad \epsilon_j = \{\delta_{j,1}, \delta_{j,2}, \ldots, \delta_{j,p}\}; \qquad \delta_{j,k} = (w_{j,m}, w_{j,m+1}, \ldots, w_{j,n}) \tag{5.2}$$

where ϵ_j is the collection of p information chunks in the j^{th} sentence and $\delta_{j,k}$ is a contiguous sequence of important words between m^{th} and n^{th} positions in the sentence. Therefore, each sentence S_j is characterized by ϵ_j, the ordered collection of sets of important words $\delta_{j,k}$.

Our objective is to identify such chunks of information from each document D_i that contains unlabeled sentences with no prior information on $\delta_{j,k}$. This is equivalent to learning the transformation function $\mathbf{\Gamma}(.)$ that operates on a document D_i to give $\{\epsilon_j\}_{j=1}^{n}$ as shown in Equation 5.2.

For instance, consider the sentence,

> *'The cell growth is assessed by a staining method using a tetrazolium salt which is converted by cellular dehydrogenases to a colored formazan product'*

The important information chunks (or groups of words) and relations in this sentence would be,

$\delta_{1,1} = (w_{1,2}, w_{1,3})$ = 'cell growth' $\quad \delta_{1,2} = (w_{1,8}, w_{1,9})$ = 'staining method'

$\delta_{1,3} = (w_{1,12}, w_{1,13})$ = 'tetrazolium salt' $\quad \delta_{1,4} = (w_{1,18}, w_{1,19})$ = 'cellular dehydrogenases'

$\delta_{1,5} = (w_{1,22}, w_{1,23}, w_{1,24})$ = 'colored formazan product'

$\delta_{1,1} \xrightarrow{\text{assessed by}} \delta_{1,2} \quad \delta_{1,2} \xrightarrow{\text{using}} \delta_{1,3} \quad \delta_{1,3} \xrightarrow{\text{converted by}} \delta_{1,4} \quad \delta_{1,3} \xrightarrow{\text{converted to}} \delta_{1,5}$

Thus, the relevant information contained in the sentence S_1 is given by $\epsilon_1 = \{\delta_{1,1}, \delta_{1,2}, \delta_{1,3}, \delta_{1,4}, \delta_{1,5}\}$ and the associated relations shown as a knowledge graph in Figure 5.1.

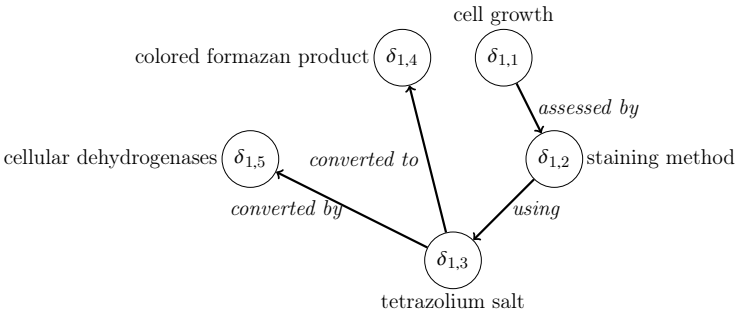


Figure 5.1: Important information representation as knowledge graph with words/phrases as words and relations between them as directed edges.

The underlying methodology characterizing the various modules in SUSIE required to

automatically generate such structured representations of important information from unstructured text are described in detail in the next section.

5.2 Methodology

The transformation function $\mathbf{\Gamma}(.)$ in Equation 5.2 is characterized by a combination of strategies that underlies our information extraction framework as shown in Figure 5.2.

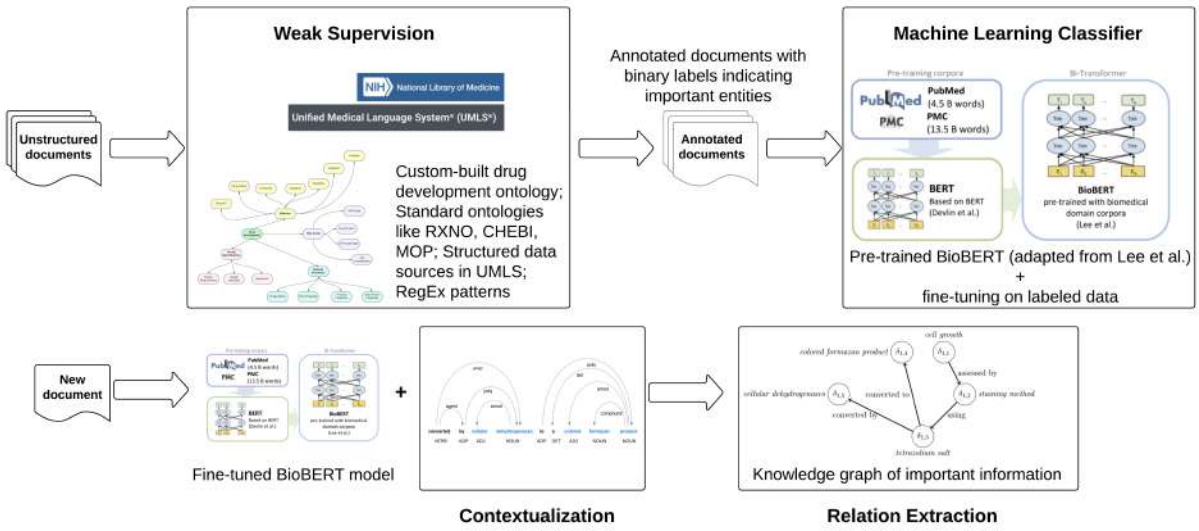


Figure 5.2: An overview of the end-to-end information extraction framework based on weak supervision, BioBERT model, contextualization, and relation extraction to generate knowledge graphs. The BioBERT schematic is adapted from Lee et al. [126].

The developed framework underlying SUSIE consists of four major components –

- first, ontologies, structured data sources, and custom rules for weak supervision, an approach in machine learning that sidesteps the requirement of having manually labeled datasets by using structured (but often noisier) data sources for assigning labels to unlabeled datasets in an automated manner. This requires a custom-built drug development ontology developed as part of our work.

- second, a BioBERT-based binary classifier that learns a functional mapping to identify important information from text and improves generalization of the approach by

learning patterns characterizing important information during the training stage. This model could then be used to identify all important words given an input text.

- third, post-processors that capture semantics and contextual information about the identified entities by utilizing the grammatical structure of sentences. The contextual information complements the important words identified in the previous step.

- and fourth, a linguistics-based relations extractor that extracts relationships between identified entities as semantic triples which are then used to generate a knowledge graph representing important information. The knowledge graph combines all important information along with relations extracted from unstructured text.

Further details on these sub-frameworks characterizing our approach are provided in the subsequent sections.

5.2.1 Pharmaceutical CMC ontology development

The goal of pharmaceutical CMC (Chemistry Manufacturing and Control) development is to design a drug product meeting patient needs and develop a safe, robust, environmentally friendly drug substance and drug product manufacturing process that delivers the drug product with desired quality attributes. CMC development involves a sequence of decisions on selecting activities and their timing to manage and mitigate risk to meeting design requirements through the development cycle [128]. The activities that are selected to be performed in turn generate information on materials, processes, and properties that reduce risks over time. The high level interplay of decisions, activities, risks and their relation to information generated has been discussed in detail in literature [128].

The CMC ontology that is the focus in this work is a more detailed version of middle level ontology [128] that seeks to capture the key information generated during development. The information is organized hierarchically in this ontology using class-subclass relationships combined with object and data properties [129, 130, 131]. The ontology was developed through

various discussions with experts in the pharmaceutical industry across various functions. A snapshot of the developed ontology is shown in Figure 5.3 below.

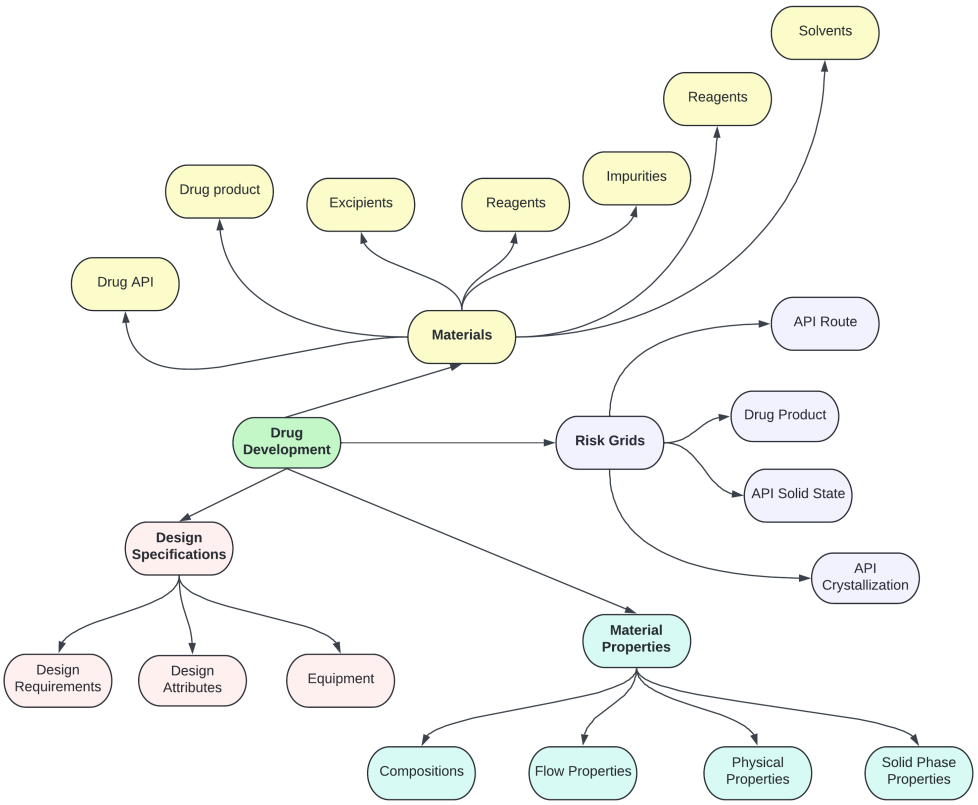


Figure 5.3: A representative snapshot of the custom-built drug development ontology

The final ontology comprised 1277 classes, 3268 axioms, 34 object properties, and 113 data properties and was developed using the Protége software [132] in .owl format, a standard ontology development and sharing format. The central class in the ontology is named Drug Manufacturing which represents information of various types and at different stages associated with drug manufacturing and development. The other classes corresponding to each information type is associated with the central class through relations (or properties). While there are several different classes in the ontology, details on the major classes are provided below.

Design specifications and risk grids There are separate risk grids for different components of manufacturing process and drug product. Figure 5.4 shows risk grid for API synthesis route, API crystallization, Drug Product, API solid state as sub-classes of the risk grid class. The risk grids capture risks associated with various stages and conditions of the drug development process including risk associated with API route, crystallization, solid state, or drug product. The level of risk (low, medium, high) is defined based on several categorical and numerical factors defined in the design specifications and design requirements. To capture additional factors contributing to drug development risk, the ontology could be expanded by creating further subclasses that capture the relevant risk factors.

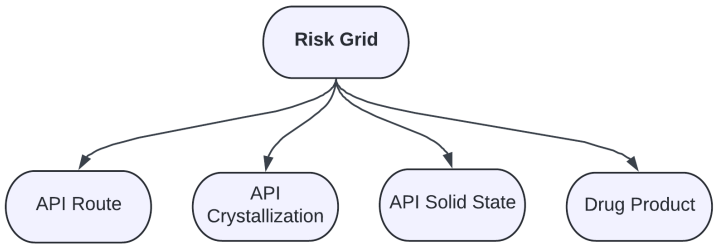


Figure 5.4: Risk grid class and its subclasses

The individual risk grids have a list of design requirements with each design requirement having a set of design attributes whose levels determine the level of risk. For example, for API solid state, a design requirement is to have desirable API properties. A few examples of design attributes that determine desirable API properties are – crystallinity, moisture sensitivity, and stability under different conditions. In the ontology, the design requirement and design attribute are captured as separate sub-classes of the class design specification as this makes it more amenable to instantiation.

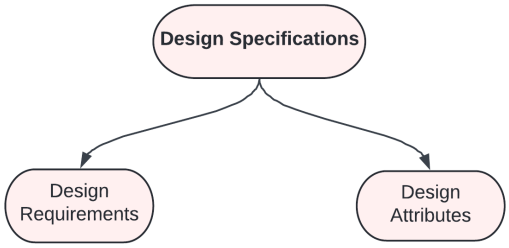


Figure 5.5: Design specifications class and its subclasses

Materials There are different types of material that make up the final packaged drug. A drug that a patient takes is referred to as drug product. The drug product comprises the main active ingredient (referred to as API or Active Pharmaceutical Ingredient) along with excipients. While API is made with desired quality, there are always low levels of impurities (well below acceptable limits for patient consumption) present in the API. The manufacturing process of API involves multiple synthesis steps each of which employ different reagents and solvents in the manufacture. These different material types are captured as sub-classes of the material class as shown in Figure 5.6.

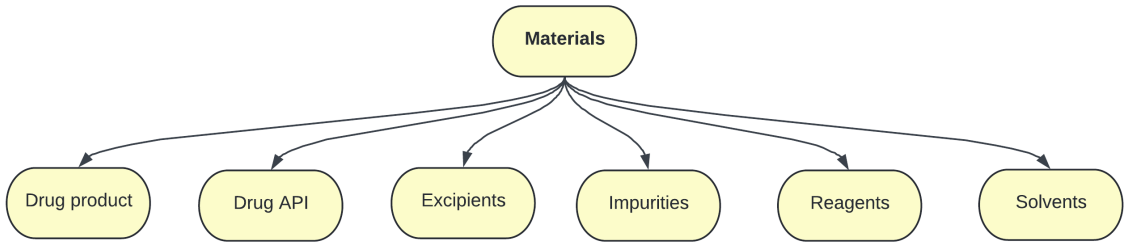


Figure 5.6: Materials class and its subclasses

The ontology also lists the solvents and reagents that are commonly used in most small molecule synthesis. The specific solvents and specific reagents are sub-classes of the class solvents and reagents, respectively.

Material properties Each material that is made through the synthesis whether it is an intermediate or is part of the final drug product has material properties associated with

it that determine its quality and suitability for use. The properties are either physical or chemical properties which are sub-classes of the material properties class. Figure5.7 shows a schematic with a few illustrative material properties.

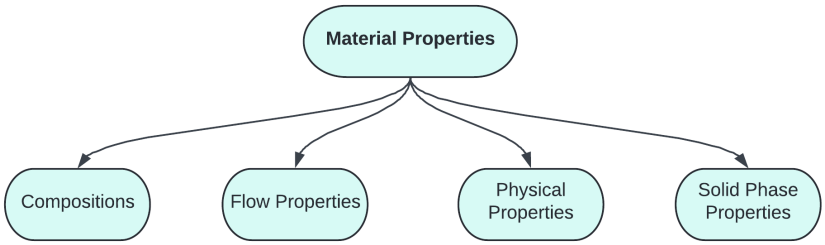


Figure 5.7: Material properties class and its subclasses

5.2.2 Standard ontologies and additional terms

To improve the coverage of the weak supervision task, the developed pharmaceutical drug development ontology (Section 5.2.1) is augmented with other sources. These include, namely – unified medical language system (or UMLS), standard ontologies that are not included in UMLS but are relevant for drug manufacturing, and custom functions based on regular expressions-based patterns to capture information not contained in either of the former two.

Unified medical language system The unified medical language system (or UMLS) is a repository of biomedical vocabularies and integrates millions of names, relations, and concepts from several different sources [124]. The three main components comprising the UMLS are – a metathesaurus containing a repository of interrelated biomedical concepts, Semantic Network mapping metathesaurus concepts to various high level categories, and lexical resources for generating lexical variants of biomedical terms. Though the UMLS ecosystem is vast, the metathesaurus is used extensively and utilize the ontologies that are relevant for our purpose. Therefore, the following subdomains from UMLS were included in our framework – UMLS MeSH ontology containing medical subject headings for documents on biomedical and health related documents; comparative toxicogenomics database (or CTD) chemical on-

tology containing information on environmental chemicals affecting human health and CTD disease ontology with corresponding information on diseases; National Cancer Institute (NCI) terminologies; and Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT) providing the core general terminology for the electronic health record (EHR). The 2021AB version of the UMLS metathesaurus was used in this work.

Chemical names and reaction ontologies The other standard ontologies that are relevant for our purpose include CHEBI (chemical entities of biological interest) [133], RXNO (reaction ontology), and MOP (molecular processes). These ontologies capture various important modules relevant for our work – CHEBI contains information on chemical entities that are of biological interest and thus have high drug-likeness, RXNO contains list of named chemical reactions that often appear in text containing reaction mechanisms, and MOP contains molecular processes that underlie named reactions and are thus a more formal way of referring to steps in chemical reactions.

Additional custom terms In addition, there are certain patterns and nomenclature that are not captured by the various ontologies or knowledge bases given the peculiar nature of their nomenclature. For instance, various pharmaceutical companies (or organizations) would have an internally different way of referencing chemicals based on a unique key or an internally understood nomenclature. To handle such cases, regular expressions that capture a certain pattern in textual data based on a set of predefined rules were used. For appropriate abbreviation detection, which again could be missing from the above discussed structured information sources, the Schwartz Hearst algorithm, a popular algorithm for identifying abbreviations in biomedical text [134], was used.

These sources of structured information in the form of UMLS, standard ontologies, and regular expressions combined with our pharmaceutical drug development and manufacturing ontology provide a rich source of structured information that could be used to perform weak supervision and assign labels to unlabeled dataset.

5.2.3 Text processing

Before performing information extraction, a given unstructured text has to be preprocessed. Such preprocessing usually involves several steps aimed at both text processing as well extraction of additional meta information that could be useful for fine-tuning the NER task later. The following sections provide details on these steps.

Document cleaning Pharmaceutical documents often contain not just text but other types of content such as tables, figures, references, footnotes, page numbers, and reaction mechanisms as equations. Several NLP techniques, both custom built as well as those offered by standard NLP processing Python libraries such as SpaCy [135] and NLTK [136], were used to perform document preprocessing. The standard document cleaning tasks involve – extracting only the document body text and section/subsection headings based on XML map and associated meta-information from the document; identifying section headings/subheadings based on regular expressions-based pattern identification on text meta-information; and identifying table/figure captions using expressions-based pattern identification. While the algorithms are developed assuming input documents as .docx files, they apply to other document types such as PDF since they could easily be converted to .docx format using off-the-shelf software, with the exception of PDFs that are scanned images of text documents where optical character recognition (OCR) techniques might be required.

Natural language processing After standard text-based cleaning, the document is processed using NLP-based approaches that includes, for each word in each sentence in the document – identifying the position of the word in the sentence and stored as an integer indicating the location of the start of the word; lemmatizing the word and stores the lemma of a word, for instance, 'mix' is the lemma for 'mixing'; assigning the part of speech (POS) tag for the word, for instance verb, pronoun, symbol, etc.; and getting the dependency parsing structure for words in a sentence, for instance, subject, modifier, etc. Such grammatical

information is used later in the framework for fine-tuning the approach. More details on these are provided in the later sections.

Document hierarchy and tables extraction In addition to the standard NLP-based document preprocessing, several custom functions were developed. First, a custom function including a document hierarchy extractor was developed that extracts the section and subsection headings by inferring the level of depth for each section heading and associate the corresponding paragraph(s) text for each level. This document hierarchy is useful for processing and analyzing the identified entities, if required, during the downstream tasks such as ontology instantiation.

Second, all the tables from the document were extracted using document processors based on the XML maps. Tables needed to be processed separately from the regular text processing due to the inherent structure present in tables. Since tables are often rich sources of information conveying some very specific information, they were processed separately during the NER stage by treating the table column names as entities. This assumption of table columns containing entities is based on the fact that pharmaceutical documents usually contain information on drug concentration profiles, material composition information, impurity information, and so on.

Chemical names tokenizer pharmaceutical documents containing information on drug development and manufacturing often has drug names being referred to using many different nomenclatures. The drugs and underlying chemicals or impurities are usually referred to using their common names, IUPAC nomenclature, chemical formula, unique identifiers corresponding to standard databases, or internal databases. These nomenclatures often are inherently complex in that they are characterized by comma, hyphens, numbers, and spaces. A standard, natural language-based text tokenizer would therefore make errors while tokenizing words containing such chemical names. Thus, a rules-based chemical names tokenizer, ChemTok [137], was used to tokenize and identify chemical names in the text.

5.2.4 Weak supervision

The hierarchical structure of an ontology could be used as a skeleton for assigning structure to unstructured text in a systematic manner. Ontologies, owing to their underlying hierarchy offer an efficient use-case in weak supervision. Weak supervision is an approach in machine learning that sidesteps the requirement of having manually labeled datasets by using structured (but often noisier) data sources such as ontologies, knowledge-bases, and so on for assigning labels to unlabeled datasets in an automated manner [138], and thus is useful for labeling and generating (synthetic) datasets. Such synthetically labeled datasets could then be used for training powerful ML models that are dependent on labeled datasets. The custom-built ontology was combined with standard structured knowledge-bases and NLP algorithms for providing structure to text in a context-aware manner. The following sections provide details on the various modules underlying our ontology-based weak supervision approach.

In the first step, words were matched in each sentence to various terms in the structured knowledge base comprising the UMLS knowledge base, ontologies, and custom functions. To do this efficiently, Snorkel, a Python-based package for programmatically performing weak supervision and assigning labels to data [139], was used. Performing weak supervision using Snorkel involves writing labeling functions that encode domain knowledge or supervision sources to assign labels to data programmatically. The labeling functions (LFs) used in [125] were adapted including – ontology-based LFs, dictionary-based LFs, task-specific LFs containing regular expressions, and synset-based LFs. Ontology-based LFs include UMLS-based ontologies such as MeSH, Schwartz Hearst for abbreviation detection; dictionary-based LFs included separated LFs for CHEBI, CTD Chemical, CTD Disease, RXNO, MOP, and the custom-built pharmaceutical drug development ontology; task-specific LFs include regular expressions to capture internally used chemicals nomenclature; synset LFs include subject predicate object relationships based on class-subclass relationships extracted from the custom-built pharmaceutical drug development ontology. Each labeling function assigns a

label 1 or 0 based on the defined rule within the labeling function based on term overlap or text-based matching results. Thus, the input to a labeling function is a word and the output would its assigned label of 0 or 1 depending on whether the input word matches the patterns/database of symbols defined in the labeling function.

For instance, given a sentence 'The cell growth is assessed by a staining method using a tetrazolium salt which is converted by cellular dehydrogenases to a colored formazan product', each labeling function would assign a label to each word as shown in Table 5.1 with the various labeling functions (listed across columns) and the corresponding labels for each word (listed across rows). The labeling functions are – $LF_{UMLS\text{-}MSH}$ (based on the MeSH ontology in UMLS), $LF_{CMC\text{-}ontology}$ (based on the custom-built CMC ontology), LF_{CHEBI} (based on the CHEBI ontology), $LF_{CTD\text{-}chemical}$ (based on the CTD chemical names), and LF_{RXNO} (based on the RXNO ontology of named chemical reactions). The final label shown in column 'assigned label' is assigned as

$$label = max(LF_{UMLS\text{-}MSH}, LF_{CMC\text{-}ontology}, LF_{CHEBI}, LF_{CTD\text{-}chemical}, LF_{RXNO})$$

implying that a word is assigned a label of '1' if any of the labeling functions assigns it a label '1'. The labels in the 'assigned label' column are used for training the BioBERT model-based classifier to learn generalized patterns for identifying words. The last column 'after contextualization' indicates word labels assigned after performing contextualization by identifying noun phrases (after identifying labels using BioBERT model on test documents) as described in the next section. The reader is referred to [139, 125] for further details on various types and functional details on labeling functions.

Table 5.1: Example word labeling with various labeling functions. The underlined labels in the last column indicate additional words labeled as important at the contextualization step.

words	$LF_{UMLS\text{-}MSH}$	$LF_{CMC\text{-}ontology}$	LF_{CHEBI}	$LF_{CTD\text{-}chemical}$	LF_{RXNO}	assigned label	after contextualization
The[1]							
cell[2]	1					1	1
growth[3]	1					1	1
is[4]							
assessed[5]							
by[6]							
a[7]							
staining[8]	1					1	1
method[9]	1	1			1	1	1
using[10]							
a[11]							
tetrazolium[12]	1			1		1	1
salt[13]	1	1	1	1		1	1
which[14]							
is[15]							
converted[16]							
by[17]							
cellular[18]							1
dehydrogenases[19]	1					1	1
to[20]							
a[21]							
colored[22]							1
formazan[23]	1			1		1	1
product[24]	1	1				1	1

5.2.5 Learning generalized patterns using fine-tuned BioBERT

Using the weak supervision-based information extraction, a labeled dataset was generated with terms that are of interest (or are important) labeled as 1 and the rest labeled as 0. Since these labels are based on a weak supervision strategy, they are inherently noisy and the labeling strategy not completely generalizable. Thus, to improve the generalization of the transformation function $\mathbf{\Gamma}(.)$, label words based on their context in a sentence and not just in isolation, and improve the framework's ability to handle and classify words unseen during the training stage, a pre-trained BioBERT model was used for fine-tuning on our dataset for the classification task. Fine-tuning involves further training a pre-trained model on customized and often much smaller datasets to achieve optimal performance. The dataset used for such fine-tuning and further details on the fine-tuning strategy are presented in detail in Section 5.3.

5.2.6 Contextualization

It is often necessary to capture the neighboring information around entities since they convey important contextual information. However, the ontology-based weak supervision labeling approach only identifies individual entities but not the neighboring contextual information. To this end, noun phrases in documents were used to capture such information. Noun phrases are words or group words that function like nouns and contains a noun accompanied by modifiers. To identify noun phrases, the dependency structure of a sentence is first identified based on the grammatical structure of the sentence and modifiers associated with nouns. Spacy's pre-trained language model, 'en_core_web_sm' [135], was used to perform dependency parsing and identify the noun phrases. For instance, for the example sentence shown in Table 5.1, a partial dependency parsing diagram between the words is shown in Figure 5.8. Here, the identified entities 'formazan' and 'product' are both part of a noun phrase – 'colored formazan product'. Hence, even though the word 'colored' was not identified as an entity, all words in the phrase 'colored formazan product' are tagged as important information and assigned a label of 1. Thus, the entities are contextualized using such noun phrases-based matching, and the labels are updated to reflect the contextual information as shown in the 'after contextualization' column in Table 5.1.

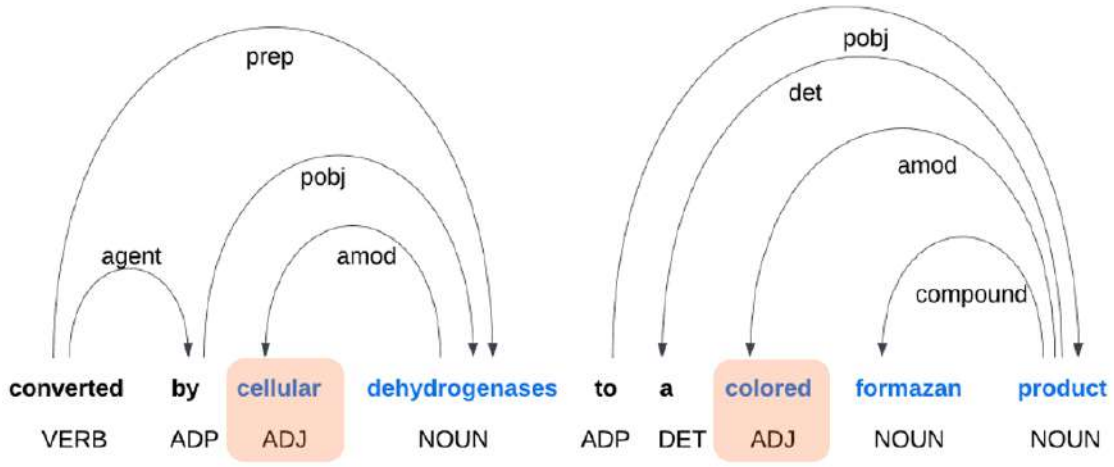


Figure 5.8: A partial dependency parsing diagram for the example sentence in Table 5.1 along with identified noun phrases highlighted in blue. The contextual information captured in each noun phrase is also highlighted.

5.2.7 Relation extraction and auto-generating knowledge graphs

After the important words and their associated context have been identified, an important next step for information extraction is identifying relations between the words. This would allow for better capturing of the meaning of the text as a knowledge graph where the entities (or important words) are represented as nodes in the knowledge graph, and relations between them are represented as directed edges between them. To extract such relations, a linguistic structure-based approach was utilized for extracting relations as semantic triples (subject, predicate, object) where the predicate is the relation of interest connecting the subject and the object. For instance, in the sentence,

'Tetrazolium salt is converted by cellular dehydrogeneases'

the extracted semantic triple would be

```
subject:  Tetrazolium salt
predicate:  is converted by
object:  dehydrogenases, cellular dehydrogenases
```

Though the idea is straightforward, extracting such triples is a challenging task that involves handling sentences with complex grammatical structures, multiple clauses that need to be split into simpler ones, and performing coreference resolution that involves finding all mentions in text referring to the same entity. A framework that mitigates these challenges was proposed in [127] and is now part of the Stanford OpenIE framework. This framework was used to extract relations from text as semantic triples. The important words identified were mapped to their relations based on the extracted relations to auto-generate knowledge graphs like the one shown in Figure 5.1.

5.3 Dataset and model training

5.3.1 Dataset

Since our framework does not rely on labeled datasets for model training (due to the presence of weak supervision stage), any data source that contains concepts relevant for drug discovery and development could be used. Thus, the publicly available [140] international conference on harmonization (ICH) guideline documents on quality, safety, and efficacy containing a wide spectrum of relevant concepts, were primarily worked with. In total, the dataset comprised 64 documents and a random 80-10-10 train-valid-test split was performed that resulted in 51 documents in the training set, 6 documents in validation set, and 7 documents in the test set. In addition, an internal technical report obtained from Eli Lilly was used as an additional test document for evaluating model performance. The list of documents used for training, validation, and testing are listed in Table 5.2. In total, there were 21571 examples (sentences) in the training set, 2338 in the validation set, and 2392 in the test set.

The training and validation datasets were first processed using the weak supervision framework to assign labels (0 or 1, depending on whether the word was identified as important or not) to each word in the sentence, thus transforming the unlabeled ICH documents to a labeled training dataset. This labeled dataset was then passed on to the supervised learning stage involving the BioBERT fine-tuning. The BioBERT model input during the training

stage would be sentences with tokenized words along with their target labels (0 or 1). For tokenizing words, the WordPiece tokenizer [141] was used which splits a given word into possibly multiple subwords (or pieces) based on their frequency of occurrence in the training corpus and uses special token '##' to indicate word pieces that are not the first part of a given word. For instance, consider the same sentence in Table 5.1. At the training stage, the input to the model would be:

$$([CLS], 0), (the, 0), (cell, 1), (growth, 1), (is, 0), (assessed, 0), (by, 0), (a, 0), (stain, 1), (\#\#ing, 1),$$
$$(method, 1), (using, 0), (a, 0), (te, 1), (\#\#tra, 1), (\#\#zo, 1), (\#\#lium, 1), (salt, 1), (which, 0), (is, 0),$$
$$(converted, 0), (by, 0), (cellular, 0), (de, 1), (\#\#hy, 1), (\#\#dr, 1), (\#\#ogen, 1), (\#\#ases, 1), (to, 0),$$
$$(a, 0), (colored, 0), (form, 1), (\#\#az, 1), (\#\#an, 1), (product, 1), ([SEP], 0)$$

where, '[CLS]' and '[SEP]' are special tokens that indicate the start and end of sentences, respectively. During the model evaluation or test stage, the input to the model are similar to those in the training stage with words in the sentence tokenized using WordPiece tokenizer, but without their target labels. The target labels are predicted using the trained or fine-tuned BioBERT model, classifying each word as not important (label 0) or important (label 1).

Table 5.2: List of documents used for training, validation, and evaluation of the model. All data (except the technical report from Eli Lilly) is publicly available at [140]

S.No	Document	Description/Title	Type
1	E2A guideline	Clinical safety data management: definitions and standards for expedited reporting E2A	train
2	E2D guideline	Post-approval safety data management: definitions and standards for expedited reporting E2D	train
3	E2E guideline	Pharmacovigilance planning E2E	train
4	E2F guideline	Development safety update report E2F	train
5	E3 guideline	Structure and content of clinical study reports E3	train
6	E4 guideline	Dose-response information to support drug registration E4	train
7	E5 R1 guideline	Ethnic factors in the acceptability of foreign clinical data E5(R1)	train
8	E7 guideline	Studies in support of special populations: geriatrics E7	train

Table 5.2: List of documents used for training, validation, and evaluation of the model. All data (except the technical report from Eli Lilly) is publicly available at [140]

S.No	Document	Description/Title	Type
9	E8-R1 guideline step 4	General considerations for clinical studies E8(R1)	train
10	E9 guideline	Statistical principles for clinical trials E9	train
11	E10 guideline	Choice of control group and related issues in clinical trials E10	train
12	E11 R1 guideline	Addendum to ICH E11: clinical investigation of medicinal products in the pediatric population E11 (R1)	train
13	E14 guideline	The clinical evaluation of qt/qtc interval prolongation and proarrhythmic potential for non- antiarrhythmic drugs E14	train
14	E15 guideline	Definitions for genomic biomarkers, pharmacogenomics, pharmacogenetics, genomic data and sample coding categories E15	train
15	E16 guideline	Biomarkers related to drug or biotechnology product development: context, structure and format of qualification submissions E16	train
16	E17 EWG step 4	General principles for planning and design of multi-regional clinical trials E17	train
17	E18 guideline	Guideline on genomic sampling and management of genomic data E18	train
18	E19 EWG guideline	Optimisation of safety data collection E19	train
19	E11A step 2 guideline	Pediatric extrapolation E11A	train
20	Q3C-R8 step 4 guideline	Impurities: guideline for residual solvents Q3C(R8)	train
21	Q14 step 2 guideline	Analytical procedure development Q14	train
22	Q1B guideline	Stability testing: photostability testing of new drug substances and products Q1B	train
23	Q1C guideline	Stability testing for new dosage forms Q1C	train
24	Q1D guideline	Bracketing and matrixing designs for stability testing of new drug substances and products Q1D	train
25	Q1E guideline	Evaluation for stability data Q1E	train
26	Q1F stability guideline	Stability testing of active pharmaceutical ingredients and finished pharmaceutical products	train
27	Q3A(R2) guideline	Impurities in new drug substances Q3A(R2)	train
28	Q3B(R2) guideline	Impurities in new drug substances Q3B(R2)	train
29	Q3D(R2) step 4 guideline	Guideline for elemental impurities Q3D(R2)	train
30	Q4B guideline	Evaluation and recommendation of pharmacopoeial texts for use in the ICH regions Q4B	train
31	Q5B guideline	Quality of biotechnological products: Analysis of the expression construct in cells used for production of r-dna derived protein products Q5B	train
32	Q5C guideline	Quality of biotechnological products: Stability testing of biotechnological/biological products Q5C	train
33	Q5E guideline	Comparability of biotechnological/biological products subject to changes in their manufacturing process Q5E	train
34	Q6B guideline	Specifications: test procedures and acceptance criteria for biotechnological/biological products Q6B	train
35	Q7 guideline	Good manufacturing practice guide for active pharmaceutical ingredients Q7	train

Table 5.2: List of documents used for training, validation, and evaluation of the model. All data (except the technical report from Eli Lilly) is publicly available at [140]

S.No	Document	Description/Title	Type
36	Q8(R2) guideline	Pharmaceutical development Q8(R2)	train
37	Q9 guideline	Quality risk management Q9	train
38	Q10 guideline	Pharmaceutical quality system Q10	train
39	Q11 guideline	Development and manufacture of drug substances (chemical entities and biotechnological/biological entities) Q11	train
40	Q12 guideline step 4	Technical and regulatory considerations for pharmaceutical product lifecycle management Q12	train
41	S1A guideline	Guideline on the need for carcinogenicity studies of pharmaceuticals S1A	train
42	S1B-R1 guideline	Testing for carcinogenicity of pharmaceuticals S1B(R1)	train
43	S2(R1) guideline	Guidance on genotoxicity testing and data interpretation for pharmaceuticals intended for human use S2(R1)	train
44	S3A guidelne	Note for guidance on toxicokinetics: the assessment of systemic exposure in toxicity studies S3A	train
45	S3B guideline	Pharmacokinetics: guidance for repeated dose tissue distribution studies S3B	train
46	S4 guideline	Duration of chronic toxicity testing in animals (rodent and non rodent toxicity testing) S4	train
47	S5-R3 step 4 guideline	Detection of reproductive and developmental toxicity for human pharmaceuticals S5(R3)	train
48	S6 R1 guideline	Preclinical safety evaluation of biotechnology-derived pharmaceuticals S6(R1)	train
49	S7B guideline	The non-clinical evaluation of the potential for delayed ventricular repolarization (qt interval prolongation) by human pharmaceuticals S7B	train
50	S10 guideline	Photosafety evaluation of pharmaceuticals S10	train
51	S11 step 4 guideline	Nonclinical safety testing in support of development of paediatric pharmaceuticals S11	train
52	E2C R2 guideline	Periodic benefit-risk evaluation report (PBRER) E2C(R2)	valid
53	Q2-R2 step 2 guideline	Validation of analytical procedures Q2(R2)	valid
54	S12 step 2 guideline	Nonclinical biodistribution considerations for gene therapy products S12	valid
55	Q6A guideline	Specifications: test procedures and acceptance criteria for new drug substances and new drug products: chemical substances Q6A	valid
56	S7A guideline	Safety pharmacology studies for human pharmaceuticals S7A	valid
57	S9 guideline	Nonclinical evaluation for anticancer pharmaceuticals S9	valid
58	E1 guideline	The extent of population exposure to assess clinical safety for drugs intended for long-term treatment of non-life-threatening conditions E1	test
59	Q1A(R2) guideline	Stability testing of new drug substances and products Q1A(R2)	test
60	Q2(R1) guideline	Validation of analytical procedures: text and methodology Q2(R1)	test
61	Q5A (R1) guideline	Viral safety evaluation of biotechnology products derived from cell lines of human or animal origin Q5A(R1)	test

Table 5.2: List of documents used for training, validation, and evaluation of the model. All data (except the technical report from Eli Lilly) is publicly available at [140]

S.No	Document	Description/Title	Type
62	Q5D guideline	Derivation and characterisation of cell substrates used for production of biotechnological/biological products Q5D	test
63	S1C(R2) guideline	Dose selection for carcinogenicity studies of pharmaceuticals S1C(R2)	test
64	S8 guideline	Immunotoxicity studies for human pharmaceuticals S8	test
65	Eli Lilly document	Eli Lilly internal technical report	test

5.3.2 BioBERT model architecture and training

The BioBERT model is a pre-trained language representation model trained on biomedical corpora, containing PubMed abstracts (4.5B words), PubMed Central full-text articles (13.5B words), and standard English language corpus (3B words) [126]. BioBERT is based on the BERT model [142], a state-of-the art sequence model used for learning word representations for English language and is trained on general domain texts. A transformer model [**vaswani2017attention**] underlies the BERT architecture, which has been shown to be successful in various engineering applications including reaction prediction [28, 47], retrosynthesis [52, 29, 5], electrocatalyst discovery [143], property prediction [88], chemical product design [40, 144], and so on. BioBERT overcomes the limitations of using BERT on BioMedical text by using a domain-focused training set and learning efficient word distributions more focused on the BioMedical domain. Hence, BioBERT is based on the BERT model architecture and is fine-tuned on a large corpus of BioMedical datasets. A schematic of the BioBERT model architecture is shown in Figure 5.9.

Large models such as BioBERT could also be fine-tuned on a downstream task without retraining the entire model. This involves initializing the BioBERT model architecture using the pre-trained model weights, and then training the model from this stage on a custom dataset or task for few epochs. In our case, the custom dataset is the set of ICH documents in the training set and the task is to use the labeled dataset after weak supervision stage to train a BioBERT-based classifier. Since this is a binary classification task with two labels, a

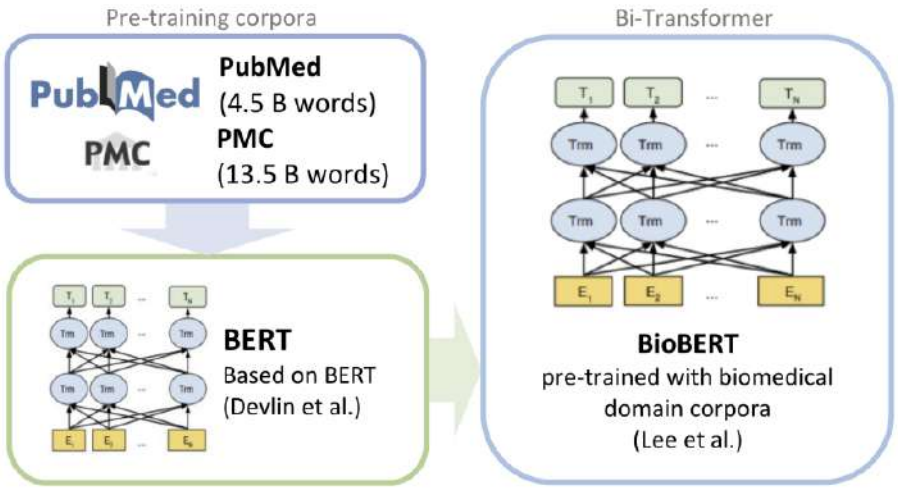


Figure 5.9: A schematic of the BioBERT architecture and pre-training strategy based on BERT. Schematic adapted from [126].

cross-entropy loss function defined by

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i.log(p(y_i)) + (1-y_i).log(1-p(y_i)) \tag{5.3}$$

was minimized, where $p(y_i)$ is the predicted probability of i being in class 1, (and hence, a probability $1-p(y_i)$ of it being in class 0), y_i is the true class label, and N is the total number of data points. The BioBERT model was fine-tuned on this task for 3 epochs since – first, fine-tuning for just a few epochs is usually sufficient to achieve optimal performance, and second, to avoid overfitting on the small custom training dataset with noisy labels from weak supervision. Moreover, as seen from the validation loss in Figure 5.10, the validation loss plateaus around 3 epochs. Since the model is being fine-tuned on a smaller dataset, extensive hyperparameter optimization was not performed except for the learning rate, weight decay, and the number of epochs using the validation set. The model is characterized by 107M total training parameters. Note that since the model is only being fine-tuned for a few epochs, the computational cost associated with the model is much smaller.

Remark 1: Note that the model considers the sequential position of the words in a sentence due to its autoregressive nature. Thus, the contextual information about the words are used

inherently while predicting the target labels.

5.4 Results

5.4.1 Model performance evaluation and statistics

The model training results showing the training loss and validation loss with training steps during the model fine-tuning stage are presented. The training and validation loss as the training progresses is shown in Figure 5.10 and it is observed that the validation loss plateaus around 1750 training steps which corresponds to 3 epochs. The predictions from the model are in the form of logits or raw probability scores which are mapped to their respective class predictions based on higher logit values. To further understand the performance of the model, the standard metrics computed for classifiers such as – precision, recall, F1 score, and accuracy are computed. Precision indicates the fraction of true positives across all the positive class predictions made by the classifier (TP/(TP+FP)); recall indicates the fraction of true positives across all the true positives in the dataset (TP/(TP+FN); F1-score is a harmonic mean of precision and recall and shows the trade-off between them as a single measure; and accuracy is the fraction of correct predictions across all the predictions ((TP+TN)/(TP+TN+FP+FN)). The precision, recall, F1-score, and accuracy on the training, evaluation, and test dataset are shown in Table 5.3.

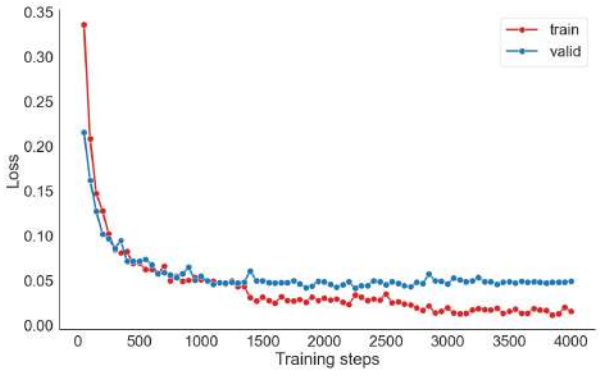


Figure 5.10: Training and evaluation loss with training steps

Table 5.3: Evaluation metrics on the train, validation, and test set. For the test set, numbers in parentheses indicate performance metrics on the ICH test set documents (i.e. excluding the Eli Lilly internal technical report)

Metric	train	validation	**test**
Precision	0.989	0.954	**0.943 (0.955)**
Recall	0.991	0.973	**0.824 (0.975)**
F1-score	0.990	0.964	**0.879 (0.965)**
Accuracy	0.996	0.988	**0.954 (0.987)**

It should be noted that the ground truth labels are auto-generated based on the weak supervision strategy which are inherently noisy and thus the labels may not be completely accurate. Although these metrics give a good indication of the model's abilities, they may not be completely representative of the true model performance given the noisy nature of assigned labels. Some of the metrics for evaluating NER performance and created other custom metrics are therefore adapted to provide further insights into the performance of our approach. First, entity detection rate indicating the fraction of words labeled as entities by our framework is defined as,

$$\text{detection rate} = \frac{\#\text{ of unique tagged words}}{\#\text{ of unique words in the entire document}} \tag{5.4}$$

Second, to assess the accuracy of these labels, an indirect route using tf-idf (term frequency-inverse document frequency)-based word importance scores [145] to identify the most important words across documents is defined as,

$$\begin{aligned} &\text{tf(t,d)} = f_{t,d} \qquad\qquad \text{idf(t,D)} = \log \frac{N}{\mid \{d \in D : t \in D\} \mid} \\ &\text{tf-idf(t,d,D)} = \text{tf(t,d)} \times \text{idf(t,D)} \end{aligned} \tag{5.5}$$

where $f_{t,d}$ is the raw term frequency of term t in document d, $idf(t, D)$ is the inverse document frequency of term t across a corpus of documents D, and $\mid \{d \in D : t \in D\} \mid$ is the number of documents across $N = \mid D \mid$ total documents where the term t appears. For each document, 500 words with the highest tf-idf scores are identified to get a list of important words that need to be captured. The fraction of these 500 words identified by the NER model as important is then computed to indirectly assess the accuracy of the model by estimating its ability to capture these important words. Hence, the actual performance of the model is possibly better than these scores, however, they could be treated as a lower bound on the model performance. The assumption here is that words with high tf-idf scores are likely to be important and hence should be captured but there is no differentiation between general words and drug discovery

and development related words. The entity detection rate and tf-idf scores-based coverage fraction of the framework estimated using the test set are presented in Table 5.4.

Table 5.4: Test set statistics of the NER approach using SUSIE

Document	detection rate (%)	tf-idf coverage (%)
E1 guideline	34.5	26.4
Q1A(R2) guideline	39.6	71.2
Q2(R1) guideline	38.2	59.4
Q5D guideline	48.4	89.6
S1C(R2) guideline	40.0	63.4
Q5A(R1) guideline	42.7	84.8
S8 guideline	48.9	85.2
Eli Lilly internal report	47.2	86.4
Average	42.4	70.8

5.4.2 Test set information chunks identified and auto-generated knowledge graphs

Recall that apart from the ICH documents, an internal Eli Lilly report was also a part of the test set. This document possibly contained several new concepts and terms that were likely not part of the ICH documents and hence, were not seen by the model during the training stage. This document therefore serves as an ideal test bed to evaluate the model performance on truly out-of-sample documents. Few examples of important information chunks identified from the document using SUSIE are highlighted below.

Remark 2: Note that certain chemicals/drug names that are confidential in nature have been masked-out using terms such as Compound_1/2/3, Chemical_1/2/3, IUPAC_1/2/3, Drug_1/2/3, and so on.

Input 1: Compound_1 is not a bacterial mutagen in the Ames assay. The same assessment criteria would be used to evaluate any future route modifications to determine if additional controls for genotoxic species will be needed in proposed starting material Compound_1.

Input 2: Compound_1 is introduced in the first step of the proposed 3-step commercial manufacturing process for Drug_1 and is incorporated into the drug substance as a significant structural fragment. It is a chemically-stable, highly-purified crystalline material that has more desirable properties for a starting material than compounds earlier in the synthetic route. Compound_1 can be readily characterized by commonly known analytical techniques, which separate and quantify known impurities, as well as potential isomeric impurities and degradation products.

Input 3: The structures for both impurities listed in Table 3.2-2 are shown in Figure 3.2-1. The structure of Compound_2 has been confirmed by independent synthesis of an authentic sample and characterization by sensitive spectroscopic methods. Benzophenone (Compound_3) is a commercially available material. Both of these impurities, and their downstream process derivatives, are either fully purged or are present in the final drug substance at levels below the ICH reporting threshold (0.05%). Structures of the downstream process derivatives are provided in Background Information.

Input 4: The fate of Compound_3 in Step 1 of the proposed commercial manufacturing process was determined by reacting Compound_2 with starting material compound_1, as shown in Scheme 6.5.2.1-2. One main component was observed in both Step 1a and Step 1b, which were determined to be Compound_4 and Compound_5, respectively, by LC-MS characterization.

For evaluation of the fate and purge of Compound_2, Step 1 of the commercial manufacturing process was performed with 1% of Compound_2 added to the proposed starting material Compound_8. The isolated product Compound_6, contained 0.09% of Compound_5, with neither Compound_2 nor Compound_4 being observed at a reporting threshold of 0.05%. Thus, a rejection efficiency of 91% was demonstrated for Compound_2 in Step 1.

Input 5: Drug_1 is being developed as 80 mg and 150 mg immediate release tablets. The following discussion summarizes the development of the dissolution conditions including justification of the testing conditions chosen; apparatus and rotation speed, media conditions selected; and demonstration of the discriminating capability of the conditions. The complete method and validation summary are provided in Appendices 3.4.A1 and 3.4.A2, respectively, at the end of this section.

For inputs 2 and 3 above, the knowledge graph representing the extracted information presented in a structured manner are shown in Figures 5.11 and 5.12. Additional knowledge graphs for examples shown in Appendix E from the technical report.

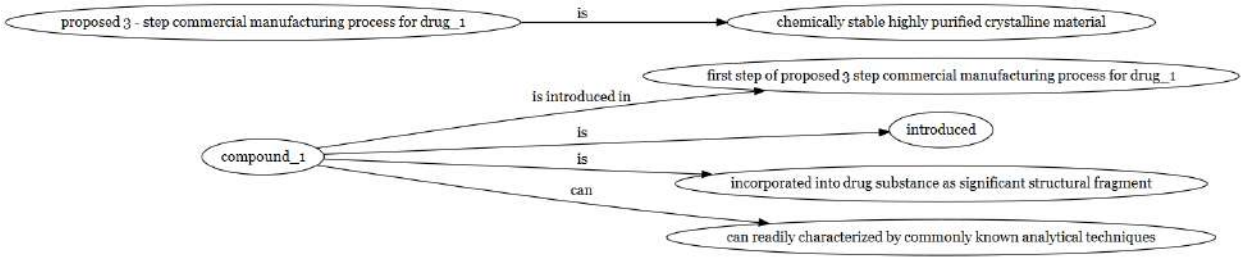


Figure 5.11: Automatically generated knowledge graph for example Input 2 above

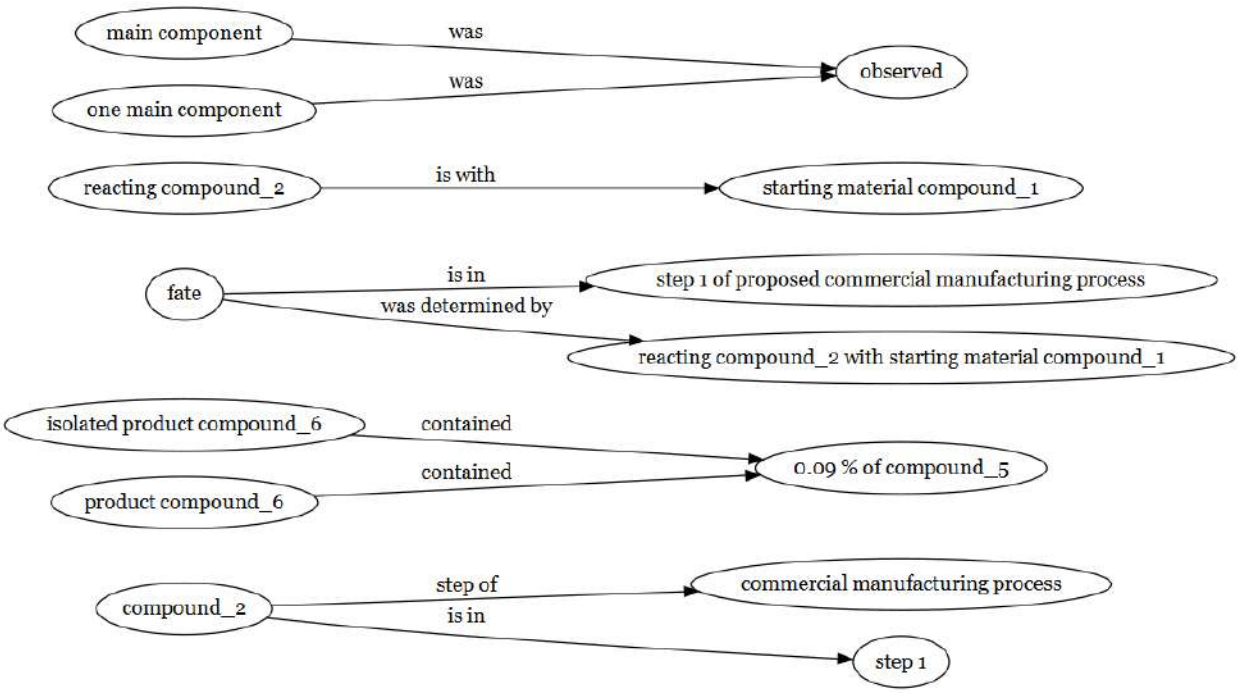


Figure 5.12: Auto-generated knowledge graph for example Input 4 above

Based on the above, it is observed that our framework (SUSIE) automatically generates structured knowledge graphs from unstructured pharmaceutical text inputs. The knowledge graphs capture the important information (from a drug development or CMC standpoint) as defined by the custom built and standard ontologies used in our framework. SUSIE also captures additional contextual information characterizing the information chunks. For instance, in input 1, 'Compound_1' along with its associated information that it's a 'proposed starting material' has been captured. In input 2, drug names that were likely not present in the training database but were automatically captured based on their neighboring context, thus pointing towards the model's ability to handle new and relevant concepts not seen during the training stage. However, it is also observed that certain misses that are difficult for the model to capture at this stage. For instance, consider input 5 where the model correctly identified the 150mg dosage information for the drug tablets. However, the 80mg dosage information that was also associated with the same drug was missed. However, a subject matter expert (or SME) input is still needed to filter out the important relationships from

generic ones. But the knowledge graph is still useful since it surfaces output in a way that makes it easier for SME to spot the important relationships. Improving the capabilities of the model to allow capturing such complex but relevant contextual information could be addressed using relation extraction and is part of our future work on this framework. Further examples from the report are provided in Appendix E.

5.5 Conclusions

An end-to-end pharmaceutical relevant information extraction framework called SUSIE was developed by combining – domain ontologies for representing important concepts, a weak supervision framework for transforming unlabeled datasets to labeled datsaets, a fine-tuned BioBERT model that represents a generalized model able to handle new documents, contextualization modules to capture relevant context associated with important entities, and a relation extraction approach for auto-generating knowledge graphs. The framework was trained on publicly available (unlabeled) ICH documents and achieves a test accuracy and F1-score of 96% and 88%, respectively, on out-of-sample documents including an internal technical report from Eli Lilly. A major contribution of this work is to develop a custom pharmaceutical drug development ontology and build an information extraction framework upon it that elimintates the need for manual data curation for model training. The underlying framework characterizing SUSIE is generalizable and adaptable to other domains.

Chapter 6: Hypergraph Network of Organic Chemistry

With the accelerated discovery of new reactions and complex molecules due to advances in computational methods, chemistry literature has been growing rapidly. The major drivers for this growth are the advances in molecule optimization, reaction engineering and optimization resulting in the discovery of novel reactions that were either unknown earlier or were infeasible, and high-throughput screening methods that have led to the re-engineering (or re-wiring) of existing reactions to make them more cost-effective and sustainable from an environmental standpoint.

To condense (and make sense of) the huge amount of chemistry literature that is available to us mostly in an unstructured format, tools that could be used to represent this knowledge in a structured format, compute coarse-grained statistics that summarize the information effectively, identify general trends on the evolution and growth of the domain, and discover new chemistry insights that were unknown earlier, are required. While a framework that addresses these requirements could be custom-developed, network theory naturally offers tools and techniques such as – structural statistics [146, 147], centrality measures [148], clustering [149], network embedding [150, 151], link prediction [152, 153] – that could be used to tackle these requirements. There are several variations of graph-based representations for chemical reactions, but the most common is a directed graph representation where nodes represent molecules and directed edges from reactant nodes to product nodes represent reactions. Studies based on such dyadic representations have reported several interesting properties of the reactions network such as their scale-free network structure similar to the World Wide Web (WWW) [154], the existence of core (most useful) and peripheral molecules across organic chemistry reactions [155], the small-world nature of reaction networks [156], which is shown to make a network robust towards node/edge deletions [157]. [158, 110] demonstrated ap-

plications of network theory-based studies in parallel synthesis, reactivity estimation, and rewiring of synthetic pathways.

The traditional directed graph representation for chemical reactions has several limitations. First, a directed graph representation does not capture the complete reaction context, i.e. it introduces independent directed edges for multi-reactant (or multi-product) reactions from each reactant to each product, thus losing contextual information on the presence of other reactants (or products). As a result, several seemingly independent, directed edges might correspond to the same reaction. Second, a dyadic graph representation does not allow for reaction (or edge)-specific molecular (or node) properties such as relative molecular complexity, reactivity, stoichiometry, reaction kinetics, and other properties that might be useful for making the graph representation more complete, rich, and chemistry-aware. Third, due to the above limitations, the analyses generally could not be analyzed in a self-contained manner to draw inferences and identify the trends in chemistry that are not an artifact of the reaction representation, as observed for degree correlations in [156].

To address these limitations, an alternative hypergraph representation is proposed where molecules are represented as vertices and an entire reaction is represented as a hyperedge. Since hypergraphs allow for an edge (or hyperedge) to connect multiple vertices together (and not just two), the entire reaction is represented using just a single, unique hyperedge. To address the issue of incorporating reaction-specific node attributes, a recently proposed annotated hypergraph framework [159] is used, which allows for each node to have hyperedge-specific annotations and makes the representation flexible to allow for reaction-specific contextual information. Therefore, compared with directed graphs, annotated hypergraphs are much more frugal in terms of the number of hyperedges, flexible in capturing reaction-level context, and due to the one-to-one correspondence between hyperedges and reactions, the statistics are self-contained, which correspond to underlying chemistry trends.

Here, the directed graph representation of chemical reactions and an annotated hypergraph representation is compared and contrasted using a standard organic chemistry reactions

database containing nearly half a million reactions. This work is the first attempt to study the network of organic chemistry using a hypergraph framework that is shown to be frugal, rich, and chemistry-aware in nature, making them suitable for deriving chemistry inferences. To allow for a one-to-one comparison between the dyadic representation and the hypergraph representation, standard network properties for the directed graph representation and an equivalent hypergraph representation are computed using the same reactions dataset. At the same time, the time-evolution of these properties are also reported. It is also shown how a hypergraph could be transformed into a weighted directed graph to allow for computation of dyadic network properties that may be ill-defined or difficult to compute for hypergraphs (at the moment). Finally, to demonstrate the use-case of such hypergraph representations not just for understanding chemistry trends but also for reaction engineering, it is shown how the hypergraph representation could be used in the reaction classification problem, i.e., predicting the reaction type given participating molecules, which has applications in reaction mechanism generation, retrosynthetic planning, and feasibility analysis.

6.1 Properties of directed graphs and hypergraphs

In this section, directed graphs, annotated hypergraphs, and the various network statistics that are used to characterize the hypergraph network of organic chemistry are formally defined. The following sections could also be treated as a tutorial that motivates various network properties using an example set of four simple reactions containing five different molecules.

6.1.1 Mathematical representation

A directed graph is an ordered pair $G = (V, E)$ of a set of vertices V and a corresponding set of edges E. Each edge e_i in E connects a source node s_i to a target node t_i, giving directionality to the set of edges, thus resulting in a *directed* graph as opposed to an undirected graph. Chemical reactions could also be represented using such directed graphs where the

reactants and products are represented as vertices, and directed edges from reactants to products representing reactions. For reactions with multiple reactants and products, the directed graph is typically constructed using all-to-all wiring with all reactants of a given reaction connecting individually to all products in the reaction through independent directed edges. Figure 6.1(a) shows a directed graph representation for the set of four reactions $(R1, R2, R3, R4)$ with 5 different molecules (A, B, C, D, E) shown in Equation 6.1.

$$\begin{aligned} R1: \quad & A \longrightarrow B & \qquad R3: \quad & C \longrightarrow E + D \\ R2: \quad & B + E \longrightarrow C & \qquad R4: \quad & D + C \longrightarrow A + E \end{aligned} \tag{6.1}$$

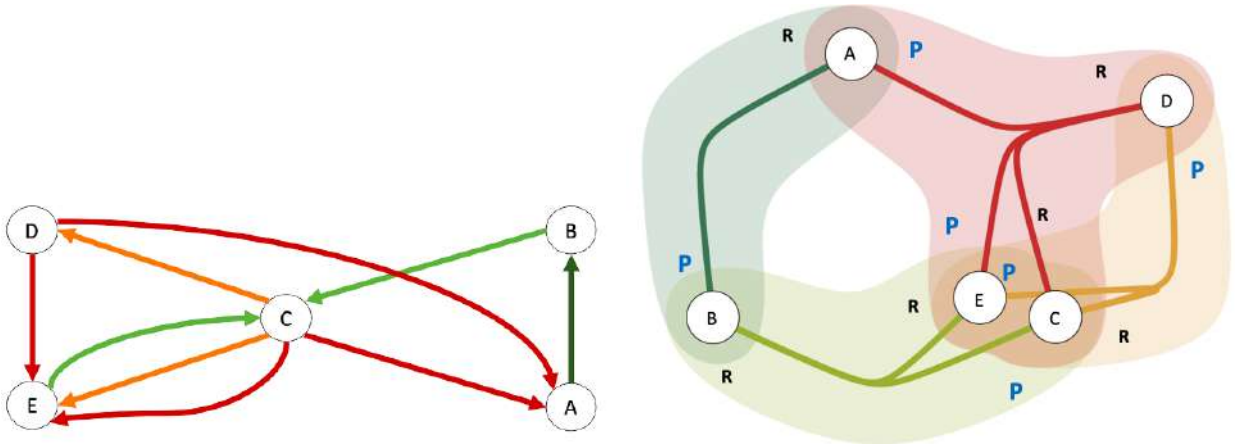


Figure 6.1: (a) Directed graph-based representation (b) Annotated hypergraph-based representation where an entire reaction is represented using a single hyperedge and the annotations indicate the vertex 'roles' as product (P) or reactant (R)

On the other hand, a hypergraph is a generalization of a graph where each edge is not limited to connecting just two vertices but could connect any number of vertices via hyperedges. Mathematically, a hypergraph is a pair $H = (V, E)$ where V is a set of vertices and E is the set of edges (or *hyperedges*) where each edge contains a non-empty subset of V. Since each chemical reaction has contextual information about molecules along with an inherent directionality, *annotated* hypergraphs [159] with hyperedge-specific annotations (or roles) for nodes in a hyperedge are used. An annotated hypergraph is defined as $A = (V, E, X, l)$ where

V is the set of nodes, E is a labeled hyperedge set where each hyperedge is a subset of V, X is a finite label set containing the possible set of labels (or annotations/roles), and l is a role labeling function for assigning roles to each edge in the label. It should be noted that each node v would have a given role x in given edge e, written as $l(v, e) = x$. Roles are contextual and they are assigned to node-edge pairs, unlike node attributes that are defined a priori for each node in dyadic graphs. For a set of chemical reactions, the set of vertices would be nodes, reactions containing the set of vertices participating in the reaction are represented as hyperedges, and the node-edge pair role could either be 'product (P)' or 'reactant (R)' for nodes that play the role of reactants or products in a reaction, respectively. Figure 6.1(b) shows the equivalent hypergraph representation for the set of four reactions in Equation 6.1.

Remark 1: Observe that the number of (hyper)edges in a hypergraph representation is the same as the number of reactions, but this is not the case with edges in a directed graph representation.

Remark 2: One of the primary benefits of using annotated hypergraphs is the incorporation of contextual information about reactions and molecules through hypergraph annotations or roles.

6.1.2 Degree distributions

Degree distributions provide a general sense of the network structure and its connectivity pattern. Generating a degree distribution involves computing the degree (or number of edges) for each node and estimating the underlying probabilistic distribution that they follow. For a directed graph, each node has two kinds of degrees – incoming degree (number of incoming edges, d_{in}) and outgoing degree (number of outgoing edges, d_{out}). The sum of the incoming and outgoing degrees, total degree ($d_{in} + d_{out} = d_{total}$), is the same as the degree of an equivalent undirected graph with directionality removed from directed edges.

For an annotated hypergraph, equivalent degree distributions could be defined. The incoming degree for a node in the annotated hypergraph would involve counting the number of

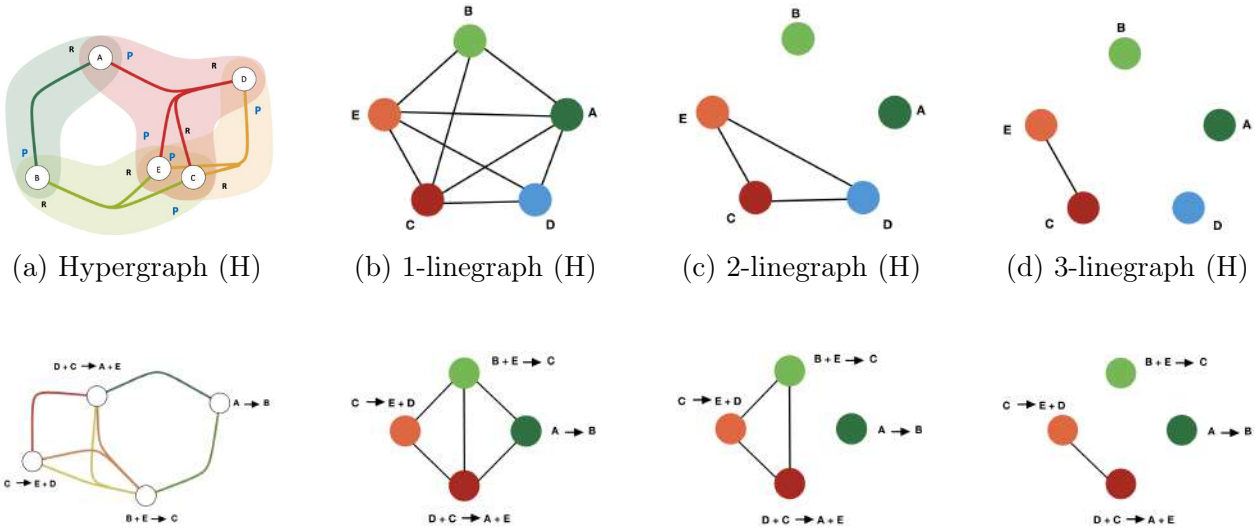
hyperedges where the node participates with a role 'product' ($d_{product}$ or d_{in}) since products have incoming edges, and the outgoing degree would involve counting the number of hyperedges where the node participates with a role 'reactant' ($d_{reactant}$ or d_{out}) since reactants have outgoing edges. The sum of the incoming and outgoing degrees would be the total degree ($d_{product} + d_{reactant} = d_{total}$).

Table 6.1 shows the incoming and outgoing degrees for each node in the set of reactions in Equation 6.1 for directed graph and hypergraph representations.

Table 6.1: Degree distributions for the example set of reactions in Equation 6.1

Node	Directed graph		Annotated hypergraph	
	in	out	in (P)	out (R)
A	2	1	1	1
B	1	1	1	1
C	2	4	1	2
D	1	2	1	1
E	3	1	2	1

Figure 6.2: The hypergraph (H), dual hypergraph (H^*), and their respective s-linegraphs for the example set of four reactions in Equation 6.1



(a) Hypergraph (H) (b) 1-linegraph (H) (c) 2-linegraph (H) (d) 3-linegraph (H)

(e) Dual hypergraph (H*) (f) 1-linegraph (H^*) (g) 2-linegraph (H^*) (h) 3-linegraph (H^*)

Figure 6.3: The hypergraph (H), dual hypergraph (H^*), and their respective s-linegraphs for the example set of four reactions in Equation 6.1

6.1.3 Average shortest path length

The average shortest path length of a network measures the separation between nodes (on average) in term of the number of edges between nodes. Since this measure involves computing the separation between *all nodes*, the network is required to be connected, i.e., there must exist a path from any node to any other node in the network. For a directed graph, the average shortest path length is the number of directed edges between nodes with the constraint that the distance should be measured along the direction of the edges. For undirected graphs, this is simply the average number of edges between nodes, irrespective of the directionality. This is often referred to as the all pairs shortest path (APSP), and is defined as,

$$l = \sum_{s,t \epsilon V} \frac{d(s,t)}{n(n-1)} \tag{6.2}$$

where $d(s,t)$ is the distance between nodes s and t, and n is the total number of nodes in the network.

To define connectivity for hypergraphs, two new concepts are introduced – dual hypergraphs and linegraphs. First, the dual hypergraph H^* of a hypergraph H is a hypergraph with nodes and edges interchanged. Therefore, in an H^*, the nodes represent reactions and the hyperedges represent the set of molecules common between the nodes that it connects. Second, a linegraph $L(H)$ of a hypergraph H is defined as a graph whose vertex set is the set of vertices of H with two vertices adjacent and connected in $L(H)$ when their corresponding hyperedges have a non-empty intersection, i.e. they have common hyperedges (or reactions in our context). Therefore, a hypergraph H is said to be connected if its linegraph $L(H)$ is connected. A generalization of linegraphs is the s-linegraph where s (an integer, ≥ 1) indicates the minimum size of the intersection, thus giving rise to s-linegraphs. Because of the duality property of hypergraphs, an equivalent linegraph $L(H^*)$ could be created for the dual hyeprgraph H^* where the set of vertices represent hyperedges and adjacent vertices are connected if they have non-empty intersections, i.e. common molecules in our context. The s-linegraphs for the example set of reactions in Equation 6.1 for different values of s is shown in Figure 6.3 for H and H^*.

Now, for hypergraphs, the average shortest path length could be defined in the same manner as for dyadic graphs by computing the distance between nodes in an s-linegraph of H (known as s-distance). For our purpose, the 1-linegraph is generated and the average shortest 1-distance between the nodes using Equation 6.2 is computed.

Since the computation of the average shortest path length requires the graph to be connected, the largest connected subcomponent both for the directed graph and the hypergraph are identified and their respective average shortest path lengths are reported. For the example set of four reactions in Equation 6.1, since both the directed graph representation and the hypergraphs's 1-linegraph representations are connected, their largest connected subcomponents are the same as their respective graphs (or hypergraphs). The average path lengths computed for the regular (undirected) graph and the hypergraph is show in Table 6.2.

Remark 3: It is evident from the above table that in a hypergraph, the distances between

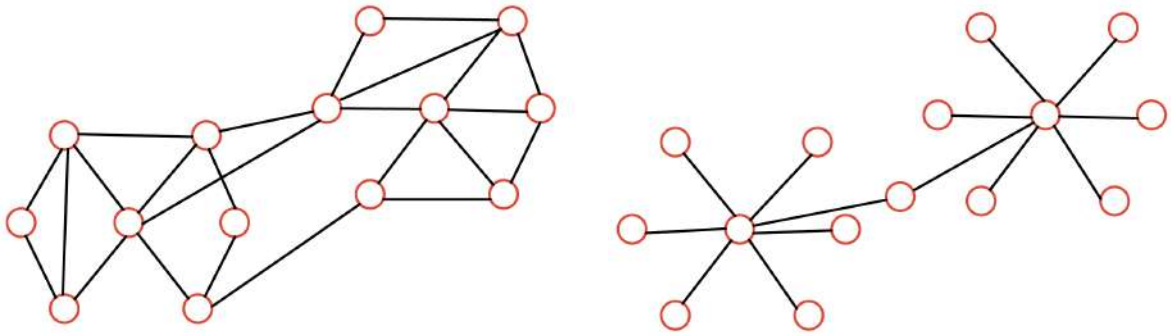
Table 6.2: All pairs shortest distance for the example reactions

Node pairs	Graph	Hypergraph
	in	s=1
d_{A-B}	1	1
d_{A-C}	1	1
d_{A-D}	1	1
d_{A-E}	2	1
d_{B-C}	1	1
d_{B-D}	2	2
d_{B-E}	2	1
d_{C-D}	1	1
d_{C-E}	1	1
d_{D-E}	1	1
Average	**1.3**	**1.1**

nodes correspond exactly to the number of reactions that separate the nodes (or molecules), whereas in the case of a directed graph representation, the distance between nodes corresponds only to partial reactions separating the nodes and not the complete reactions.

6.1.4 Assortativity

Assortativity is a measure of the *mixing patterns* in networks that indicates the general mixing behavior of nodes with other nodes in the network to give rise to a bigger network. Assortativity is defined as the degree correlations between nodes, and therefore, the mixing pattern could either be assortative (positive correlation) or diassortative (negative correlation). The assortativity is often computed as the Pearson correlation coefficient between the degrees of a pair of nodes and takes values between -1 and 1 – a network with an assortativity coefficient of -1 indicates a perfectly disassortative mixing, an assortativity coefficient of 1 points towards a perfectly assortative mixing, and an assortativity coefficient of 0 indicates a non assortative graph. Figure 6.4 shows an example of assortative and disassortative networks.



(a) Assortative (b) Disassortative

Figure 6.4: Different mixing patterns. Assortative networks have mixing patterns that arise due to nodes with similar degree connecting to other nodes with similar degrees, whereas disassortative networks are a result of mixing patterns where nodes with dissimilar degrees connect to each other.

For a directed graph, the in-assortativity ($r_{in,in}$), out-assortativity ($r_{out,out}$), and in-out assortativity ($r_{out,in}$) measures the tendencies of nodes to connect with other nodes that have similar in-degrees, out-degrees, and out-in degrees, respectively. For $\alpha, \beta \in \{in, out\}$, the assortativity $r_{\alpha,\beta}$ for directed graphs is defined as

$$r(\alpha,\beta) = \frac{\sum_i (j_i^\alpha - \bar{j^\alpha})(k_i^\beta - \bar{k^\beta})}{\sqrt{\sum_i (j_i^\alpha - \bar{j^\alpha})^2}\sqrt{\sum_i (k_i^\beta - \bar{k^\beta})^2}} \tag{6.3}$$

where j_i^α is the α-degree of the source node for edge i, k_i^β is the β-degree of the target node for edge i, $\bar{j^\alpha}$ is the average α-degree of source nodes, and $\bar{k^\alpha}$ is the average β-degree of target nodes. For the annotated hypergraph, assortativity is defined with respect to the roles (or annotations) in a manner similar to the directed graph representation in Equation 6.3, replacing the concept of edges with hyperedges and in-out degrees with role-specific (or annotation-specific) node degrees.

For the example set of reactions in Equation 6.1, the assortativity coefficients for the directed graph and the annotated hypergraph are reported in the Table 6.3 below.

Remark 4: These assortativity values could be used to answer questions such as – how likely is it for products with high degree to connect to other products with high degrees, or

Table 6.3: Degree assortativity coefficients for the directed and hypergraph representations for the example set of four reactions

roles pair	directed	hypergraph
p-p	-0.19	-0.43
r-r	-0.53	-0.43
r-p	0.27	0.15

Table 6.4: Network structure overview for the directed and hypergraph representation for the USPTO dataset

	all		1976-1985		1985-2005		after 2005	
	graph	hypergraph	graph	hypergraph	graph	hypergraph	graph	hypergraph
Num reactions	487,724		69,692		259,214		158,818	
Num (hyper)edges	1,245,533	487,724	106,977	69,692	389,072	259,214	289,623	158,818
Num nodes	440,207	440,207	71,268	71,268	238,872	238,872	180,348	180,348

how likely is it that the reactants would connect to other reactants of similar degrees (appear in reactions together), and so on.

6.2 Network statistics on organic chemistry dataset

In this section, the network of organic chemistry is studied through the lens of various network statistics defined in the previous section using a standard organic chemistry reactions database. The primary objective is to highlight the differences and similarities between the network statistics for the directed graph and the hypergraph representations. At the end of each section, chemistry insights that are drawn from such analyses along with the time-evolution of these properties are presented.

6.2.1 Dataset description

The Jin's USPTO-reactions dataset [49] derived from Lowe's text mining work [60] for chemical reactions on the US patents office applications (1976-2016) is the primary dataset that is used to report and compare network statistics. Minimal preprocessing (removed incorrect, incomplete, and duplicate reactions) is performed to allow for the network statistics to capture network properties without possibly losing information due to such preprocessing

exercises. Along with information on reactants and products, the dataset also contained information on the year in which the reaction was reported, allowing us to investigate the time-evolution of the network properties. The final dataset contained 487,724 *single-product reactions* containing information on participating reactants, major products of each reaction, and the year in which the reactions were reported.

Using this dataset, a directed graph and an annotated hypergraph-based networks of organic chemistry is constructed. The directed graph representation was constructed using the all-to-all node connectivity for each reaction. The other wiring possibilities are one-to-one or many-to-one but it has been shown previously that the actual connectivity pattern does not change the network structure and properties [154, 156]. The annotated hypergraph, on the other hand, represents all the reactants and products as part of the same hyperedge with node annotations based on

- reaction roles: 'reactant' or 'product'
- relative length of SMILES strings *in a reaction* with respect to the median SMILES length per reaction: 'SMILES_short', 'SMILES_medium', 'SMILES_long'
- molecular weight *across the entire dataset*: 'molwt_light', 'molwt_medium', 'molwt_heavy'

To perform an analysis of the time-evolution of network properties over different stages of chemistry research, the data is split into three time regimes – regime 1 with reactions reported from 1976 to 1985, regime 2 with reactions reported after 1985 until 2005, and regime 3 with reactions reported from 2005 until 2016. An overview of the directed graph and hypergraph representation obtained using the entire dataset and also using dataset in the three time-regimes is presented in Table 6.4.

Remark 5: Note that in the case of the hypergraph, the number of hyperedges exactly equals the number of reactions in the dataset, whereas for the graph representation, the

number of edges is much higher. Of course, the number of nodes remain the same in both the representations since each node corresponds to a unique molecule in both the representations.

6.2.2 Degree distributions

Degree distribution comparison

First, the degree distributions of both the incoming and outgoing degrees for the directed graph and annotated hypergraph representations are compared. Recall from Section 6.1.2 that for the annotated hypergraph, the incoming degree is the same as the node-degree for annotation 'product' and the outgoing degree is the same as the node-degree for annotation 'reactant'. The degree distributions for the directed graph and for the various annotations in the hypergraph (based on reaction roles, relative SMILES length, and molecular weights as defined in the foregoing section) are presented respectively in Figures 6.5 and 6.6.

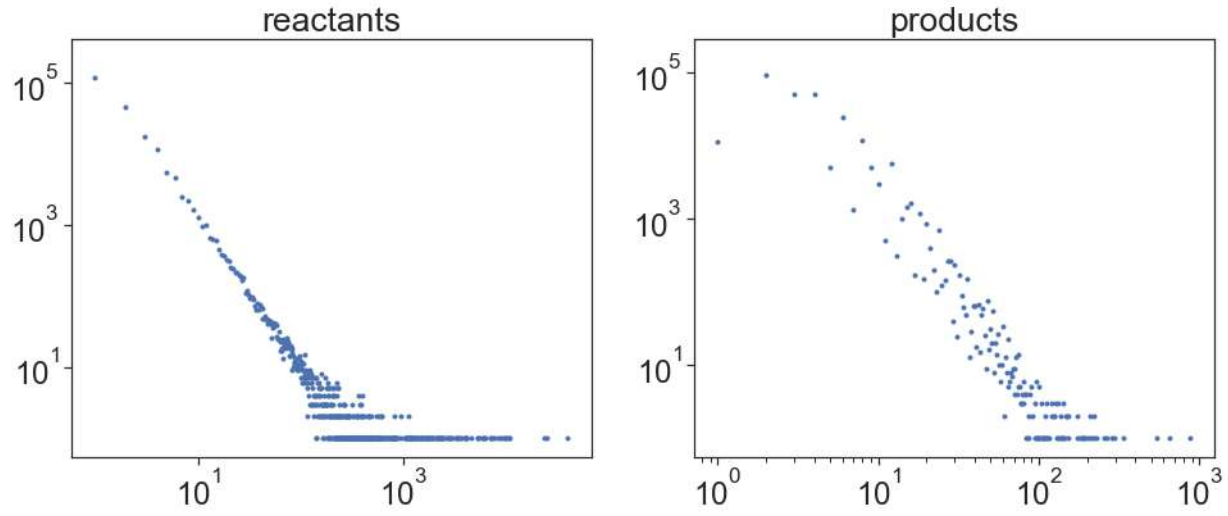


Figure 6.5: Degree distributions for outgoing (reactants) and incoming (product) edges in a directed graph

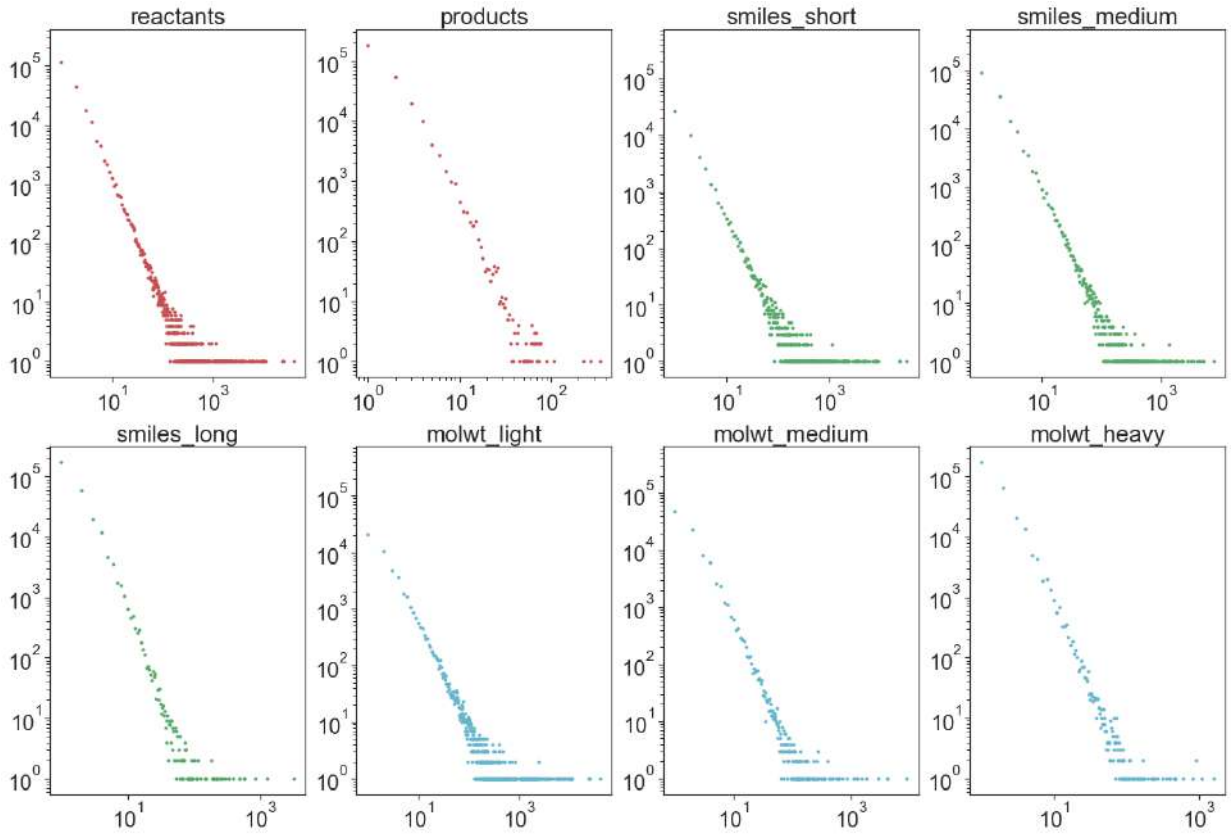


Figure 6.6: Degree distributions for the various hypergraph node-annotations (or roles)

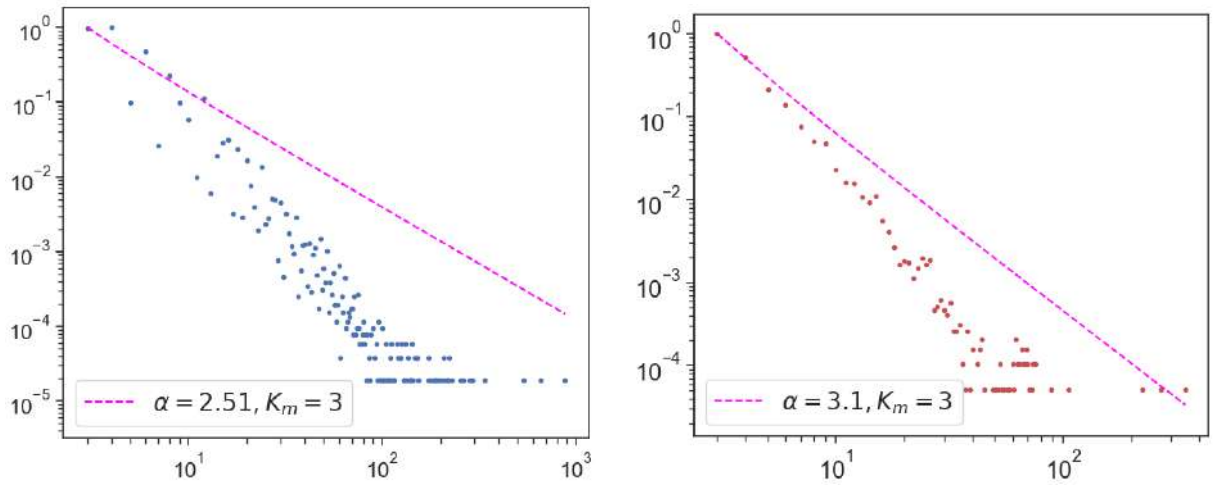
Remark 6: Note that since our dataset only contains single-product reactions, the outgoing degree distribution (reactants) is the same in both representations, and only the incoming (product) degree distributions differ. This is because the directed graph representation for a reaction would have as many incoming edges for the product as the number of reactants whereas in the hypergraph representation the product would have just one incoming edge.

Power law fit for degree distributions

A visual inspection of the degree distributions indicates a possible power law distribution, which is defined as

$$p(k) \propto k^{-\alpha} \tag{6.4}$$

where $p(.)$ is the degree distribution, k is the degree, and α is the scale-free or power law distribution parameter. The existence of a power law distribution points towards an underlying network structure known as the scale-free network structure [147], ubiquitous in real-world networks that often results in 'small-world' behavior. A mathematically rigorous fit is preformed to ensure the existence of a power law using the powerlaw package in Python and estimate the underlying scale-free distribution parameter. The power law fit for the incoming degrees (products) for the directed graph and the hypergraph-based network of organic chemistry are shown in Figure 6.7.



(a) Directed graph (b) Hypergraph

Figure 6.7: Scale-free distribution fit on incoming (products) degrees for (a) Directed graph (b) Hypergraph; K_m is the minimum degree cutoff threshold that is required as a hyperparameter in the powerlaw package.

it is observed that the degree distributions for both the directed graph and hypergraph incoming degrees could be assumed to be coming from a power law distribution, thus pointing towards an underlying scale-free network structure, agreeing with several other studies that have shown that chemistry networks exhibit a scale-free or small-world behavior [156, 154, 158]. However, the scale-free parameter, α differs in both the cases – α is 2.51 for the directed graph (close to 2.7 reported in [158, 154] on another reactions dataset) and 3.1 for the hypergraph.

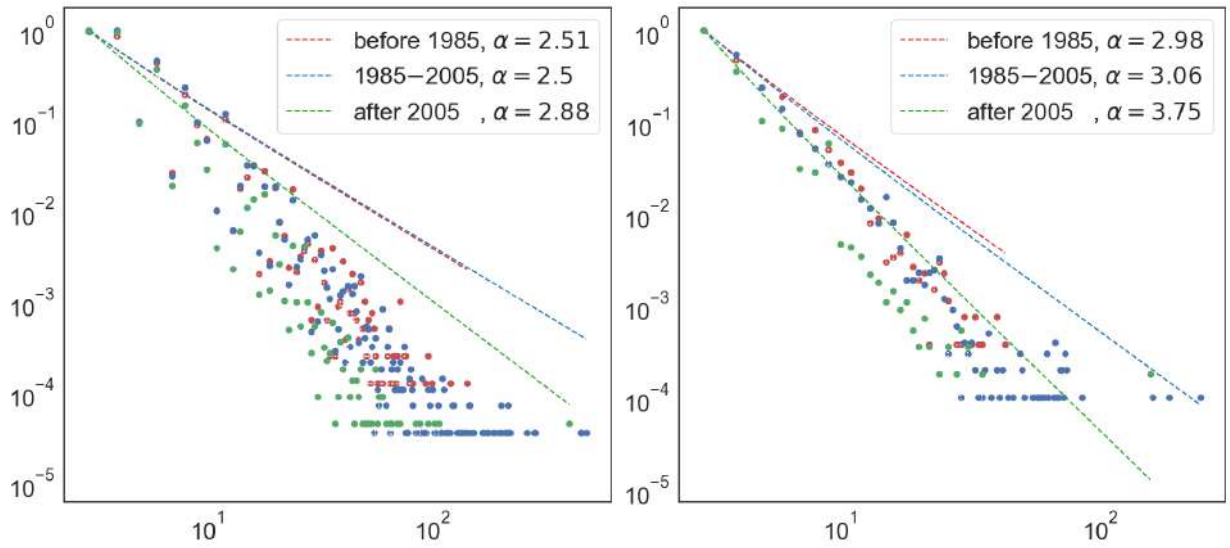
In order to ascertain the difference in α values for the degree distributions, the scale-free parameter is estimated by randomly sub-sampling different fractions of the network in a step-forward manner in time, i.e., by utilizing the reaction year information, reactions were sampled starting from 1976 by sequentially sampling additional reactions from the following years. A $0.1 - 1.0$ fraction of the network was sub-sampled in steps of 0.1 and repeated 10 times to perform bootstrapping and compute the deviations in α. The results are presented in Table 6.5. It is clear from the table that the scale-free distribution is indeed different in the two representations and remains the same irrespective of the fraction of network sub-sampled for estimating the distribution.

Table 6.5: Scale-free distribution parameter values, α, for different fractions of the network sampled using step-forward sampling in time using 10 bootstrapped samples for each fraction

frac	graph α		hyeprgraph α	
	mean	std	mean	std
0.1	2.54	0.0009	3.1	0.0014
0.2	2.54	0.0003	3.18	0.0013
0.3	2.48	0.0008	3.13	0.0021
0.4	2.48	0.0005	3.1	0.0013
0.5	2.47	0.0001	3.02	0.0003
0.6	2.48	0.0004	3.03	0.0014
0.7	2.49	0.0005	3.04	0.0014
0.8	2.5	0.0003	3.07	0.0005
0.9	2.51	0.0001	3.1	0.0001
1.0	2.52	0.00014	2.97	0.0034

Time-evolution of scale-free network property

Next, he time-evolution of the scale-free parameter α was studied by computing it across the three time regimes – before 1985, 1985 – 2005, and after 2005. The degree distributions, power law fit, and the estimated α values for the power law fit are show in Figure 6.8.



(a) Directed graph incoming degrees (b) Hypergraph 'product' degrees

Figure 6.8: Scale-free fit for reactions reported in the three regimes with estimated α in inset.

it is observed that the scale-free parameter α has been increasing over the years with significant increase post 2005, pointing towards accelerated growth nature of the hypergraph network [146] and a similar observation has been made on reactions dataset in [158]. The accelerated growth of the network of chemistry is also evident from the average path length analysis presented in Section 6.2.3.

Inferences from degree distributions analysis

First, it is observed that the degree distributions in both the cases follow a scale-free distribution, pointing towards an underlying mechanism of 'preferential attachment' or 'preferential linking' where new nodes attach to existing nodes in the network with probability proportional to their connectivity or node degrees. Mathematically, preferential attachment is characterized by

$$\Pi(k) \sim k^c \tag{6.5}$$

where $\Pi(k)$ is the probability of a new node attaching to an existing node with degree k, and c is a constant controlling the degree of non-linearity in preferential attachment. This

expression translates to the inference that chemistry growth is largely driven by a relatively small set of highly important molecules that are highly connected (higher degree, k) and they have a higher likelihood of playing a central role in the discovery of new molecules or reactions because of the underlying phenomenon of preferential linking.

Second, for the directed and hypergraph representations, the parameter characterizing the scale-free distributions is higher for the latter. This could be related to chemistry by looking at the concept of 'initial attractiveness' in scale-free networks that assigns a non-zero probability of connecting to an isolated node, given by

$$\Pi(k) = A + k^c \tag{6.6}$$

which ensures that, for non-zero values of A, $\Pi(k) \neq 0$ for disconnected nodes. The presence of A in the expression for preferential attachment $\Pi(k)$ does not affect the scale-free structure of the network but has a direct-impact on the α parameter as,

$$\alpha = 2 + (k1 + A)/k2 \tag{6.7}$$

where $k1, k2$ are constants with values depending on the underlying generating model and A characterizes the initial attractiveness of nodes. Thus, it could be inferred from Equation 6.7 that the initial attractiveness based on the hypergraph representation is higher than that of the directed-graph representation since the former has a higher α of 3.1 characterizing the scale-free distribution compared with α of 2.51 for directed-graph representation. Moreover, the gradually increasing α values for the scale-free distribution in both directed and hypergraph representations indicates that the initial attractiveness has been increasing over time, with the trend being much more evident in the latter representation where α grew from 2.98 in regime 1 to 3.75 in regime 3.

Third, a higher initial attractiveness translates to a higher likelihood of discovering new connections (or reactions) to isolated nodes (rare or complex molecules). Since the initial

attractiveness is the highest and much different in regime 3 (after 2005) than the other two regimes, it could be inferred that in the recent years, there has been an emphasis on the rewiring of existing reactions to create connections between previously disconnected nodes, or the synthesis of rarer molecules. It will become clear from the analysis in the next section on average shortest path length that the major driver of chemistry evolution in the recent years is the rewiring of existing reactions.

6.2.3 Average path length

Average path length comparison

The average separation between the molecules (vertices) in terms of number of reactions (edges) is captured by the average path length of the network. The average path length was computed on the largest connected subgraph for both the representations. Recall from the Section 6.1.3 that for the hypergraph, in order to make a one-to-one comparison, s was chose as 1 to generate a 1-linegraph and compute the 1-distance between nodes to compute the average shortest path length for the hypergraph. The average shortest path lengths for the largest connected subgraph obtained for the two representations for different fraction of nodes sampled from the entire dataset using step-forward sampling is shown in Table 6.6.

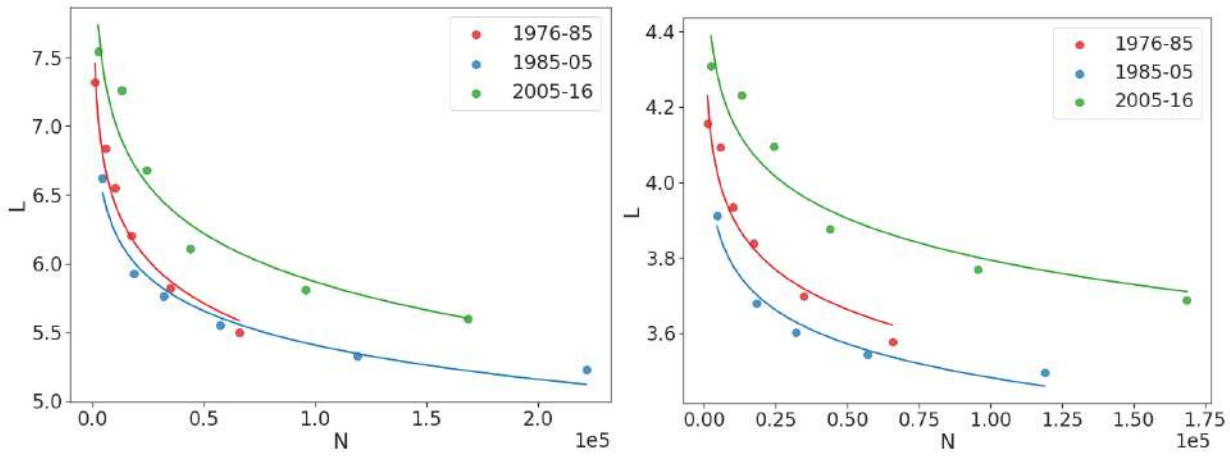
Table 6.6: All pairs shortest path (or APSP) on the entire dataset

Fraction	Reactions	Directed		Hypergraph	
		nodes	APSP	nodes	APSP
1%	4,877	7,528	6.62	7,516	3.99
5%	24,386	26,222	5.98	26,176	3.75
10%	48,772	47,043	5.69	47,069	3.64
20%	97,544	91,903	5.36	91,870	3.52
50%	243,862	209,790	5.16	209,790	3.37[2]
100%	**487,724**	**411,396**	**5.11**	**411,396**	**3.25**[2]

[2]extrapolated values since the network size was prohibitively large for the hypernetX package in python with no C-optimized libraries

Time evolution of Average path length

Similar to the degree distribution analysis, the time-evolution of the average path lengths of the networks was studied across the three time regimes. The average path length as a function of the number of nodes in the network using time-based step-forward sampling is shown in Figure 6.9 for both the representations for different fractions of the networks, namely $1\%, 5\%, 10\%, 20\%, 50\%$ and 100% of the network in each regime.



(a) Graph (b) Hypergraph

Figure 6.9: Average shortest path lengths for various regimes as a function of the number of nodes in the sub-sampled graph.

it is observed that across both the representations, the average path length between the nodes decreases exponentially as the number of nodes in the networks is increased. Moreover, in both the cases, the average path lengths for the time regimes 1 and 2 are very similar to each other but the average path length for regime 3 is significantly higher than those in other two across all values of N. The phenomenon of decreasing path length as number of nodes is increased has been reported in the literature as network densification [160] where the network grows more and more dense over time – this makes sense for the USPTO dataset containing patented reactions where the nodes are mostly sparsely connected and they get more connected over time after either new reactions discovered, or existing nodes become more connected.

In terms of differences, the average path length is much smaller for the hypergraph representation than for the directed graph. This could be an outcome of the frugal representation of hypergraphs where number of edges is the same as number of reactions but that is not the case with graphs. This is one of the major advantages of using hypergraphs – the edges-based analysis has a one to one correspondence with reactions-based analysis, meaning that the separation in terms of hyperedges between nodes corresponds exactly to the separation between molecules in terms of reactions. Therefore, the average separation between nodes in a hypergraph not only differs from a dyadic graph representation, but the separation corresponds to the number of reactions (on average) the separate the nodes in the network.

Inferences from average path lengths analysis

First, it is observed that as expected, the average shortest path length for the directed graph representation is higher than that of the hypergraph representation. This again is an artifact of the directed graph representation which introduces several additional edges for each reaction depending on the number or reactants and products per reaction. On the other hand, in the hypergraph representation, since each hyperedge connects all the molecules taking part in a reaction using a single hyperedge, the separation exactly equals the number of reactions separating any two given nodes. Thus, in the directed graph average shortest path length for the entire network is 5.11 while in the hypergraph it is 3.25.

Second, the average all pairs shortest distance for the hypergraph could be interpreted as separation between nodes (or molecules) in terms of number of reactions. Recall that since each hyperedge corresponds to a unique reaction, there exists a one to one mapping between the number of reactions and the number of hyperedges separating the molecules. Thus, the hypergraph network of organic chemistry indicates that the network of organic chemistry is much more compact than previously understood with nearly 3.25 degrees of separation between molecules, pointing towards an even stronger small-world nature than previously observed with five degrees of separation [156].

Third, in both the cases, the time-evolution of the network suggests that over time, network densification takes place primarily due to the creation of links between existing nodes in the network rather than by the addition (or discovery) of new nodes. A characteristic of network densification is shrinking diameter [160], i.e., the average separation between nodes decreases as the network grows, similar to the exponential decrease in average shortest path length reported in Table 6.6 and Figure 6.9. This phenomenon is observed for both the representations and across time-regimes, pointing towards an underlying process causing the densification. There exist models for explaining such densification such as the community guided attachment similar to preferential attachment but at a bigger community (or cluster) level with separation between the communities [160]. However, the exact quantitative model guiding densification in reaction chemistry network needs further studies. Nevertheless, densification suggests that chemistry has been evolving mostly based on the rewiring of existing reactions (edges) rather than the discovery of completely new molecules (nodes addition), that has brought the molecules closer to each other over time. This is intuitive for the reaction patents dataset that was used since most molecules are initially well separated given that they are patented molecules/reactions which get more connected (reachable) over time due to the discovery of new reactions over the years.

Third, the time-evolution analysis of the average shortest path length in Figure 6.9 suggests that in regime 1 and 2, the average separation between molecules was nearly the same for a given number of nodes, N in the network. However, in regime 3, there was a significant upward shift of average separation across all values of N. This suggests that the time-regime post 2005 is characterized by the discovery of complex chemistry leading to the synthesis of molecules via complex routes that has led to the increase in their average separation, possibly due to significant advances in computational capabilities around this time. This increase in average separation is more evident in the hypergraph representation than the directed-graph representation.

6.2.4 Assortativity

Assortativity comparison

To understand the mixing patterns of nodes in the two network representations, the assortativity values are computed between different node-type (or role) combinations – 'in' and 'out' degree roles for directed graphs and pairwise roles-based node degrees for the annotated hypergraph. Table 6.7 shows the assortativity values for the two representations on the entire network. The assortativity values for the two representations agree qualitatively with each other but differ in terms of their relative strengths. From Table 6.7, it is seen that in the hypergraph representation, the reactant nodes exhibit strong assortative mixing (out-out). On the other hand, the product-product and reactant-product exhibit very weakly assortative or non-assortative behavior pointing towards a lack of degree correlation between such nodes. Owing to the flexibility offered by annotations in the hypergraph, additional

Table 6.7: Assortativity values on the entire dataset

node-pairs	directed graph	hypergraph
in-in	0.0107	0.0734
out-out	0.0049	0.1159
out-in	0.0187	0.0032

assortativity were computed between node roles of reactant/product with roles based on molecular weights and relative SMILES length of molecules, as shown in Tables 6.8 and 6.9. it is observed that reactant-MW_{light} and reactant-$\text{SMILES}_{\text{short}}$ exhibit strong assortative mixing whereas this is not usually the case with other node-role pairs.

Table 6.8: Hypergraph assortativity between reactant & product roles with roles based on molecular weights

	MW_{light}	MW_{medium}	MW_{heavy}
reactant	0.1337	0.0074	0.0061
product	0.0119	0.0003	0.0004

Table 6.9: Hypergraph assortativity between reactant & product roles with roles based on relative SMILES lengths

	SMILES_{short}	SMILES_{medium}	SMILES_{long}
reactant	0.1782	0.0713	0.0015
product	0.0237	0.0083	-0.0009

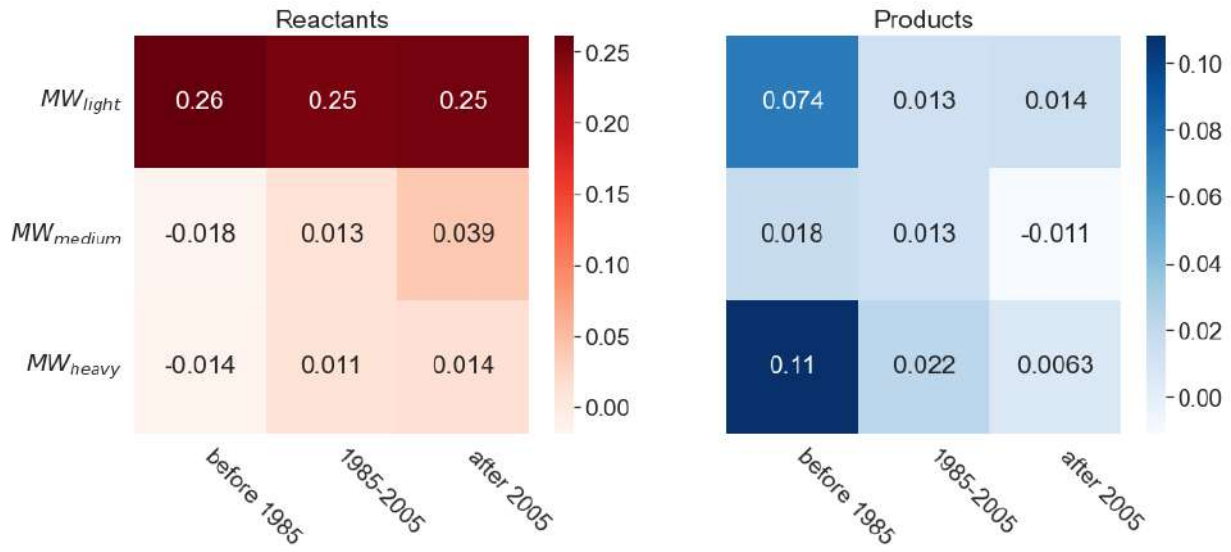
Time evolution of assortativity

To study the evolution of mixing patterns in the network over time, the time-evolution of assortativity was studied during the three time regimes. The assortativity values for the in-in, out-out, and out-in node-role pairs are shown in Table 6.10 below.

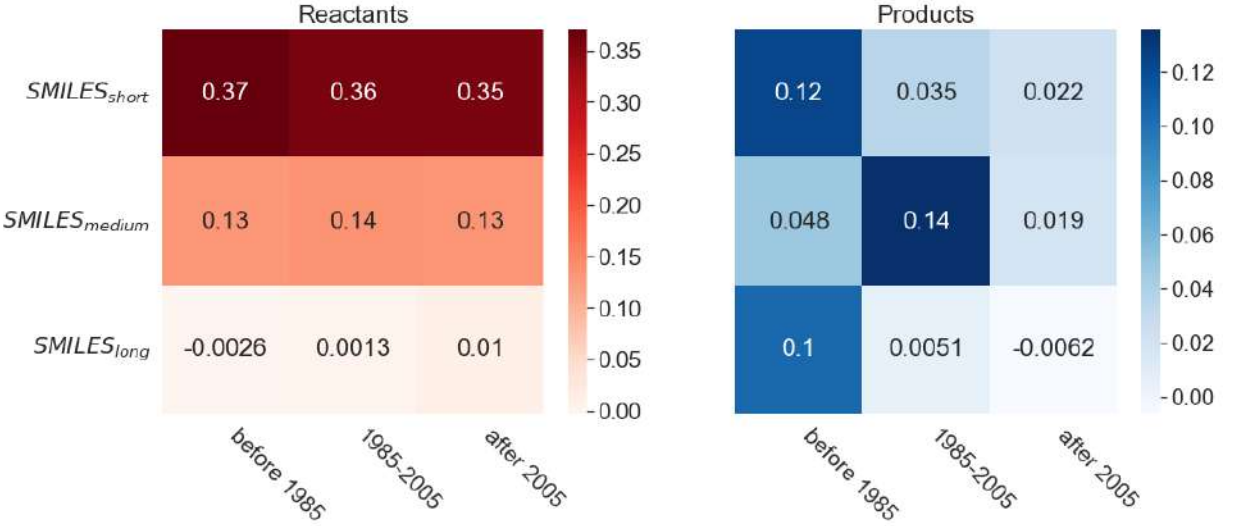
Table 6.10: Time evolution of assortativity for the directed graph and hypergraph for various node-role pairs

node-pairs	Directed graph			Hypergraph		
	before 1985	1985-2005	after 2005	before 1985	1985-2005	after 2005
in-in	0.0630	0.0157	0.0175	0.3623	0.2302	0.1674
out-out	0.0047	0.0042	0.0069	0.2368	0.2241	0.2259
out-in	0.0274	0.0257	0.0286	0.0005	0.0057	0.0076

it is observed from the table above that the directed-graph representation does not show any strong trend in various assortativity values, an observation also reported in [156]. On the other hand, the hypergraph representation shows a decreasing assortativity of in-in nodes over time and an increasing assortativity of out-in nodes. A further analysis on additional assortativity values with different node-role pairs reveal additional trends as shown in Figure 6.10. It is observed that reactants show assortative mixing with nodes with MW_{light} across time regimes, whereas products show assortative mixing with MW_{heavy} before 1985. Similarly, it is also observed that reactants show assortative mixing with nodes with SMILES_{short} and SMILES_{medium} across time regimes, whereas products show assortative mixing with SMILES_{short} and SMILES_{long} before 1985 and with SMILES_{medium} from 1985-2005.



(a) Assortativity between MW and reactant/product roles. Reactants show assortative mixing with nodes with MW_{light} across time regimes, whereas products show assortative mixing with MW_{heavy} before 1985.



(b) Assortativity between relative SMILES length and reactant/product roles. Reactants show assortative mixing with nodes with $\text{SMILES}_{\text{short}}$ and $\text{SMILES}_{\text{medium}}$ across time regimes, whereas products show assortative mixing with $\text{SMILES}_{\text{short}}$ and $\text{SMILES}_{\text{long}}$ before 1985 and with $\text{SMILES}_{\text{medium}}$ from 1985-2005.

Figure 6.10: Time evolution of assortativity for reactants and products with respect to additional node annotations (or roles).

Inferences from assortativity analysis

The assortativity analysis highlights another limitation of the directed-graph representation in terms of obscuring the underlying network characteristics induced by the network wiring scheme. Based on the observations in Table 6.7 for the directed-graph representation, it appears that the network is non assortative or very weakly assortative with respect to all the node-role pairs. It was shown in [156] that this is an artifact of the network preprocessing and the assortativity values change drastically if one chooses to perform network preprocessing to remove parallel edges. On the contrary, the hypergraph representation shows that the network is assortative with respect to certain node-role pairs such as out-out degree assortativity indicating that commonly used reactants tend to take part in reactions together.

Second, due to the flexibility of the hypergraph representation in terms of allowing additional node annotations, additional assortativity analyses are performed with respect to different node-role pairs as shown in Tables 6.8 and 6.9. It was observed that reactants are assortative with molecules of light molecular weight and relatively short/medium SMILES length, highlighting the mixing patterns of reactant nodes in the network. Products, on the other hand, seem to be non-assortative with these properties, thus highlighting the wide spectrum of products with varying degrees of complexity present in the dataset.

An analysis of the time-evolution of assortativity presented in Table 6.10 shows node-mixing trend across time-regimes, with no clear trend in assortativity for directed-graphs. However, from the hypergraph representation it is observed that the reactants exhibit assortative mixing at nearly the same level across time regimes, whereas the products show a decreasing assortativity over time. The latter points towards the general trend in earlier years (regime 1) to discover several different routes for synthesizing a given molecule, which has been decreasing over the years (but still significant) due to the synthesis of new products molecules with different chemistry.

Finally, based on the time-evolution of assortativity with respect to additional node annotations in Figure 6.10, it is observed that reactants are assortative at the same level with

heavy molecular weight as well as relative molecular complexity across time regimes, with decreasing assortativity as the molecular weight or complexity is increased. Products on the other hand, show a positive assortativity before 1985 with heavy and complex molecules, in regime 2 assortative with medium complexity, and non-assortative in regime 3 with all roles. The latter indicates towards the diversity of products synthesized in the recent years.

6.3 Additional hypergraph statistics

Even though many dyadic network properties could also be defined equivalently for hypergraphs, sometimes it is necessary to work with the directed graph framework for reasons among – interpretability from a traditional graph-theoretic standpoint, easy availability of tools for computation of dyadic properties, or aversion towards adopting hypergraphs due to their seemingly high complexity. The annotated hypergraph could, therefore, be projected as a directed graph with edge-weights defined using a role-interaction kernel [159]. The role-interaction kernel defines the mapping of the annotated hypergraph to a projected-directed graph, that maps various nodes to annotations in the hypergraph using weighted edges. The following three kernels are used:

- $R1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$: each hyperedge split into multiple weighted directed edges from reactants to products each with weight 1; emphasis is on forward reactions only
- $R2 = \begin{bmatrix} 0 & 0.75 \\ 0.25 & 0 \end{bmatrix}$: each hyperedge split into multiple weighted directed edges with directed edges from reactants to products with weight 0.75 and also directed edges in the reverse direction (from products to reactants) with weight 0.25; unequal emphasis on forward and inverse reactions
- $R3 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$: each hyperedge split into multiple weighted directed edges in the reverse direction (from products to reactants) each with weight 1; emphasis on inverse reactions

only

Using such projected dyadic graphs, two additional studies are preformed on the entire network – first, a PageRank [148] analysis of reaction nodes to identify the most important molecules, and second, a graph-based community-detection (or clustering) [149] to identify clusters in the reaction networks based on their connectivity patterns.

6.3.1 PageRank analysis

The PageRank algorithm was originally proposed for ranking of webpages on the internet [148] based on the number and quality of links to webpages and is based on a random-surfer model that performs random walks along incoming and outgoing edges from webpages. A page that has a higher likelihood of being visited by a random surfer is therefore considered more important by PageRank, thus requiring both higher connectivity as well as connectivity to other important webpages for higher PageRank.

Extending the idea of PageRank to chemical reactions and molecules, the set of molecules that are most important based on their connectivity (high reactivity) as well as their connectivity to other important molecules (chemical importance due to ease of synthesizability or criticality for other compounds) could be identified. Thus, a molecule with high PageRank in a network of chemical reactions should be crucial both from a reactivity/synthesizability as well as reachability/criticality standpoint. In contrast, a molecule with merely the highest degree does not say much about the molecule except that the molecule participates in many reactions.

Using the three role-interaction kernels – R1, R2 and R3 defined above, the PageRank and degree centrality of nodes in the resulting network defined as d_v/d_{max} where d_v is the degree of node v and d_{max} is the maximum degree across all nodes in the network are computed. Since PageRank and degree centrality are two different measures, their absolute values should not be compared and only the relative values or ranked order of molecules should be compared. The top-5 molecules based on PageRank and degree centralities computed using

the weighted directed reaction networks obtained using different role-interaction kernels are shown in Figure 6.11.

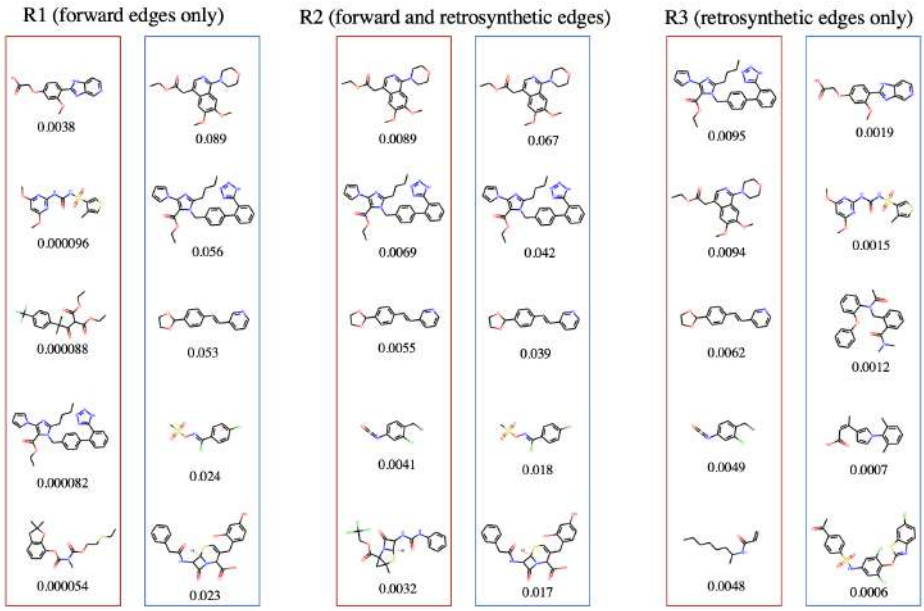


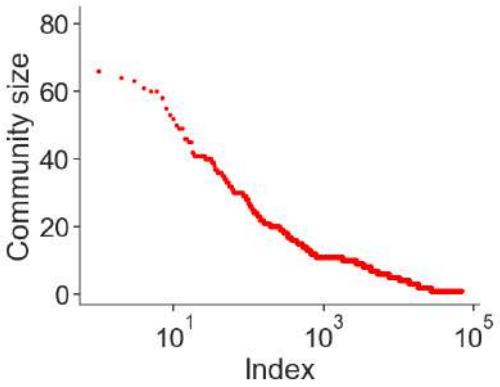
Figure 6.11: PageRank and degree centrality analysis for the three role interaction kernels with $R1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $R2 = \begin{bmatrix} 0 & 0.75 \\ 0.25 & 0 \end{bmatrix}$, and $R3 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$ corresponding to forward edges only, forward and retrosynthetics edges, and retrosynthetic edges only.

Based on the above ranked order of molecules, it is first observed that the molecules that are important from a PageRank standpoint are not the same as those important from a degree centrality standpoint. Second, across the role interaction kernels, the ranked order changes, i.e., molecules critical based on R1 kernel-based projection (forward edges) of hypergraph differ from those based on R3 kernel-based projection (retrosynthetic edges). This highlights the flexibility of the hypergraph reaction representation in incorporating custom importance for forward and retrosynthetic reaction directions through role-interaction kernels. Such an analysis of molecular importance in a reaction network would have application in optimizing reaction networks, designing robust supply chain networks, and performing efficient product design.

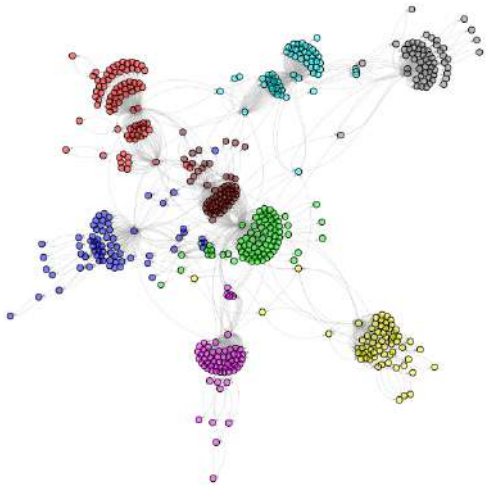
6.3.2 Community detection analysis

To study the formation of communities or clusters in the reaction network based on the mutual connectivity patterns and node-densities, graph-based clustering is performed on the network of reactions. The Leiden algorithm [149] is used to perform optimal graph partitioning that results in well-connected set of dense nodes in the network (called communities) and is a suitable algorithm for weighted, directed networks. For this study, the R2 role-interaction kernel is used to preserve both forward and retrosynthetic edges but with unequal weights in the network. The applications of such a graph-based community detection exercise is to get a general sense of the distribution and connectivity patterns of reactions in a large reactions dataset and understand the possible different types of reactions in the absence of any other information about the reactions. Note that the projected hypergraph is a dyadic, weighted directed graph obtained by using a role-interaction kernel that decomposes a hyperedge into a set of weighted directed edges. Community detection is performed on such projected hypergraph. The alternative is to perform clustering directly using the hypergraph representation. However, given the scale of the hypergraph network of organic chemistry, the current clustering methods are computationally prohibitive.

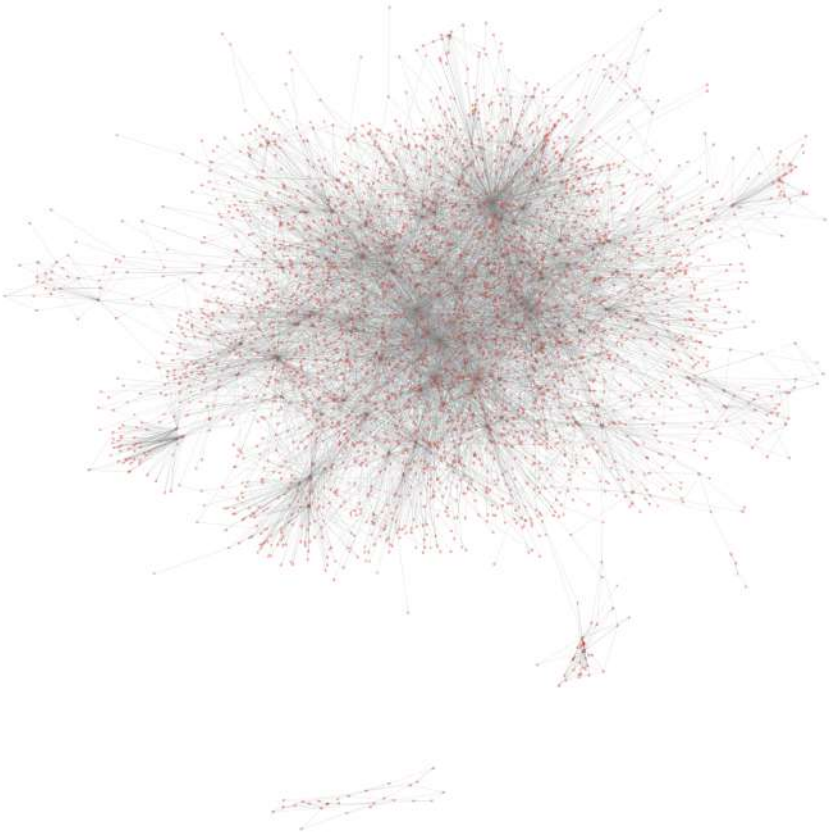
For the entire network, the Leiden algorithm identifies nearly $65,000$ communities with a size-distribution as shown in Figure 6.12(a); the top-8 largest communities are shown in Figure 6.12(b) with different color for each identified community, and the top-100 communities are show in Figure 6.12(c).



(a) Size distribution of identified communities



(b) Top-8 largest communities with nodes in each community in a different color



(c) Top-100 largest communities indicating showing clear regions of high and low densities along with an island community disconnected from the rest of the network

Figure 6.12: Community detection results on the weighted projected directed graph obtained using role-interaction kernel R2.

It is observed from Figure 6.12(a) that most of the communities are really small in size consisting of less than 10 reactions, whereas there are around 8 biggest communities containing over 60 reactions in each of them, as shown in Figure 6.12(b). The close-knit nature of these communities point towards a possible segregation of different types of reactions just based on their connectivity patterns and the nodes (molecules) that take part in those reactions. This idea is utilized to perform reaction type classification in the next section. Finally, the top-100 communities visualized in Figure 6.12 shows clear regions of high density with highly connected and localized clusters, and regions of low density further apart from the biggest clusters. In addition, it is also observed that there is a cluster that is completely separated from all the other communities and is therefore an *island* community. The existence of core-periphery regions in the reaction network was also shown in [155] but the analysis was not based on graph clustering but on identifying strongly connected components in the network by representing reactions using a single directed edge from the heaviest reactant to the heaviest product in each reaction. In the community detection algorithm, the directionality as well as the weights of the edges are taken into account, making it more flexible and the results more generalizable.

6.4 Application in reaction class prediction problem

In the foregoing sections, it is shown how the hypergraph representation could be used to uncover hidden insights contained in large reactions datasets and study their time-evolution through network-theoretic properties. In this section, the usefulness of the hypergraph representation in capturing the context of reactions and thereby their reaction type or class is demonstrated. We, therefore, use the hypergraph representation in the reaction-type classification problem where the objective is to estimate the reaction class from a given set of reactants and products. This problem has practical applications in retrosynthetic planning where several different routes could be eliminated just by knowing the possible reaction types. The other problems where such a problem would find significance is the reaction feasibility

estimation problem where the objective is to estimate the feasibility of a reaction given the possible participating molecules. Other studies that have proposed data-driven frameworks for reaction classification problems are [161, 162, 163].

6.4.1 Dataset description

For this problem, since reaction class information for reactions is required, a subset of the USPTO reactions dataset that is typically used for retrosynthesis problem, containing about 50K reactions annotated with their corresponding reaction class from 10 possible classes is used. The equivalent reaction hypergraph network for this dataset is generated. The largest connected component in the hypergraph is used since the hyperedge (or reaction) embedding framework used for representing reactions subsequently in the classification framework is dependent on the connectivity and neighborhood contextual information. The distribution of reactions across different reaction classes in the sub-hypergraph is shown in Table 6.11 below.

Table 6.11: Distribution of reactions across different reaction classes in the largest connected subcomponent

Rxn class	Rxn name	Num rxns
1	Heteroatom alkylation and arylation	11,526
2	Acylation and related processes	8,488
3	C–C bond formation	3,909
4	Heterocycle formation	588
5	Protections	646
6	Deprotections	760
7	Reductions	459
8	Oxidations	305
9	Functional group interconversion (FGI)	1,168
10	Functional group addition (FGA)	196

6.4.2 Reaction embeddings using random hyperwalks

To perform reaction classification by training a data-driven classifier, numeric representations for reactions that are generated from their hypergraph representations are need to be used as features to train a classifier. Hyperedge (or reaction) embeddings are generated by adapting the deep hyperedges framework [151] and modifying it to incorporate the contextual information contained in chemical reactions, as explained in the pseudocode provide in Algorithm **??**.

The hyperedge embeddings are generated by performing random hyperwalks that capture the co-member information in each vertex by traversing hyperedges in the hypergraph network of chemical reactions. For each hyperedge, the hyperwalk starts at a randomly selected reactant node that is part of the current hyperedge, and either hops to a node in the adjacent hyperedge or stays in the current hyperedge to select another reactant node in the current hyperedge. This is repeated until the desired length of the hyperwalk is achieved. The adjacent hyperedge traversal is done only with respect to reactants since this would – first, differentiate reactants from products, and second, mimic chemistry more realistically where only those reactions are accessible where either the current reactants participate as reactants or the product of the current reaction participates as reactant. Such a random hyperwalk would closely mimic a chemist performing experiments randomly.

Formally, a node v_m is selected at random with the annotation 'reactant' in a hyperedge e_i. The probability of traversing an adjacent hyperedge is inversely proportional to the cardinality of the current vertex; i.e. $p = min(\frac{\alpha}{|v_m|} + \beta, 1)$ where α and β are tunable hyperparameters and $| v_m |$ is the cardinality of the vertex v_m. As in a random walk, if p is less than a randomly generated number, the traversal is performed to an adjacent hyperedge; otherwise the current hyperedge is added to the random walk and the next is chosen randomly from the adjacent hyperedges of the current vertex v_m. For each hyperedge e_i, 50 random walks of length 50 each are constructed. Examples of such hyperedge random walks on the four example reactions in Equation 6.1 is shown in Table 6.12. The hyperwalks are

then embedded into dense vectors of dimension R^{256} using skip-gram approach for generating embeddings [21]. At the end of the hyperedge embedding exercise, a 256 dimensional vector for each hyperedge in the network would be generated.

The pseudocode for the hyperwalk generating algorithm is presented in Algorithm **??** and example hyperwalks using the example set of four reactions in Equation 6.1 is shown in Table 6.12. A 2D visualization of the resulting 256-dimensional hyperedge embeddings on the entire dataset of reactions is visualized in Figure 6.13.

Table 6.12: Two example hyperwalks generated for each reaction (hyepredge) in the example set of reactions. For each walk, $v_i \xrightarrow{e_k} v_j$ represents a walk along hyperedge e_k via nodes v_i and v_j. The hyperwalks for each hyperedge are the sequential collection of such e_k's starting at that hyperedge.

$\mathbf{H_{id}}$	**Hyperwalk**
0	$B, 0 \xrightarrow{1} C \xrightarrow{2} C \xrightarrow{2} C \xrightarrow{3} D \xrightarrow{3} A \xrightarrow{0} A \xrightarrow{3} A \xrightarrow{3} A \xrightarrow{3} A$
	$B, 0 \xrightarrow{1} B \xrightarrow{1} C \xrightarrow{2} C \xrightarrow{2} C \xrightarrow{3} C \xrightarrow{3} C \xrightarrow{1} C \xrightarrow{2} C \xrightarrow{1} C$
1	$E, 1 \xrightarrow{2} E \xrightarrow{1} C \xrightarrow{2} C \xrightarrow{1} C \xrightarrow{2} D \xrightarrow{3} D \xrightarrow{3} D \xrightarrow{2} E \xrightarrow{1} E$
	$B, 1 \xrightarrow{1} E \xrightarrow{3} E \xrightarrow{2} E \xrightarrow{1} E \xrightarrow{1} E \xrightarrow{2} D \xrightarrow{3} A \xrightarrow{3} A \xrightarrow{0} A$
2	$E, 2 \xrightarrow{1} B \xrightarrow{1} B \xrightarrow{0} B \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{3} D \xrightarrow{2} D \xrightarrow{2} D$
	$C, 2 \xrightarrow{2} C \xrightarrow{1} C \xrightarrow{3} C \xrightarrow{3} C \xrightarrow{2} E \xrightarrow{1} E \xrightarrow{2} C \xrightarrow{2} C \xrightarrow{3} A$
3	$A, 3 \xrightarrow{3} A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{1} B \xrightarrow{0} A \xrightarrow{3} D \xrightarrow{3} D$
	$C, 3 \xrightarrow{2} C \xrightarrow{2} E \xrightarrow{2} E \xrightarrow{3} C \xrightarrow{1} C \xrightarrow{3} C \xrightarrow{1} B \xrightarrow{0} A \xrightarrow{0} A$

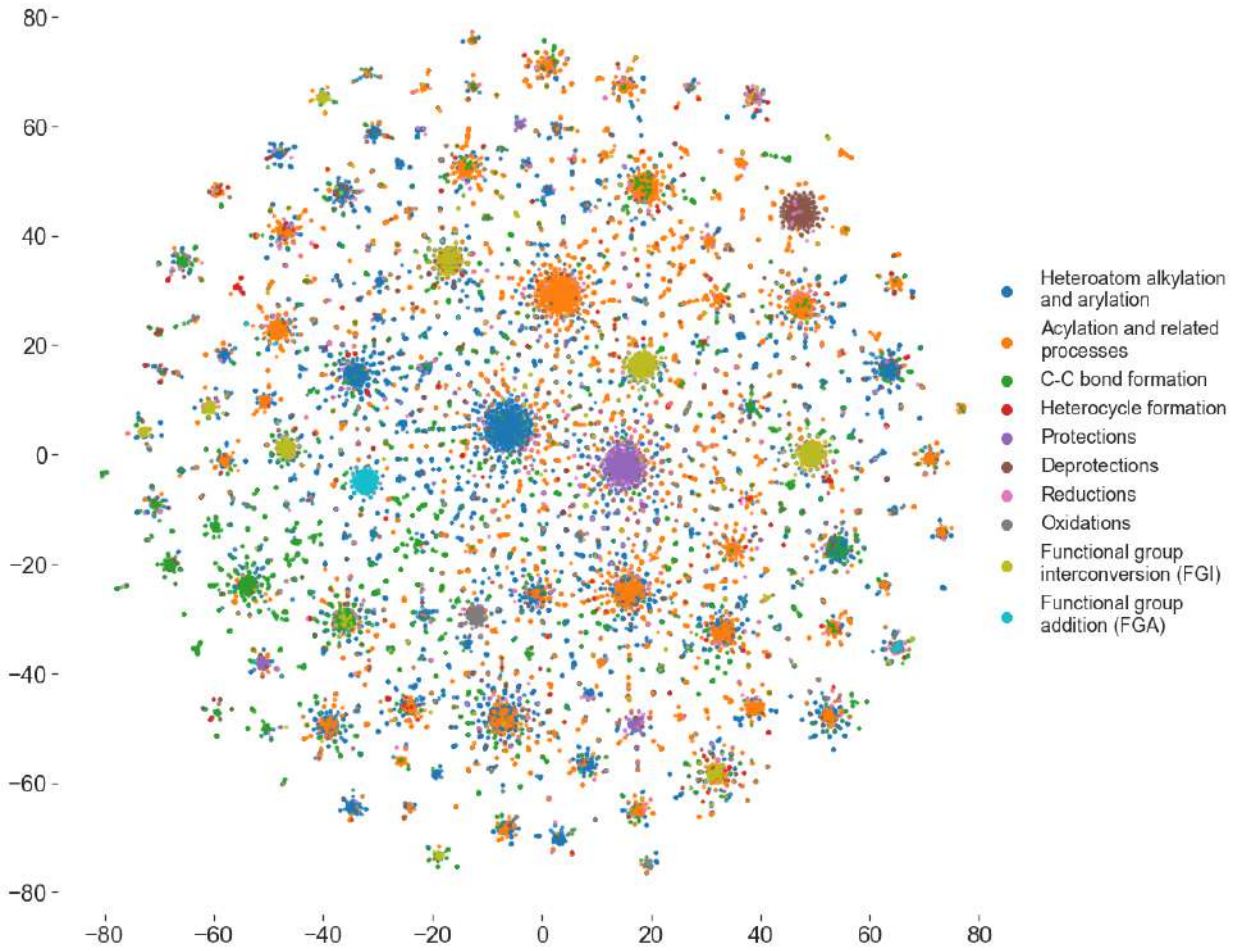
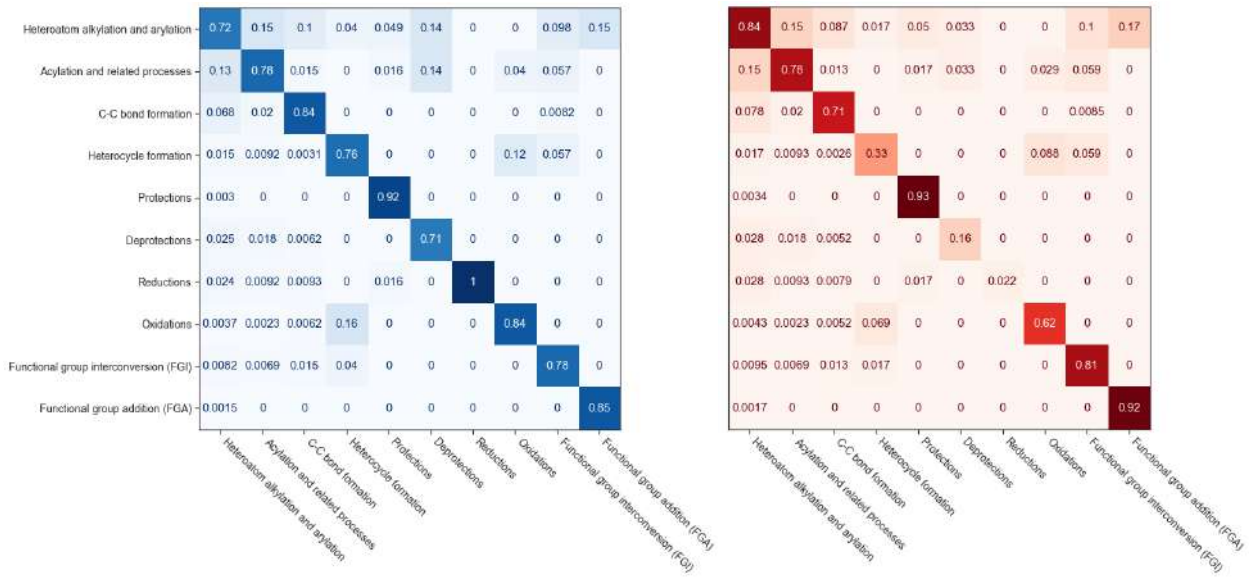


Figure 6.13: A 2D t-SNE projection of the 256-dimensional hyperedge embeddings

6.4.3 Reaction class prediction results

To predict the reaction classes, a one-vs-rest classifier is trained based on support vector machines (SVM) that learns a multi-class classification decision boundary. A randomized cross validation search strategy to perform hyperparameter tuning of the SVM model with a radial basis function. A detailed description of the SVM model and the mathematical framework that underlies it is provided in [88].

The precision and recall metrics for each of the reaction classes computed using the test-set containing unseen reactions at the training stage are shown in Figure 6.14 below.



(a) (b)

Figure 6.14: Performance metrics for the multi-class reaction classification on the test-set. (a) Precision, (b) Recall.

From the results above, it is observed that the trained model accurately predicts the reaction class for most of the reaction classes except for the reaction classes – reductions, deprotections, and heterocycle formation. The precision metrics across all 10 reaction classes shown in Figure 6.14(a) highlights the model's high precision in identifying the correction reaction class. However, since the recall shown in Figure 6.14(b) is lower for the three underperforming classes, there could be overlapping reaction classes in the feature (embedding) space. This is indeed observed for these classes in 2D visualization of the learned embeddings in Figure 6.13. The separation between various reaction classes could be addressed in future by also incorporating additional molecular descriptors that, in combination with the connectivity-specific embeddings, would more accurately distinguish the different reaction types. Nevertheless, the embeddings generated just based on the reactions and node-connectivity information in the hypergraph representation seems to have separated a majority of the reaction types into distinct clusters, consequently resulting in the model learning to predict them accurately. This again highlights the ability of hypergraphs to capture reaction context accurately.

6.5 Conclusions

Network theory offers natural tools and techniques for understanding the growth of chemistry over time by representing reactions as time-evolving real-world networks. Though most of the work in this area has been done using a dyadic graph representation, a hypergraph representation with hyperedges between nodes for representing reactions is a more natural, intuitive, and flexible representation that allows for the incorporation of additional reaction context.

It is shown that the hypergraph representation is more flexible, allows for incorporation of reaction-specific node context, and facilitates one-to-one correspondence of network properties with chemistry. Detailed network statistics of the resulting hypergraph network of organic chemistry were computed and the time evolution of these properties was studied. As with several previous studies, it is observed that the network exhibits a scale-free behavior with preferential attachment of nodes, has small average path length indicative of small-world nature, and shows assortative mixing with respect to certain node types. All the network statistics presented, namely, degree distributions, average path length, assortativity or degree correlations, PageRank analysis, and community detection, were correlated with chemistry inferences that were drawn from such analysis. In addition, it was discovered that the network exhibits the phenomenon of initial attractiveness and network densification as chemistry evolves over time time.

To demonstrate the AI-applications of the hypergraph representation of chemical reactions, reaction classification was performed using embeddings generated from chemistry-informed random walks on hyperedges. The embeddings resulted in well-separated clusters for different reaction classes and consequently, resulted in accurate reaction classification results. Future work involves extending this study to diverse (and possibly bigger) datasets across various subdomains, incorporating additional molecular descriptors for generating hyperedge embeddings for reaction classification, utilizing the results in a retrosynthestic planning framework, and performing hyperedge prediction to discover new reactions.

Epilogue

Summary of research

Several domain-informed language modeling frameworks were presented for a variety of central problems in process systems engineering. Although these problems spanned several different scales, language models combined with important domain knowledge were shown to offer several advantages over purely data-driven or purely process knowledge-based solutions across all the problems. Such integration of domain knowledge helped sidestep the necessary condition of availability of large amounts of data to the scale not encountered in PSE for efficient training of language models. More importantly, this resulted in the development of systems that suffer from a mismatch between data-driven predictions and process knowledge, which might otherwise lead to costly mistakes in process systems engineering. Integration of the rich domain knowledge developed over several decades in PSE and computational advantages of language models is therefore shown to be necessary to develop systems that are more reliable, scalable, and explainable while also solving problems in an intelligent and consistent manner. Highlights of the major achievements and contributions of this thesis are outlined below.

- *Molecular structure-to-property prediction*: A molecular representation learning framework, grammar2vec, was developed for learning efficient, dense vector representations of molecules and is analogous to the word2vec framework in natural language. Grammar2vec was shown to result in chemistry-rich molecular descriptors, and more ac-

curate, and more interpretable machine learning models for thermodynamic property prediction.

- *Computer-aided forward and retrosynthesis reaction prediction*: Computer-aided forward and retrosynthesis (or inverse) reaction prediction was formulated as sequence-to-sequence and tree-to-sequence modeling problems, respectively. SMILES grammar, analogous to context-free grammar in natural language, was shown as a promising alternative representation compared to the commonly used text-based SMILES representation. Grammar representations were shown to result in lower conditional entropy for the prediction task, relatively simpler model architectures, and reduced grammatically invalid predictions. The retrosynthesis prediction was formulated as a tree-to-sequence modeling problem, and the transformer model was adapted on the basis of group contribution-like tree convolution operations. The resulting model achieved nearly state-of-the-art performance with relatively simpler models and was in agreement with reaction chemistry principles, thus highlighting the advantages of incorporating chemistry knowledge in language models.

- *Process flowsheet representation and generation*: A novel framework, *eSFILES* or extended Simplified Flowsheet-Input Line-Entry Scheme, was developed to hierarchically represent chemical flowsheet information. The developed framework efficiently captured flowsheet information at different levels using purely text-based flowsheet representations (akin to SMILES strings for molecules), flowsheet hypergraph-based representations, and process ontology connected to hypergraph-based representations. The *eSFILES* framework enabled the integration of process knowledge and artificial intelligence techniques to solve flowsheet synthesis and design problems using hybrid AI methods. Linguistics-inspired flowsheet syntax grammar rules were also developed to ensure a correct and consistent flowsheet representation.

- *Automated pharmaceutical information extraction*: SUSIE, or Schema-based Unsu-

pervised Semantic Information Extraction framework, was developed that facilitated end-to-end relevant information extraction from pharmaceutical documents. A custom pharmaceutical drug-developed ontology was also developed to capture domain-specific concepts. SUSIE comprised domain ontologies for representing important concepts, a weak supervision framework for transforming unlabeled datasets to labeled datasets, a fine-tuned BioBERT model to learn a generalizable pattern for handling new documents, contextualization modules to capture relevant context associated with important entities, and a relation extraction approach for automatically generating knowledge graphs.

- *Network of organic chemistry and reaction classification*: A hypergraph representation for studying the network of organic chemistry was shown to be a more natural, intuitive, and flexible representation that allowed the incorporation of additional reaction context. Detailed network statistics were computed, and the time evolution of these properties was studied. The network of organic chemistry was observed to exhibit a scale-free behavior with preferential attachment of nodes, a small average path length indicative of the small-world nature, and assortative mixing with respect to certain node types. Reaction classification was performed using embeddings generated from chemistry-informed random walks on hyperedges. The embeddings resulted in well-separated clusters for different reaction classes, and consequently, resulted in accurate reaction classification results.

Future recommendations and outlook

Given that the cost of a mistake in PSE applications is higher by orders of magnitude compared to other domains like game playing, natural language translation, or text analysis, a fundamental integration of process knowledge with data-driven approaches is necessary to ensure its success from a practical standpoint. In the current era of large language models (LLMs) comprising billions of model parameters and trained on terabytes of data, an impor-

tant differentiator that is often overlooked is that such data-driven approaches need to be adapted appropriately before they can be used in engineering domains like process systems engineering. Our domain does not lack a formal understanding of the underlying process, unlike computer science, for example, in tasks like image recognition. Thus, ignoring decades of fundamental domain knowledge and well-established principles would lead to suboptimal, difficult, and *reinventing the wheel* scenarios where a machine learning approach is praised for rediscovering fundamental principles or governing equations that are well known and widely understood.

Going forward, a broader integration of the underlying physics, chemistry, and formal theory with data-driven models is required. This would need an appropriate knowledge representation and imposing prior knowledge-based constraints – an appropriate knowledge representation is crucial to ensure the problem formulation, underlying governing principles, and so on, are captured appropriately; imposing prior knowledge-based constraints would require a deeper understanding of the model architectures used in data-driven models so that they could be adapted to impose domain-based restrictions. It is hoped that this thesis would provide guidelines for the development of hybrid AI systems through the various domain knowledge integration frameworks presented for multiscale problems in process systems engineering.

References

[1] R. Sargent, "Advances in modelling and analysis of chemical process systems," *Computers & chemical engineering*, vol. 7, no. 4, pp. 219–237, 1983.

[2] V. Venkatasubramanian, "The promise of artificial intelligence in chemical engineering: Is it here, finally?" *AIChE Journal*, vol. 65, no. 2, pp. 466–478, 2019. eprint: `https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.16489`.

[3] E. N. Pistikopoulos *et al.*, "Process systems engineering–the generation next?" *Computers & Chemical Engineering*, vol. 147, p. 107 252, 2021.

[4] R. Gani *et al.*, "A multi-layered view of chemical and biochemical engineering," *Chemical Engineering Research and Design*, vol. 155, A133–A145, 2020.

[5] V. Venkatasubramanian and V. Mann, "Artificial intelligence in reaction prediction and chemical synthesis," *Current Opinion in Chemical Engineering*, vol. 36, p. 100 749, 2022.

[6] D. P. Visco Jr, R. S. Pophale, M. D. Rintoul, and J.-L. Faulon, "Developing a methodology for an inverse quantitative structure-activity relationship using the signature molecular descriptor," *Journal of Molecular Graphics and Modelling*, vol. 20, no. 6, pp. 429–438, 2002.

[7] R. Gani, "Group contribution-based property estimation methods: Advances and perspectives," *Current Opinion in Chemical Engineering*, vol. 23, pp. 184–196, 2019.

[8] Z Zhang, H Li, H Chang, Z Pan, and X Luo, *Machine learning predictive framework for co2 thermodynamic properties in solution. j co2 util 26: 152–159*, 2018.

[9] K. K. Yalamanchi *et al.*, "Machine learning to predict standard enthalpy of formation of hydrocarbons," *The Journal of Physical Chemistry A*, vol. 123, no. 38, pp. 8305–8313, 2019.

[10] D. A. Saldana, L. Starck, P. Mougin, B. Rousseau, N. Ferrando, and B. Creton, "Prediction of density and viscosity of biofuel compounds using machine learning methods," *Energy & fuels*, vol. 26, no. 4, pp. 2416–2426, 2012.

[11] A. S. Alshehri, A. K. Tula, F. You, and R. Gani, "Next generation pure component property estimation models: With and without machine learning techniques," *AIChE Journal*, e17469, 2021.

[12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[13] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[14] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988. eprint: `https://pubs.acs.org/doi/pdf/10.1021/ci00057a005`.

[15] K. Yang *et al.*, "Analyzing learned molecular representations for property prediction," *Journal of chemical information and modeling*, vol. 59, no. 8, pp. 3370–3388, 2019.

[16] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," 2017. arXiv: `1703.01925 [stat.ML]`.

[17] L. S. Shapley *et al.*, "A value for n-person games," 1953.

[18] A. S. Hukkerikar, B. Sarup, A. Ten Kate, J. Abildskov, G. Sin, and R. Gani, "Group-contribution+ (gc+) based estimation of properties of pure components: Improved property estimation and uncertainty analysis," *Fluid Phase Equilibria*, vol. 321, pp. 25–43, 2012.

[19] S. Honda, S. Shi, and H. R. Ueda, "Smiles transformer: Pre-trained molecular fingerprint for low data drug discovery," *arXiv preprint arXiv:1911.04738*, 2019.

[20] S. Jaeger, S. Fulle, and S. Turk, "Mol2vec: Unsupervised machine learning approach with chemical intuition," *Journal of chemical information and modeling*, vol. 58, no. 1, pp. 27–35, 2018.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, vol. 26, 2013, pp. 3111–3119.

[22] G. B. Goh, N. O. Hodas, C. Siegel, and A. Vishnu, "Smiles2vec: An interpretable general-purpose deep neural network for predicting chemical properties," *arXiv preprint arXiv:1712.02034*, 2017.

[23] Z. Gong, Y. Wu, L. Wu, and H. Sun, "Predicting thermodynamic properties of alkanes by high-throughput force field simulation and machine learning," *Journal of chemical information and modeling*, vol. 58, no. 12, pp. 2502–2516, 2018.

[24] G. A. Pinheiro, J. Mucelini, M. D. Soares, R. C. Prati, J. L. Da Silva, and M. G. Quiles, "Machine learning prediction of nine molecular properties based on the smiles representation of the qm9 quantum-chemistry dataset," *The Journal of Physical Chemistry A*, vol. 124, no. 47, pp. 9854–9866, 2020.

[25] J. Ding *et al.*, "Machine learning for molecular thermodynamics," *Chinese Journal of Chemical Engineering*, vol. 31, pp. 227–239, 2021.

[26] M. N. Aldosari, K. K. Yalamanchi, X. Gao, and S. M. Sarathy, "Predicting entropy and heat capacity of hydrocarbons using machine learning," *Energy and AI*, vol. 4, p. 100 054, 2021.

[27] N. Chomsky, "On certain formal properties of grammars," *Information and control*, vol. 2, no. 2, pp. 137–167, 1959.

[28] V. Mann and V. Venkatasubramanian, "Predicting chemical reaction outcomes: A grammar ontology-based transformer framework," *AIChE Journal*, vol. 67, no. 3, e17190, 2021. eprint: `https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.17190`.

[29] V. Mann and V. Venkatasubramanian, "Retrosynthesis prediction using grammar-based neural machine translation: An information-theoretic approach," *Computers & Chemical Engineering*, vol. 155, p. 107 533, 2021.

[30] R. Rehurek and P. Sojka, "Gensim–python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[31] Y. Su, Z. Wang, S. Jin, W. Shen, J. Ren, and M. R. Eden, "An architecture of deep learning in qspr modeling for the prediction of critical properties using molecular signatures," *AIChE Journal*, vol. 65, no. 9, e16678, 2019.

[32] V. N. Vapnik, "The nature of statistical learning," *Theory*, 1995.

[33] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[34] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2242–2251.

[35] L. Merrick and A. Taly, "The explanation game: Explaining machine learning models using shapley values," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2020, pp. 17–38.

[36] R. Rodriguez-Perez and J. Bajorath, "Interpretation of machine learning models using shapley values: Application to compound potency and multi-target activity predictions," *Journal of computer-aided molecular design*, vol. 34, no. 10, pp. 1013–1026, 2020.

[37] M. Smith and F. Alvarez, "Identifying mortality factors from machine learning using shapley values–a case of covid19," *Expert Systems with Applications*, vol. 176, p. 114 832, 2021.

[38] C. E. Shannon, "A mathematical theory of communications," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.

[39] S. Szymkuć *et al.*, "Computer-assisted synthetic planning: The end of the beginning," *Angewandte Chemie International Edition*, vol. 55, no. 20, pp. 5904–5937, 2016,
A comprehensive expert-system with human encoded reaction rules combined with search strategies, heuristics, and synthetic route correction for synthesis planning and for suggesting reaction mechanisms.

[40] V. Mann, R. Gani, and V. Venkatasubramanian, "Group contribution-based property modeling for chemical product design: A perspective in the ai era," *Fluid Phase Equilibria*, vol. 568, p. 113 734, 2023.

[41] D. A. Pensak and E. J. Corey, "Lhasa—logic and heuristics applied to synthetic analysis," in ACS Publications.

[42] J. Law *et al.*, "Route designer: A retrosynthetic analysis tool utilizing automated retrosynthetic rule generation," *Journal of chemical information and modeling*, vol. 49, no. 3, pp. 593–602, 2009.

[43] C. W. Coley, L. Rogers, W. H. Green, and K. F. Jensen, "Computer-assisted retrosynthesis based on molecular similarity," *ACS central science*, vol. 3, no. 12, pp. 1237–1245, 2017.

[44] C. A. Nicolaou, I. A. Watson, M. LeMasters, T. Masquelin, and J. Wang, "Context aware data-driven retrosynthetic analysis," *Journal of Chemical Information and Modeling*, vol. 60, no. 6, pp. 2728–2738, 2020.

[45] B. Liu *et al.*, "Retrosynthetic reaction prediction using neural sequence-to-sequence models," *ACS Central Science*, vol. 3, no. 10, pp. 1103–1113, 2017, PMID: 29104927. eprint: `https://doi.org/10.1021/acscentsci.7b00303`.

[46] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon *et al.*, Eds., Curran Associates, Inc., 2017, pp. 5998–6008.

[47] P. Schwaller *et al.*, "Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction," *ACS central science*, vol. 5, no. 9, pp. 1572–1583, 2019.

[48] J. Nam and J. Kim, "Linking the neural machine translation and the prediction of organic chemistry reactions," *arXiv preprint arXiv:1612.09529*, 2016. arXiv: 1612.09529 [cs.LG].

[49] W. Jin, C. W. Coley, R. Barzilay, and T. Jaakkola, *Predicting organic reaction outcomes with weisfeiler-lehman network*, 2017. arXiv: 1709.04555 [cs.LG].

[50] C. W. Coley *et al.*, "A graph-convolutional neural network model for the prediction of chemical reactivity," *Chemical science*, vol. 10, no. 2, pp. 370–377, 2019.

[51] P. Karpov, G. Godin, and I. V. Tetko, "A transformer model for retrosynthesis," in *International Conference on Artificial Neural Networks*, Springer, 2019, pp. 817–830.

[52] I. V. Tetko, P. Karpov, R. Van Deursen, and G. Godin, "State-of-the-art augmented nlp transformer models for direct and single-step retrosynthesis," *Nature communications*, vol. 11, no. 1, p. 5575, 2020.

[53] X. Wang *et al.*, "Retroprime: A diverse, plausible and transformer-based method for single-step retrosynthesis predictions," *Chemical Engineering Journal*, vol. 420, p. 129 845, 2021.

[54] E. Kim, D. Lee, Y. Kwon, M. S. Park, and Y.-S. Choi, "Valid, plausible, and diverse retrosynthesis using tied two-way transformers with latent variables," *Journal of Chemical Information and Modeling*, vol. 61, no. 1, pp. 123–133, 2021.

[55] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[56] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st. USA: Prentice Hall PTR, 2000, ISBN: 0130950696.

[57] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[58] K.-D. Thellmann, B. Stadler, R. Usbeck, and J. Lehmann, "Transformer with tree-order encoding for neural program generation," 2022. arXiv: 2206.13354 [cs.CL].

[59] J. Harer, C. Reale, and P. Chin, "Tree-transformer: A transformer-based method for correction of tree-structured data," 2019. arXiv: 1908.00449 [cs.LG].

[60] D. M. Lowe, *Patent reaction extraction: Downloads*, 2014.

[61] D. M. Lowe, "Extraction of chemical structures and reactions from the literature," Ph.D. dissertation, University of Cambridge, 2012.

[62] N. Schneider, N. Stiefl, and G. A. Landrum, "What's what: The (nearly) definitive guide to reaction role assignment," *Journal of chemical information and modeling*, vol. 56, no. 12, pp. 2336–2346, 2016.

[63] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. arXiv: 1412.6980 [cs.LG].

[64] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[65] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[66] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen, "Prediction of organic reaction outcomes using machine learning," *ACS Central Science*, vol. 3, no. 5, pp. 434–443, 2017, PMID: 28573205. eprint: https://doi.org/10.1021/acscentsci.7b00064.

[67] P. Schwaller, T. Gaudin, D. Lanyi, C. Bekas, and T. Laino, *"found in translation": Predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models*, 2017. arXiv: 1711.04810 [cs.LG].

[68] D. Bajusz, A. Rácz, and K. Héberger, "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?" *Journal of cheminformatics*, vol. 7, no. 1, pp. 1–13, 2015.

[69] B. Liu *et al.*, "Retrosynthetic reaction prediction using neural sequence-to-sequence models," *ACS Central Science*, vol. 3, no. 10, pp. 1103–1113, 2017, PMID: 29104927. eprint: https://doi.org/10.1021/acscentsci.7b00303.

[70] S. Zheng, J. Rao, Z. Zhang, J. Xu, and Y. Yang, "Predicting retrosynthetic reactions using self-corrected transformer neural networks," *Journal of Chemical Information and Modeling*, vol. 60, no. 1, pp. 47–55, 2019, PMID: 31825611. eprint: https://doi.org/10.1021/acs.jcim.9b00949.

[71] K. Lin, Y. Xu, J. Pei, and L. Lai, "Automatic retrosynthetic route planning using template-free models," *Chemical Science*, vol. 11, no. 12, pp. 3355–3364, 2020.

[72] H. Duan, L. Wang, C. Zhang, L. Guo, and J. Li, "Retrosynthesis with attention-based nmt model and chemical analysis of "wrong" predictions," *RSC Advances*, vol. 10, no. 3, pp. 1371–1378, 2020.

[73] B. Chen, T. Shen, T. S. Jaakkola, and R. Barzilay, "Learning to make generalizable and diverse predictions for retrosynthesis," *arXiv preprint arXiv:1910.09688*, 2019.

[74] D. F. Rudd, G. J. Powers, and J. J. Siirola, "Process synthesis," *(No Title)*, 1973.

[75] J. Douglas, *Conceptual design of chemical processes*. McGraw-Hill, 1998.

[76] R. Banares-Alcantara, A. W. Westerberg, and M. D. Rychener, "Development of an expert system for physical property predictions," *Computers & chemical engineering*, vol. 9, no. 2, pp. 127–142, 1985.

[77] C. A. Jaksland, R. Gani, and K. M. Lien, "Separation process design and synthesis based on thermodynamic insights," *Chemical Engineering Science*, vol. 50, no. 3, pp. 511–530, 1995.

[78] L. Mencarelli, Q. Chen, A. Pagot, and I. E. Grossmann, "A review on superstructure optimization approaches in process system engineering," *Computers & Chemical Engineering*, vol. 136, p. 106 808, 2020.

[79] A. K. Tula, D. K. Babi, J. Bottlaender, M. R. Eden, and R. Gani, "A computer-aided software-tool for sustainable process synthesis-intensification," *Computers & Chemical Engineering*, vol. 105, pp. 74–95, 2017.

[80] L. d'Anterroches, *Process Flowsheet Generation & Design through a Group Contribution Approach*. [CAPEC], Department of Chemical Engineering, Technical University of Denmark, 2005.

[81] S. Bommareddy, M. R. Eden, and R. Gani, "Computer aided flowsheet design using group contribution methods," *Computer aided chemical engineering*, vol. 29, pp. 321–325, 2011.

[82] A. K. Tula, M. R. Eden, and R. Gani, "Process synthesis, design and analysis using a process-group contribution method," *Computers & Chemical Engineering*, vol. 81, pp. 245–259, 2015.

[83] G. Vogel, L. S. Balhorn, and A. M. Schweidtmann, "Learning from flowsheets: A generative transformer model for autocompletion of flowsheets," *Computers & Chemical Engineering*, vol. 171, p. 108 162, 2023.

[84] E. Hirtreiter, L. S. Balhorn, and A. M. Schweidtmann, "Towards automatic generation of piping and instrumentation diagrams (p&ids) with artificial intelligence," *arXiv preprint arXiv:2211.05583*, 2022.

[85] T. Zhang, N. V. Sahinidis, and J. J. Siirola, "Pattern recognition in chemical process flowsheets," *AIChE Journal*, vol. 65, no. 2, pp. 592–603, 2019.

[86] R. Gómez-Bombarelli *et al.*, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS central science*, vol. 4, no. 2, pp. 268–276, 2018.

[87] P. Pogány, N. Arad, S. Genway, and S. D. Pickett, "De novo molecule design by translating from reduced graphs to smiles," *Journal of chemical information and modeling*, vol. 59, no. 3, pp. 1136–1146, 2018.

[88] V. Mann, K. Brito, R. Gani, and V. Venkatasubramanian, "Hybrid, interpretable machine learning for thermodynamic property estimation using grammar2vec for molecular representation," *Fluid Phase Equilibria*, vol. 561, p. 113 531, 2022.

[89] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, "Smiles-bert: Large scale unsupervised pre-training for molecular property prediction," in *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, 2019, pp. 429–436.

[90] V. Mann and V. Venkatasubramanian, "Ai-driven hypergraph network of organic chemistry: Network statistics and applications in reaction classification," *Reaction Chemistry & Engineering*, 2023.

[91] W. D. Seider, D. R. Lewin, J. Seader, S. Widagdo, R. Gani, and K. M. Ng, *Product and process design principles: synthesis, analysis, and evaluation*. John Wiley & Sons, 2017.

[92] V. Mann, R. Gani, and V. Venkatasubramanian, "Intelligent process flowsheet synthesis and design using extended sfiles representation," in *Computer Aided Chemical Engineering*, vol. 52, Elsevier, 2023, pp. 221–226.

[93] C. Simon, K. Davidsen, C. Hansen, E. Seymour, M. B. Barnkob, and L. R. Olsen, "Bioreader: A text mining tool for performing classification of biomedical literature," *BMC bioinformatics*, vol. 19, pp. 165–170, 2019.

[94] U.S. Food and Drug Administration, *Understanding CDER's risk-based selection model*, `https://www.fda.gov/media/116004/download`, Accessed October 28, 2022., 2018.

[95] X. Y. Lawrence, A. Raw, L. Wu, C. Capacci-Daniel, Y. Zhang, and S. Rosencrance, "Fda's new pharmaceutical quality initiative: Knowledge-aided assessment & structured applications," *International journal of pharmaceutics: X*, vol. 1, p. 100 010, 2019.

[96] K. Beuls, P. Van Eecke, and V. S. Cangalovic, "A computational construction grammar approach to semantic frame extraction," *Linguistics Vanguard*, vol. 7, no. 1, p. 20 180 015, 2021.

[97] M. Schmitz, S. Soderland, R. Bart, O. Etzioni, *et al.*, "Open language learning for information extraction," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, 2012, pp. 523–534.

[98] D. Zhang, S. Mukherjee, C. Lockard, X. L. Dong, and A. McCallum, "Openki: Integrating open information extraction and knowledge bases with relation inference," *arXiv preprint arXiv:1904.12606*, 2019.

[99] P. Gamallo, M. Garcia, and S. Fernández-Lanza, "Dependency-based open information extraction," in *Proceedings of the joint workshop on unsupervised and semi-supervised learning in NLP*, 2012, pp. 10–18.

[100] J. Christensen, Mausam, S. Soderland, and O. Etzioni, "An analysis of open information extraction based on semantic role labeling," in *Proceedings of the sixth international conference on Knowledge capture*, 2011, pp. 113–120.

[101] Y. Versley *et al.*, "Bart: A modular toolkit for coreference resolution," in *Proceedings of the ACL-08: HLT Demo Session*, 2008, pp. 9–12.

[102] S. Viswanath, J. W. Fennell, K. Balar, and P. Krishna, "An industrial approach to using artificial intelligence and natural language processing for accelerated document preparation in drug development," *Journal of Pharmaceutical Innovation*, vol. 16, pp. 302–316, 2021.

[103] R. Kulkarni, H. Kulkarni, K. Balar, and P. Krishna, "Cognitive natural language search using calibrated quantum mesh," in *2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, IEEE, 2018, pp. 174–178.

[104] T. Kang *et al.*, "Eliie: An open-source information extraction system for clinical trial eligibility criteria," *Journal of the American Medical Informatics Association*, vol. 24, no. 6, pp. 1062–1071, 2017.

[105] C. Yuan *et al.*, "Criteria2query: A natural language interface to clinical databases for cohort definition," *Journal of the American Medical Informatics Association*, vol. 26, no. 4, pp. 294–305, 2019.

[106] H. Xu, S. P. Stenner, S. Doan, K. B. Johnson, L. R. Waitman, and J. C. Denny, "Medex: A medication information extraction system for clinical narratives," *Journal of the American Medical Informatics Association*, vol. 17, no. 1, pp. 19–24, 2010.

[107] S. Harmata, K. Hofer-Schmitz, P.-H. Nguyen, C. Quix, and B. Bakiu, "Layout-aware semi-automatic information extraction for pharmaceutical documents," in *Data Inte-*

gration in the Life Sciences: 12th International Conference, DILS 2017, Luxembourg, Luxembourg, November 14-15, 2017, Proceedings 12, Springer, 2017, pp. 71–85.

[108] A. L. Gentile, D. Gruhl, P. Ristoski, and S. Welch, "Personalized knowledge graphs for the pharmaceutical domain," in *The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18*, Springer, 2019, pp. 400–417.

[109] R. Saidi, M. Maddouri, and E. M. Nguifo, "Comparing graph-based representations of protein for mining purposes," in *Proceedings of the KDD-09 Workshop on Statistical and Relational Learning in Bioinformatics*, 2009, pp. 35–38.

[110] C. M. Gothard *et al.*, "Rewiring chemistry: Algorithmic discovery and experimental validation of one-pot reactions in the network of organic chemistry," *Angewandte Chemie*, vol. 124, no. 32, pp. 8046–8051, 2012.

[111] T. Washio and H. Motoda, "State of the art of graph-based data mining," *Acm Sigkdd Explorations Newsletter*, vol. 5, no. 1, pp. 59–68, 2003.

[112] M. Skeppstedt, M. Kvist, G. H. Nilsson, and H. Dalianis, "Automatic recognition of disorders, findings, pharmaceuticals and body structures from clinical text: An annotation and machine learning study," *Journal of biomedical informatics*, vol. 49, pp. 148–158, 2014.

[113] U. Leser and J. Hakenberg, "What makes a gene name? named entity recognition in the biomedical literature," *Briefings in bioinformatics*, vol. 6, no. 4, pp. 357–369, 2005.

[114] R. Gaizauskas, G. Demetriou, P. J. Artymiuk, and P. Willett, "Protein structures and information extraction from biological texts: The pasta system," *Bioinformatics*, vol. 19, no. 1, pp. 135–143, 2003.

[115] N. Collier, C. Nobata, and J. Tsujii, "Extracting the names of genes and gene products with a hidden markov model," in *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, 2000.

[116] D. Shen, J. Zhang, G. Zhou, J. Su, and C. L. Tan, "Effective adaptation of hidden markov model-based named entity recognizer for biomedical domain," in *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, 2003, pp. 49–56.

[117] Y. Sasaki, Y. Tsuruoka, J. McNaught, and S. Ananiadou, "How to make the most of ne dictionaries in statistical ner," *BMC bioinformatics*, vol. 9, no. 11, pp. 1–9, 2008.

[118] R. Bhatnagar, S. Sardar, M. Beheshti, and J. T. Podichetty, "How can natural language processing help model informed drug development?: A review," *JAMIA open*, vol. 5, no. 2, ooac043, 2022.

[119] M.-S. Huang, P.-T. Lai, P.-Y. Lin, Y.-T. You, R. T.-H. Tsai, and W.-L. Hsu, "Biomedical named entity recognition and linking datasets: Survey and our recent development," *Briefings in Bioinformatics*, vol. 21, no. 6, pp. 2219–2238, 2020.

[120] M. T. Pilehvar, A. Bernard, D. Smedley, and N. Collier, "Phenebank: A literature-based database of phenotypes," *Bioinformatics*, vol. 38, no. 4, pp. 1179–1180, 2022.

[121] L. Luo, P.-T. Lai, C.-H. Wei, C. N. Arighi, and Z. Lu, "Biored: A rich biomedical relation extraction dataset," *Briefings in Bioinformatics*, vol. 23, no. 5, bbac282, 2022.

[122] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[123] P. Lee, S. Bubeck, and J. Petro, "Benefits, limits, and risks of gpt-4 as an ai chatbot for medicine," *New England Journal of Medicine*, vol. 388, no. 13, pp. 1233–1239, 2023.

[124] O. Bodenreider, "The unified medical language system (umls): Integrating biomedical terminology," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D267–D270, 2004.

[125] J. A. Fries *et al.*, "Ontology-driven weak supervision for clinical entity classification in electronic health records," *Nature communications*, vol. 12, no. 1, pp. 1–11, 2021.

[126] J. Lee *et al.*, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[127] G. Angeli, M. J. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 344–354.

[128] S. Viswanath, S. Guntz, J. Dieringer, S. Vaidyaraman, H. Wang, and C. Gounaris, "An ontology to describe small molecule pharmaceutical product development and methodology for optimal activity scheduling," *Journal of Pharmaceutical Innovation*, pp. 1–15, 2021.

[129] L. Hailemariam and V. Venkatasubramanian, "Purdue ontology for pharmaceutical engineering: Part i. conceptual framework," *Journal of Pharmaceutical Innovation*, vol. 5, pp. 88–99, 2010.

[130] L. Hailemariam and V. Venkatasubramanian, "Purdue ontology for pharmaceutical engineering: Part ii. applications," *Journal of Pharmaceutical Innovation*, vol. 5, pp. 139–146, 2010.

[131] M. F. M. Remolona *et al.*, "Hybrid ontology-learning materials engineering system for pharmaceutical products: Multi-label entity recognition and concept detection," *Computers & Chemical Engineering*, vol. 107, pp. 49–60, 2017.

[132] M. A. Musen, "The protégé project: A look back and a look forward," *AI matters*, vol. 1, no. 4, pp. 4–12, 2015.

[133] K. Degtyarenko *et al.*, "Chebi: A database and ontology for chemical entities of biological interest," *Nucleic acids research*, vol. 36, no. suppl_1, pp. D344–D350, 2007.

[134] A. S. Schwartz and M. A. Hearst, "A simple algorithm for identifying abbreviation definitions in biomedical text," in *Biocomputing 2003*, World Scientific, 2002, pp. 451–462.

[135] M. Honnibal and I. Montani, "Spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, vol. 7, no. 1, pp. 411–420, 2017.

[136] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[137] A. Akkasi, E. Varoğlu, and N. Dimililer, "Chemtok: A new rule based tokenizer for chemical named entity recognition," *BioMed research international*, vol. 2016, 2016.

[138] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[139] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, NIH Public Access, vol. 11, 2017, p. 269.

[140] *Efficacy guidelines - international council for harmonisation*, `https://www.ich.org/page/efficacy-guidelines`, Accessed: Month Day, Year.

[141] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[142] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[143] A. Muthukkumaran, S. Raghunathan, A. Ravichandran, and R. Rengaswamy, "Perovskite-based electrocatalyst discovery and design using word embeddings from retrained scibert language model," *AIChE Journal*, e18068,

[144] C. Trinh, D. Meimaroglou, and S. Hoppe, "Machine learning in chemical product engineering: The state of the art and a guide for newcomers," *Processes*, vol. 9, no. 8, p. 1456, 2021.

[145] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, Citeseer, vol. 242, 2003, pp. 29–48.

[146] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[147] A.-L. Barabási and E. Bonabeau, "Scale-free networks," *Scientific american*, vol. 288, no. 5, pp. 60–69, 2003.

[148] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999.

[149] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: Guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[150] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[151] J. Payne, "Deep hyperedges: A framework for transductive and inductive learning on hypergraphs," *arXiv preprint arXiv:1910.02633*, 2019.

[152] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[153] D. Maurya and B. Ravindran, "Hyperedge prediction using tensor eigenvalue decomposition," *Journal of the Indian Institute of Science*, vol. 101, no. 3, pp. 443–453, 2021.

[154] M. Fialkowski, K. J. Bishop, V. A. Chubukov, C. J. Campbell, and B. A. Grzybowski, "Architecture and evolution of organic chemistry," *Angewandte Chemie International Edition*, vol. 44, no. 44, pp. 7263–7269, 2005.

[155] K. J. Bishop, R. Klajn, and B. A. Grzybowski, "The core and most useful molecules in organic chemistry," *Angewandte Chemie International Edition*, vol. 45, no. 32, pp. 5348–5354, 2006.

[156] P.-M. Jacob and A. Lapkin, "Statistics of the network of organic chemistry," *Reaction Chemistry & Engineering*, vol. 3, no. 1, pp. 102–118, 2018.

[157] V. Mann, A. Sivaram, L. Das, and V. Venkatasubramanian, "Robust and efficient swarm communication topologies for hostile environments," *Swarm and Evolutionary Computation*, vol. 62, p. 100 848, 2021.

[158] B. A. Grzybowski, K. J. Bishop, B. Kowalczyk, and C. E. Wilmer, "The'wired'universe of organic chemistry," *Nature Chemistry*, vol. 1, no. 1, pp. 31–36, 2009.

[159] P. Chodrow and A. Mellor, "Annotated hypergraphs: Models and applications," *Applied network science*, vol. 5, no. 1, pp. 1–25, 2020.

[160] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, 2–es, 2007.

[161] D. Probst, P. Schwaller, and J.-L. Reymond, "Reaction classification and yield prediction using the differential reaction fingerprint drfp," *Digital discovery*, vol. 1, no. 2, pp. 91–97, 2022.

[162] J. L. Baylon, N. A. Cilfone, J. R. Gulcher, and T. W. Chittenden, "Enhancing retrosynthetic reaction prediction with deep learning using multiscale reaction classification," *Journal of chemical information and modeling*, vol. 59, no. 2, pp. 673–688, 2019.

[163] N. Schneider, D. M. Lowe, R. A. Sayle, and G. A. Landrum, "Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity," *Journal of chemical information and modeling*, vol. 55, no. 1, pp. 39–53, 2015.

Appendix A: SMILES grammar

The SMILES grammar used in this work is the same as that used in our previous works [28, 29]. This grammar comprises 80 production rules with 24 non-terminals symbols specifying the different structural components of a SMILES string. All the production rules for the grammar used in our work are summarized in Table A.1. The first and the last production rules, SMILES ⟶ CHAIN and NOTHING ⟶ NONE, are additional rules included signifying the start and end of a SMILES string, which is analogous to the <START> and <END> tokens in natural language processing marking the beginning and the end of sentences, respectively.

Table A.1: SMILES grammar used for retrosynthesis and forward reaction prediction. '|' separates multiple production rules applicable for the same non-terminal symbol.

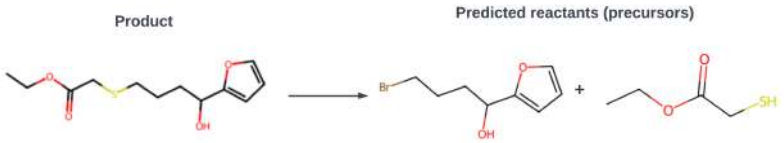
S.No	Production rules
1	SMILES ⟶ CHAIN
2	ATOM ⟶ BRACKET_ATOM | ALIPHATIC_ORGANIC | AROMATIC_ORGANIC
3	ALIPHATIC_ORGANIC ⟶ B | C | N | O | S | P | F | I | Cl | Br
4	AROMATIC_ORGANIC ⟶ c | n | o | s | p
5	BRACKET_ATOM ⟶ LEFT_BRACKET BAI RIGHT_BRACKET
6	BAI ⟶ SYMBOL BAC
7	BAC ⟶ CHIRAL BAH | BAH | CHIRAL
8	BAH ⟶ HCOUNT BACH | BACH | HCOUNT
9	BACH ⟶ CHARGE
10	SYMBOL ⟶ ALIPHATIC_ORGANIC | AROMATIC_ORGANIC
11	DIGIT ⟶ 1 | 2 | 3 | 4 | 5 | 6 | 7
12	CHIRAL ⟶ @ | @@
13	HCOUNT ⟶ H | H DIGIT
14	CHARGE ⟶ - | +
15	BOND ⟶ - | = | # | / | \

Table A.1: SMILES grammar used for retrosynthesis and forward reaction prediction. '|' separates multiple production rules applicable for the same non-terminal symbol.

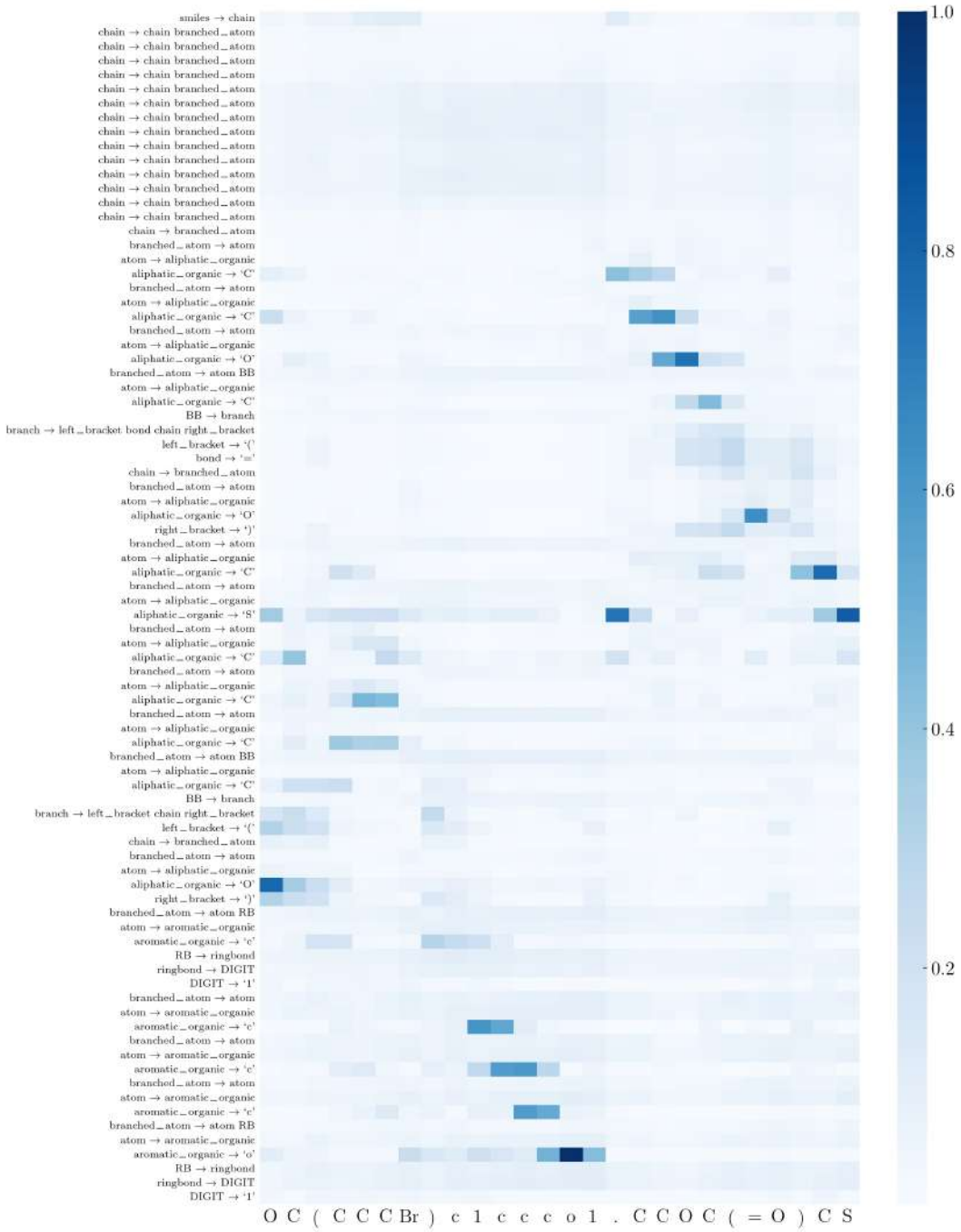
S.No	Production rules
16	RINGBOND ⟶ DIGIT | BOND DIGIT
17	BRANCHED_ATOM ⟶ ATOM | ATOM RB | ATOM BB | ATOM RB BB
18	RB ⟶ RB RINGBOND | RINGBOND
19	BB ⟶ BB BRANCH | BRANCH
20	BRANCH ⟶ LEFT_BRACKET CHAIN RIGHT_BRACKET | LEFT_BRACKET BOND CHAIN RIGHT_BRACKET
21	CHAIN ⟶ BRANCHED_ATOM | CHAIN BRANCHED_ATOM | CHAIN BOND BRANCHED_ATOM
22	LEFT_BRACKET ⟶ BRANCHED_ATOM | CHAIN BRANCHED_ATOM | CHAIN BOND BRANCHED_ATOM
23	RIGHT_BRACKET ⟶ BRANCHED_ATOM | CHAIN BRANCHED_ATOM | CHAIN BOND BRANCHED_ATOM
24	NOTHING ⟶ NONE

Appendix B: Additional cross-attention maps from G-MATT

The attention maps were computed for a more complex reaction `COc1ccc(-c2ccc(C)cc2)cc1` → `COc1ccc(Br)cc1.Cc1ccc(B(O)O)cc1` in Figure B.1 and `CCOC(=O)CSCCCC(O)c1ccco1` → `OC(CCCBr)c1ccco1.CCOC(=O)CS` in Figure B.2.

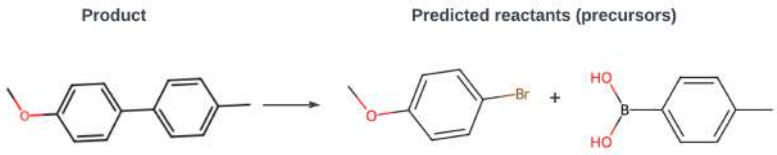


(a) An example top-1 prediction to study the transformer attention map



(b) Average attention scores for top-1 prediction on the retrosynthesis reaction `CCOC(=O)CSCCCC(O)c1ccco1` → `OC(CCCBr)c1ccco1.CCOC(=O)CS` extracted from the transformer decoder cross-attention sublayer

Figure B.1: Molecular attention map for the SMILES grammar tree of the product and SMILES strings of the reactants (precursors) for an example reaction



(a) An example top-1 prediction to study the transformer attention map

(b) Average attention scores for top-1 prediction on the retrosynthesis reaction `COc1ccc(-c2ccc(C)cc2)cc1` → `COc1ccc(Br)cc1.Cc1ccc(B(O)O)cc1` extracted from the transformer decoder cross-attention sublayer

Figure B.2: Molecular attention map for the SMILES grammar tree of the product and SMILES strings of the reactants (precursors) for an example reaction

Appendix C: Flowsheet grammar syntax rules

A complete list of formal syntax rules developed for the SFILES grammar are shown in Table C.1. Each syntax rule comprises non-terminal and terminal symbols. The terminal symbols are written in lowercase with quotes and correspond to the SFILES vocabulary (or list of symbols). The non-terminal symbols on the other hand are written in block letters and correspond to the underlying meta information corresponding to the various process equipment, streams, materials, and so on, organized hierarchically based on process knowledge. The syntax rules define how the symbol on the left hand side is manipulated to give rise to the symbols on the right hand side, which could be a combination of non-terminal and terminal symbols. In other words, all syntactically valid SFILES string could be obtained by sequentially applying the syntax rules presented in Table C.1, always starting with the first rule `SFILES` $\longrightarrow$ `PG`, and applying the rest wherever applicable. Thus, a hierarchical tree could be constructed for each SFILES string using these syntax rules similar to that shown in Figure 4 in the manuscript.

Table C.1: Developed grammar syntax rules corresponding to the text-based SFILES representation for flowsheets.

Rule	**Grammar rules**	**Meaning/explanation**
R_1	`SFILES` $\longrightarrow$ `PG`	top node resulting into a PG node
R_2	`PG` $\longrightarrow$ `PA PG`	process groups (PG) contains one or more process-atoms (PA)
R_3	`PG` $\longrightarrow$ `PA`	PG with exactly one PA
R_4	`PG` $\longrightarrow$ `BRANCHED_PG`	PG enclosed in square brackets; used when a given process group has two or more process bonds as output

Table C.1: Developed grammar syntax rules corresponding to the text-based SFILES representation for flowsheets.

Rule	Grammar rules	Meaning/explanation
R_5	BRANCHED_PG ⟶ SQBRAC1 PA SQBRAC2	PA enclosed in square brackets; used when a given process group has two or more process bonds as output
R_6	BRANCHED_PG ⟶ SQBRAC1 PA RECYCLE_OUT SQBRAC2	PA followed by recycle out stream enclosed in square brackets; used when a given process group has two or more process bonds as output along with a recycle stream
R_7	BRANCHED_PG ⟶ PA	Branched PG with exactly one PA
R_8	PA ⟶ BRAC1 PA BRAC2	PA enclosed in parentheses
R_9	PA ⟶ EQUIPMENT	PA that is a process equipment
R_{10}	PA ⟶ INLET	PA that is an inlet stream
R_{11}	PA ⟶ OUTLET	PA that is an outlet stream
R_{12}	INLET ⟶ BRAC1 INLETmark STREAM BRAC2	inlet as inlet-mark followed by stream in parentheses
R_{13}	INLET ⟶ INLETmark STREAM	flowsheet inlet stream represented as inlet-mark followed by stream process-atom enclosed in parentheses
R_{14}	OUTLET ⟶ BRAC1 OUTLETmark STREAM BRAC2	flowsheet outlet stream as outlet-mark followed by stream process-atom in parentheses
R_{15}	OUTLET ⟶ OUTLETmark STREAM	outlet as outlet-mark followed by stream
R_{16}	STREAM ⟶ MAT	stream with a single material
R_{17}	STREAM ⟶ MAT STREAM	a recursive rule that allows a process stream to contain more than one material
R_{18}	EQUIPMENT ⟶ REACTOR	equipment that is a reactor

Table C.1: Developed grammar syntax rules corresponding to the text-based SFILES representation for flowsheets.

Rule	Grammar rules	Meaning/explanation
R_{19}	EQUIPMENT ⟶ SEP	equipment that is a separator
R_{20}	EQUIPMENT ⟶ DIV	equipment that is a divider
R_{21}	SEP ⟶ DISTIL	separator that is a distillation column
R_{22}	SEP ⟶ MEMBRANE	separator that is a membrane separator
R_{23}	SEP ⟶ MS	separator that is a sieve separator
R_{24}	SEP ⟶ CRS	separator that is a crystallization separator
R_{25}	SEP ⟶ FSH	separator that is a flash
R_{26}	MEMBRANE ⟶ LMEM	liquid membrane separator
R_{27}	MEMBRANE ⟶ GMEM	gas membrane separator
R_{28}	GMEM ⟶ MEMBRANEmark_gas TOP BCKSLSH BOTTOM	gas mem/. sep. as gmem-mark, top prod., backslash, bottom prod.
R_{29}	LMEM ⟶ MEMBRANEmark_liq TOP BCKSLSH BOTTOM	liq. mem. sep. as lmem-mark, top prod., backslash, bottom prod.
R_{30}	FSH ⟶ FSHmark TOP BCKSLSH BOTTOM	flash as flash-mark, top product, backslash, bottom product
R_{31}	MS ⟶ SIEVE_SEPmark TOP BCKSLSH BOTTOM	sieve sep. as ms-mark, top prod., backslash, bottom prod.
R_{32}	CRS ⟶ CRYSTALmark TOP BCKSLSH BOTTOM	crystal sep. as crystal-mark, top prod., backslash, bottom prod.
R_{33}	DISTIL ⟶ DISTILmark TOP BCKSLSH BOTTOM	distillation column as top product, backslash, bottom-product
R_{34}	REACTOR ⟶ REACTANTmark REACTANT BCKSLSH PRODUCT	reactor as reactant-mark, reactant, backslash, product
R_{35}	DIV ⟶ PRG	divider that is purge
R_{36}	PRG ⟶ PRGmark TOP BCKSLSH BOTTOM	purge as prg-mark, top product, backslash, bottom product
R_{37}	PA ⟶ RECYCLE_OUT	process-atom that is recycle-outlet

Table C.1: Developed grammar syntax rules corresponding to the text-based SFILES representation for flowsheets.

Rule	Grammar rules	Meaning/explanation
R_{38}	RECYCLE ⟶ RECYCLE_IN PA	recycle as recycle inlet followed by another PA
R_{39}	RECYCLE ⟶ RECmark INLET	recycle as rec-mark followed by inlet stream
R_{40}	RECYCLE ⟶ RECmark OUTLET	recycle as rec-mark followed by outlet stream
R_{41}	RECYCLE_IN ⟶ RECYCLE_IN RECYCLE_IN	two recycle in adjacent to each other
R_{42}	RECYCLE_IN ⟶ RECmark REC_STREAM_ID	recycle in as rec-mark and recycle stream id
R_{43}	RECYCLE_IN ⟶ RECmark STREAM	recycle in as rec-mark and stream
R_{44}	RECYCLE_OUT ⟶ REC_STREAM_ID	recycle out as recycle stream id
R_{45}	REACTANT ⟶ STREAM	reactant stream
R_{46}	PRODUCT ⟶ STREAM	product stream
R_{47}	TOP ⟶ STREAM	top product stream
R_{48}	BOTTOM ⟶ STREAM	bottom product stream
R_{49-52}	RECYCLE_STREAM_ID ⟶ '1' | '2' | '3' | '4'	recycle stream that could be one amont 1, 2, 3, or 4
R_{53-57}	MAT ⟶ 'A' | 'B' | 'C' | 'D' | 'E'	material that could be one among A, B, C, D, or E
R_{58}	BRAC1 ⟶ '('	opening parentheses
R_{59}	BRAC2 ⟶ ')'	closing parentheses
R_{60}	SQBRAC1 ⟶ '['	opening square bracket; used when a given process atom has two or more process bonds as output
R_{61}	SQBRAC2 ⟶ ']'	closing square bracket; used when a given process atom has two or more process bonds as output

Table C.1: Developed grammar syntax rules corresponding to the text-based SFILES representation for flowsheets.

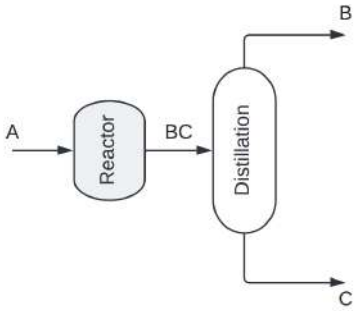
Rule	Grammar rules	Meaning/explanation
R_{62}	BCKSLSH ⟶ '/'	backslash symbol that separates reactants/products for reactor, or top/bottom product for separators, dividers
R_{63}	REACTANTmark ⟶ 'r'	reactant-mark for reactor
R_{64}	MEMBRANEmark_gas ⟶ 'gmem'	gas-membrane mark for gas membrane separator
R_{65}	MEMBRANEmark_liq ⟶ 'lmem'	liq-membrane mark for liquid membrane separator
R_{66}	SIEVE_SEP_mark ⟶ 'ms'	sieve-separator mark
R_{67}	CRYSTALmark ⟶ 'crs'	crystal-separator mark
R_{68}	FSHmark ⟶ 'f'	flash-mark for flash separator
R_{69}	RECmark ⟶ '<'	recycle-mark for recycle process bonds
R_{70}	PRGmark ⟶ 'prg'	purge-mark for purge divider
R_{71}	DISTILmark ⟶ 'd'	distillation column-mark for representing distillation columns
R_{72}	INLETmark ⟶ 'i'	inlet-mark for indicating flowsheet inlet process atom
R_{73}	OUTLETmark ⟶ 'o'	outlet-mark for indicating flowsheet outlet process atom
R_{74}	EQUIPMENT ⟶ ABSORBER	an equipment that is an absorber
R_{75}	ABSORBER ⟶ ABSmark STREAM BCKSLSH SOLVENT BCKSLSH STREAM	absorber with top-outlet stream, solvent input stream, and bottom-outlet stream; the bottom feed stream could be inferred with simple material balance
R_{76}	SOLVENT ⟶ SOLVENTEmark STREAM	solvent stream for the absorber

Table C.1: Developed grammar syntax rules corresponding to the text-based SFILES representation for flowsheets.

Rule	Grammar rules	Meaning/explanation
R_{77}	`ABSmark` $\longrightarrow$ `'abs'`	mark for the absorber
R_{78}	`SOLUTEmark` $\longrightarrow$ `'sol'`	mark for the solvent stream

'|' separates multiple syntax rules available for the same non-terminal symbol. Also, PG here refers to process-groups a combination of one or more process atoms (PAs) and has been introduced to enforce a hierarchy in the grammar rules.

Appendix D: Additional SFILES grammar trees



(a) Illustration of a process flow diagram with reactor followed by distillation

(iA)(rA/BC)(dB/C)(oB)(oC)

(b) Text-based SFILES representation for the process shown above

Figure D.1: Process flow diagram and text-based SFILES representation for a system with reactor followed by a distillation column with grammar tree shown in Figure D.2

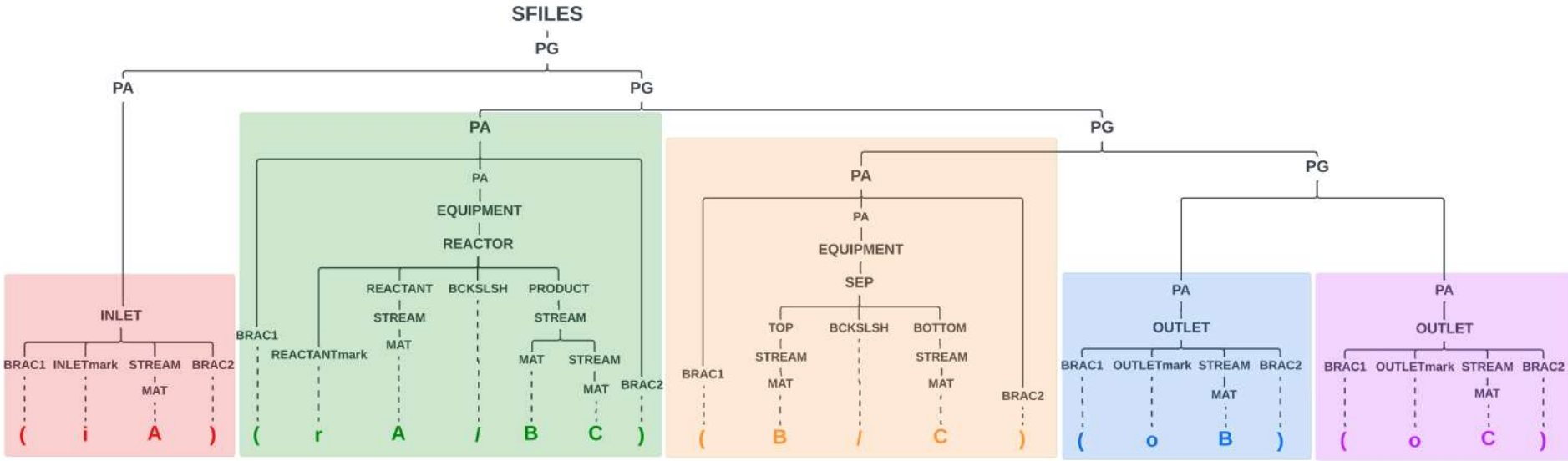


Figure D.2: SFILES grammar parse tree obtained for the SFILES string shown in Figure D.1b

Appendix E: Additional information extraction examples from SUSIE

Input 1: Compound_1, IUPAC_1, and Compound_8, IUPAC_2, are proposed starting materials for the commercial manufacture of Drug_1, as shown in Scheme 3.0-1. The drug substance, Drug_1, is manufactured as a monohydrate and is also referred to using the reference numbers Compound_9 and Compound_10 monohydrate. The non-proprietary name Drug_1 can refer to the anhydrous or monohydrate form interchangeably. Reference to the water of hydration is retained in the chemical information (chemical names, formulas, weight). In the CMC sections of this document, Drug_1 refers to the monohydrate form. When referring specifically to the anhydrous form, the text will state either Compound_10 or Drug_1 (anhydrous).

Input 2: The two proposed starting materials provide all the non-hydrogen atoms that form Drug_1, except the amide oxygen atom and the water of hydration, and therefore, are both considered significant structural fragments of the drug substance. These starting materials are readily synthesized, have been well-characterized, and demonstrate good stability.

Input 3: Supporting data are provided herein to indicate that the two proposed starting materials, compound_1 and Compound_8, are ideal candidates for the control of impurities entering the synthetic process. The proposed starting materials enter the process with adequate purity, guided by a rigorous set of proposed specifications, that ensure the drug substance does not contain significant (greater than 0.10%) impurities originating from the starting materials. Additionally, the downstream processes required to convert these proposed starting materials to drug substance also provide significant impurity reduction and control capability as demonstrated through batch history data and impurity spiking studies. This combination of factors is the basis the proposal that the two proposed starting materials are appropriate for the commercial manufacture of Drug_1.

Input 4: The drug substance manufacturing process uses two starting materials, compound_1 and Compound_8. It consists of three synthetic steps and employs 3 covalent bond making/breaking processes. In Step 1a, a mixture of the two starting materials (compound_1 and Compound_8), IUPAC_2 and toluene are heated at reflux under a partial vacuum, with water removal, to form Compound_11 by a condensation reaction. In Step 1b, IUPAC_3 and additional IUPAC_2 are added, nitrogen is used to reduce the oxygen level, and most of the toluene is removed by vacuum distillation. Nitrogen is again used to reduce the oxygen level in the reaction mixture, before it is heated to produce Compound_6 by a cyclocondensation reaction. As the reaction is cooled, seed crystals may be added to aid crystallization of Compound_6. To complete the crystallization, water is added, followed by an aqueous solution of potassium carbonate. The product is then isolated by filtration, washed, and dried.

Input 5: In Step 2, aqueous sodium hydroxide is added to a heated solution of Compound_6 in IUPAC_4 to hydrolyze the nitrile to an amide. When the reaction is complete, the partially-cooled solution may be seeded to initiate crystallization. The crystallization is completed by the addition of water to the warm slurry followed by cooling. The product is then isolated by filtration, washed and dried.

In Step 3, technical-grade Compound_10 monohydrate, ethanol, which may be denatured with methanol, and water are heated and the resulting solution is polish filtered. Following concentration, the solution may be seeded with Drug_1 to initiate crystallization. The temperature is reduced slightly and water is added to the warm mixture. The slurry is then cooled to complete the crystallization. The particle size distribution is adjusted by slurry-milling followed by thermal-cycling. The product is isolated by filtration, washed and dried.