AI-BIO INNOVATE CHALLENGE

Team Name: Vageesh.1

Team Members:

Vageesh Jangra

Sweety Khut

Shreyansh Lodha

Problem Statement: RS2

Build a seq to seq based model which could predict the reaction conditions for the reaction i.e. given reaction smiles as an input it should be able to predict the conditions.

ABSTRACT

The prediction of chemical reactions has become a focal point in the machine learning community due to its intricate nature and critical implications in the field of chemistry. However, the evaluation of models for this task has predominantly relied on simplistic metrics like top-k accuracy, which may oversimplify a model's shortcomings. Drawing inspiration from advancements in other domains, we propose a novel assessment framework that extends current methodologies, aiming for a more comprehensive evaluation. Our approach incorporates CHORISO, a meticulously curated dataset with specialized splits to simulate chemically relevant scenarios. Additionally, we introduce a set of metrics designed to offer a holistic understanding of a model's strengths and weaknesses. When applied to state-of-the-art models, our method uncovers noteworthy distinctions in sensitive aspects, particularly in stereoselectivity and generalization to out-of-distribution chemical scenarios. This work signifies a step towards the development of robust prediction models capable of expediting chemical discovery processes.

INDEX

INTRODUCTION

Retrosynthetic analysis, a process that involves working backward from a target molecule to simpler precursor compounds, is a fundamental aspect of organic chemistry. While existing approaches have primarily focused on predicting the sequence of reactions needed to synthesize a compound, our project addresses a critical yet understudied dimension—anticipating the conditions under which a reaction occurs. The conditions, encompassing factors like choice of catalyst, solvent, and temperature, play a pivotal role in the success and efficiency of a chemical synthesis.

Traditional methods rely heavily on the expertise of organic chemists to manually select optimal conditions. However, the automation of this decision-making process through machine learning holds the promise of significantly expediting the synthesis planning workflow. In our pursuit, we aim to construct a seq2seq-based model that learns the intricate relationship between reaction SMILES and the corresponding reaction conditions. This innovative approach is expected to contribute to the advancement of automated retrosynthesis planning, providing chemists with a predictive tool that enhances efficiency and accelerates the pace of chemical discovery.

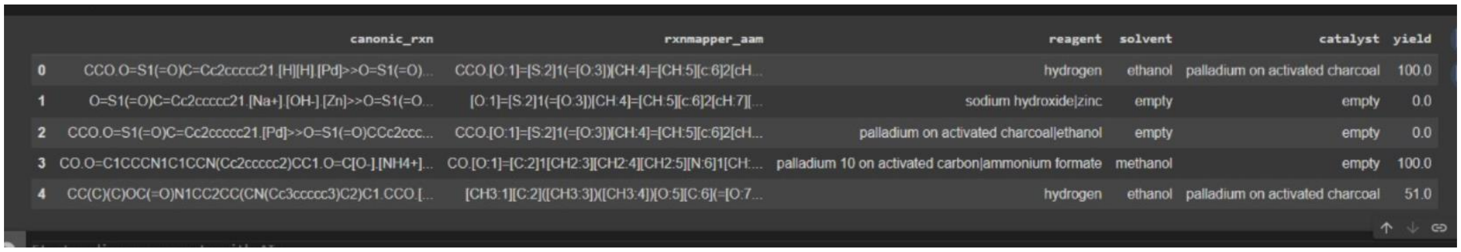
OBJECTIVE

The endeavor to automate and enhance the process of retrosynthesis planning in organic chemistry has gained significant momentum with the application of machine learning techniques. In this project, our focus is on a specific facet of this intricate problem—the prediction of reaction conditions based on reaction SMILES (Simplified Molecular Input Line Entry System) representations. The synthesis of organic compounds involves a myriad of complex reactions, each influenced by a unique set of conditions such as catalysts, solvents, and reagents. The ability to predict these reaction conditions from the given reaction SMILES serves as a crucial step towards automating and optimizing the chemical synthesis process. By leveraging sequence-to-sequence (seq2seq) models, a powerful category of neural networks, we aim to develop a predictive model that can ingest reaction SMILES as input and accurately forecast the associated reaction conditions. This task holds immense potential to revolutionize the field of organic synthesis, offering researchers a valuable tool for streamlining experimentation and accelerating the discovery of novel chemical compounds.

Dataset and Preprocessing

The CHORISO dataset is the basis for our project aiming to predict the conditions under which chemical reactions happen. Comprising six columns, this dataset encapsulates diverse chemical reactions along with their corresponding reactants, reagents, solvents, catalysts, and reaction yields.

1. **canonic_rxn:** This column contains the canonical representation of chemical reactions using SMILES (Simplified Molecular Input Line Entry System) strings. Each row represents a distinct chemical transformation, capturing the reactants and products involved.
2. **rxnmapper_aam**: This column encodes atom-to-atom mappings (AAM) for the reaction. AAM is a technique used to establish correspondence between atoms in reactants and products, facilitating the tracking of atom transformations during a reaction.
3. **Reagent**: This column details the reagents utilized in the chemical reactions. Reagents are chemical substances introduced into a reaction to initiate or facilitate the transformation.
4. **Solvent**: For each reaction, the solvents used are listed in the solvent column. Reaction conditions are greatly influenced by solvents, which also have an impact on reaction rates and results.
5. **Catalyst**: Substances known as catalysts quicken reactions without changing permanently on their own. Understanding catalysts helps one gain understanding of the efficiency and mechanism of reactions.
6. **Yield**: The yield column measures the effectiveness of the chemical reactions. Reaction yields are an essential metric for assessing the efficacy of the reaction conditions as well as a measure of the success of the reaction.

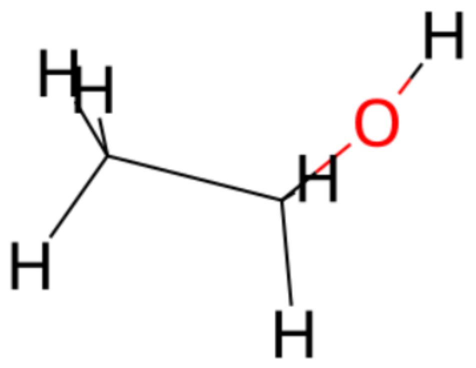


	canonic_rxn	rxnmapper_aam	reagent	solvent	catalyst	yield
0	CCO.O=S1(=O)C=Cc2ccccc21.[H][H].[Pd]>>O=S1(=O)...	CCO.[O:1]=[S:2]1(=[O:3])[CH:4]=[CH:5][c:6]2[cH...	hydrogen	ethanol	palladium on activated charcoal	100.0
1	O=S1(=O)C=Cc2ccccc21.[Na+].[OH-].[Zn]>>O=S1(=O...	[O:1]=[S:2]1(=[O:3])[CH:4]=[CH:5][c:6]2[cH:7][...	sodium hydroxide|zinc	empty	empty	0.0
2	CCO.O=S1(=O)C=Cc2ccccc21.[Pd]>>O=S1(=O)CCc2ccc...	CCO.[O:1]=[S:2]1(=[O:3])[CH:4]=[CH:5][c:6]2[cH...	palladium on activated charcoal|ethanol	empty	empty	0.0
3	CO.O=C1CCCN1C1CCN(Cc2ccccc2)CC1.O=C[O-].[NH4+]...	CO.[O:1]=[C:2]1[CH2:3][CH2:4][CH2:5][N:6]1[CH:...	palladium 10 on activated carbon|ammonium formate	methanol	empty	100.0
4	CC(C)(C)OC(=O)N1CC2CC(CN(Cc3ccccc3)C2)C1.CCO.[...	[CH3:1][C:2]([CH3:3])([CH3:4])[O:5][C:6](=[O:7...	hydrogen	ethanol	palladium on activated charcoal	51.0

Conversion of SMILES into molecular Graphs

We are using Graph Neural Network for preparing and formatting the data into a representation which would be suitable for seq-2-seq model. Converting SMILES (Simplified Molecular Input Line Entry System) representations into graph tensors is a crucial step when preparing data for graph neural networks (GNNs). Graph tensors are data structures that represent the molecular structure of compounds as graphs, where atoms are nodes and chemical bonds are edges. This conversion allows GNNs to operate on molecular graphs, capturing the relational information and spatial arrangement of atoms within a molecule.

The SMILE strings are passed as input to function smile_to_mol and use the RDKit library to convert it into a molecular object (mol). RDKit is a cheminformatics toolkit that provides functionality for working with chemical informatics data. The mol object represents the chemical structure encoded by the SMILES string. The add_hydrogens function is then applied to the molecular object, adding explicit hydrogen atoms to the structure. Finally, the draw_molecule function allows for the visualization of the molecular structure, either by displaying it in memory or saving it to a file.



Molecular Image for string 'CCO'

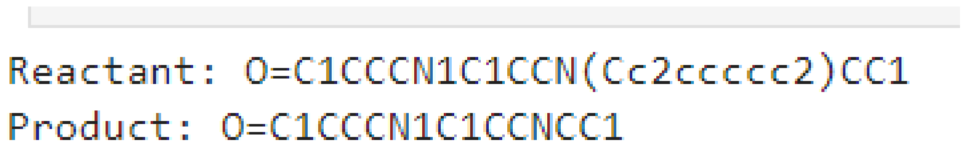
The smiles_to_graph function extends the conversion process by transforming the molecular structure into a graph representation. It first uses RDKit to generate explicit hydrogen atoms and then embeds the molecule in 3D space. The function then extracts atom features and constructs an adjacency matrix to capture the connectivity information between atoms. Atom features include the atomic number, the number of hydrogen atoms, and the formal charge.

The resulting data is converted into PyTorch tensors, and a PyTorch Geometric data object (Data) is created, which includes the atom features and edge indices representing the connectivity of atoms in the graph. This graph data object is suitable for training and utilizing GNNs, which can then be learned from the molecular structures encoded in the SMILES strings.

Separation of canonic reaction into Reactant and Product:

The function separate_compounds serve the purpose of parsing a given SMILES-encoded chemical reaction and separating it into its constituent reactant and product compounds.

The function accepts a SMILES reaction string as input, which normally consists of two molecular structures with the transformation from reactant to product indicated by the '>>' symbol separating them. The separator '>>' is then used to divide the reaction string into two compounds. It is anticipated that the final list, compounds, will have exactly two elements, signifying the reactant and product. Below is the example for reaction O=C1CCCN1C1CCN(Cc2ccccc2)CC1>>O=C1CCCN1C1CCNCC1



```
Reactant: O=C1CCCN1C1CCN(Cc2ccccc2)CC1
Product: O=C1CCCN1C1CCNCC1
```

Separated Reactant and Product for reaction

Vocab Size and Padding:

Preparation of data is done by tokenizing and padding the input sequences. The goal is to tokenize and standardize the sequences in specific columns—'reagent,' 'solvent,' and 'catalyst.' The process begins with tokenization word_tokenize function, breaking down each textual entry into individual tokens. These tokens are then combined to form a vocabulary unique to each column, capturing the diverse set of reactants, solvents, and catalysts present in the dataset.

To ensure uniformity in sequence length across all entries within a given column, a common maximum sequence length is determined based on the longest tokenized sequence in the entire dataset. Subsequently, a pad_sequence function is applied to each entry, adding padding tokens to the end of the sequences to match the determined maximum length. This step is crucial for maintaining consistent input dimensions, allowing for efficient batch processing during model training.

Through the creation of tokenized sequences, they standardize the representation of chemical components, making the subsequent modeling process easier. Second, the dataset is made suitable for use with machine learning models that need fixed-size inputs by padding sequences to a standard length.

Tokenizing and Mapping to Embeddings of data for training:

The first part involves the creation of a mapping between tokens and integers, allowing for efficient numerical representation of categorical data. The token_to_int function is designed to map each unique token to an integer value, assigning a random integer if the token is encountered for the first time. This mapping is essential for converting categorical information, such as reagents, solvents, and catalysts, into a format suitable for machine learning algorithms.

Subsequently, the tokenize_and_pad_sequence function is employed to tokenize a given sequence and pad it with zeros to achieve a consistent length specified by max_seq_length. This function utilizes the token-to-integer map function converting each token to its corresponding integer representation. The resulting sequence is then padded with zeros to match the predetermined maximum length.

The final sequence is converted into a PyTorch tensor, ensuring compatibility with PyTorch-based models.

Parsing of Data using Dataloader:

The ReactionDataset class is a PyTorch dataset tailored for chemical reaction data. It takes a Pandas DataFrame and a maximum sequence length as inputs. The class is structured to work seamlessly with PyTorch's data handling utilities.

The __init__ method initializes the dataset with the chemical reaction DataFrame and the desired maximum sequence length for padding. The __len__ method returns the total number of samples in the dataset, corresponding to the length of the DataFrame. The __getitem__ method retrieves an individual sample, transforming the raw reagent sequence into a padded PyTorch tensor and obtaining the canonical reaction associated with the sample.

Methodology

Our approach to predicting reaction conditions from reaction SMILES (Simplified Molecular Input Line Entry System) involves the utilization of a Sequence-to-Sequence (seq2seq) model with both an encoder and a decoder architecture.

The featurize_molecule function helps in the graph generation process by computing Morgan fingerprints for each atom in each molecular structure (mol). These fingerprints are generated using the RDKit function AllChem.GetMorganFingerprintAsBitVect, represent the local chemical environment around each atom within a specified radius. The function iterates over each atom, calculates the Morgan fingerprint, and appends the resulting feature to a list. The computed Morgan fingerprints are then returned as a NumPy array, encapsulating the local structural information for each atom in the molecule. Together, these functions provide a comprehensive approach to converting raw SMILES representations into graph-based data suitable for training graph neural networks.

Graph Attention Network

Choice of Graph Attention Network:

The choice to handle the SMILES-encoded reactions using a Graph Attention Network (GAT) stem from GATs' exceptional capacity to represent complex relationships seen in graph-structured data, including molecular structures. SMILES representations encode chemical reactions as graphs, where atoms and bonds form nodes and edges, respectively. However, GATs perform exceptionally well in this scenario by dynamically allocating attention scores to adjacent nodes as information spreads.

The molecular structure can be emphasized in different locations by the model, which assigns varying degrees of importance to specific atoms and links. The attention mechanism in GATs seems to be useful when managing reactions where the spatial arrangement of atoms plays a critical role in determining the outcome. The model's ability to recognize fine-grained structural elements and comprehend intricate patterns within the chemical transformations.

Working:

The GCN processes the featurized molecular graphs obtained from the smiles_to_graph and featurize_molecule functions, leveraging the Graph Attention Network (GAT) layer for feature extraction.

- The GAT Model class comprises two GAT layers. The first GAT layer (self. conv1) takes the input node features x and the edge indices edge_index of the molecular graph. The GAT layer performs graph convolution, aggregating information from neighboring nodes and producing an output feature representation for each node. The activation function ReLU is applied elementwise to introduce non-linearity to the output. The resulting feature representation captures the local structural information of each atom within the molecular graph.
- If external attention heads (external_attention_heads) are specified during the instantiation of the GAT Model, an additional step is introduced. An external attention mechanism, implemented by the MultiHeadAttention class, is applied to the output of the first GAT layer.
- This external attention mechanism computes attention scores based on the interaction between different nodes, enhancing the model's ability to capture long-range dependencies within the molecular graph. The external attention output is then concatenated with the output of the first GAT layer along the feature dimension.
- The second GAT layer (self. conv2) processes the concatenated feature representation obtained from the first GAT layer and the external attention mechanism. This layer refines the feature representations through another round of graph convolution, producing a final output that encapsulates both local and long-range structural information.

Encoder

Choice of Encoder:

In our sequence-to-sequence approach, the use of an encoder is essential to encoding and capturing the complex information included in the chemical processes expressed in SMILES strings. This inherent graph structure necessitates a specialized approach for feature extraction, making the encoder an essential component.

The model can comprehend the complex interactions among atoms, the influence of various reactants, and the function of solvents and catalysts in the reaction process as a result. The encoder is a crucial part of our sequence-to-sequence model, enabling it to produce meaningful and contextually aware predictions for reaction situations by utilizing the power of attention processes and graph-based learning.

Working of Encoder:

This Encoder will process the output from Graph Attention Network and generate a context vector based on pairwise distances and attention scores.

- Positional Encoding class: This embedding introduces positional information to the input sequence, aiding the model in capturing the order and relationships between elements in the sequence.
- The embedded sequence undergoes positional encoding using an embedding layer (self.encoding). This layer generates positional embeddings based on the positions of elements in the sequence. These embeddings are then added to the original input sequence, providing the model with information about the positions of tokens.
- The encoder employs a linear decoder (self. decoder) to generate attention scores. The attention scores are calculated as the hyperbolic tangent (tanh) of the encoded sequence. This design choice introduces non-linearity to the attention score calculation, allowing the model to capture complex relationships in the input data.
- The attention scores are subsequently passed through a SoftMax activation function (self. SoftMax) along the sequence dimension, yielding attention weights. The SoftMax operation normalizes the attention scores, converting them into a probability distribution. These attention weights reflect the importance assigned to each element in the input sequence.

The final step involves applying the attention weights to the encoded sequence, generating a context vector. This context vector is obtained by element-wise multiplication of the attention weights with the encoded sequence, followed by a summation along the sequence dimension. The resulting context vector encapsulates the most relevant information from the input sequence based on the learned attention mechanism.

Cross Attention

Choice of Cross Attention:

Cross-attention is used when processing two input sequences at once in sequence-to-sequence models or transformers. While generating the components of one sequence, the model pays attention to various parts of another sequence. The function concatenate_with_cross_attention seamlessly integrates the embeddings derived from two GAT models, corresponding to the reactant and product molecules, enriching the model's understanding of their molecular interactions.

Working:

Cross attention component of the architecture seamlessly integrates the embeddings derived from two GAT models, corresponding to the reactant and product molecules, enriching the model's understanding of their molecular interactions.

- In the get_cross_attention_output function, these processes are applied to a given SMILES-encoded reaction. The reactant and product are individually transformed into graph-based embeddings using GAT models.
- These embeddings are then fed into the concatenate_with_cross_attention function, facilitating cross attention and yielding an output embedding that encapsulates the enriched molecular interaction information.
- The function addresses potential disparities in the dimensions of the embeddings, ensuring uniformity by projecting the smaller embedding to the same dimension as the larger one. Subsequently, the embeddings are concatenated along the feature dimension.
- The use of MultiheadAttention to apply cross attention can dynamically focus on various joint embedding regions while considering the importance of chemical properties. The resulting cross_attended_emb provides a thorough

representation that is essential for downstream prediction tasks, encapsulating richer information about the reactant-product interaction.

- The function includes train_mode parameter to toggle between training and evaluation modes for the GAT models. If train_mode is set to True, the models are set to training mode; otherwise, they are set to evaluation mode.

Decoder

Choice of Decoder:

One key element in our sequence-to-sequence model that is essential to the prediction of reaction conditions is the TransformerDecoder module. Its significance stems from its capacity to decipher encoded data intelligently and produce precise predictions for a chemical reaction's settings. The output from the encoder will be passed to the decoder as an input for predicting the reaction conditions.

Working:

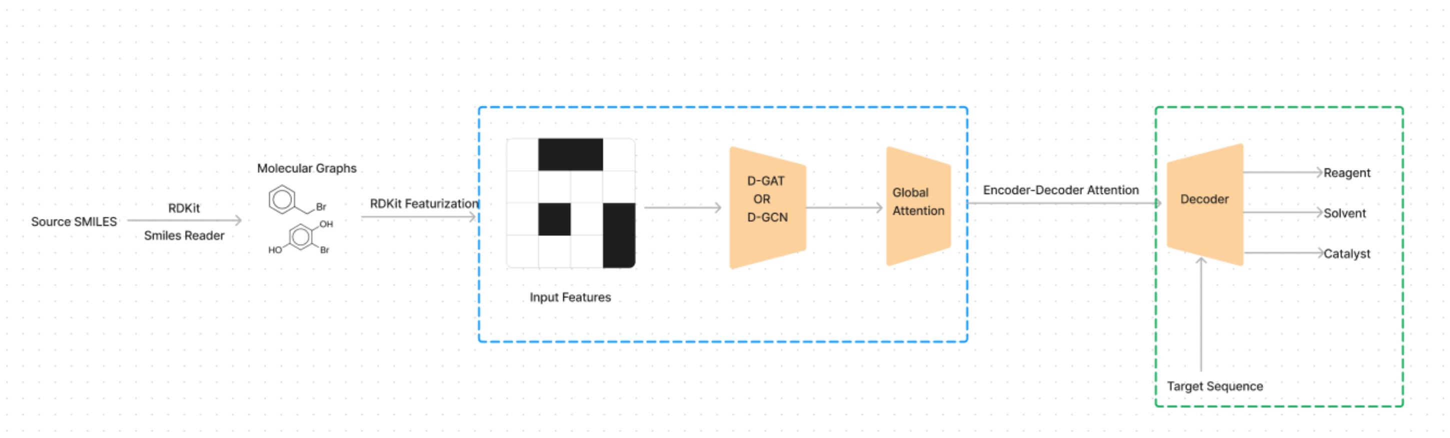
The combination of embedding layers, TransformerDecoder, and attention mechanisms allows our model to capture the intricate relationships and dependencies within the input conditions, leading to predicting reaction conditions.

- The model is designed to predict three distinct components related to reaction conditions. The input consists of three sequences: reagents, solvents, and catalysts. Each sequence is individually embedded using separate embedding layers.
- These embeddings are then combined and processed through a TransformerDecoder, where attention mechanisms capture complex relationships within the input sequences.
- The encoded sequence is further projected into a lower-dimensional space using a linear layer. The cross-attention mechanism is employed to combine information from the decoder and the encoder. Attention weights are calculated and used to mix the encoded and decoded features, creating a context vector.

This context vector is then projected to produce the final output for each of the three conditions, utilizing separate linear layers.

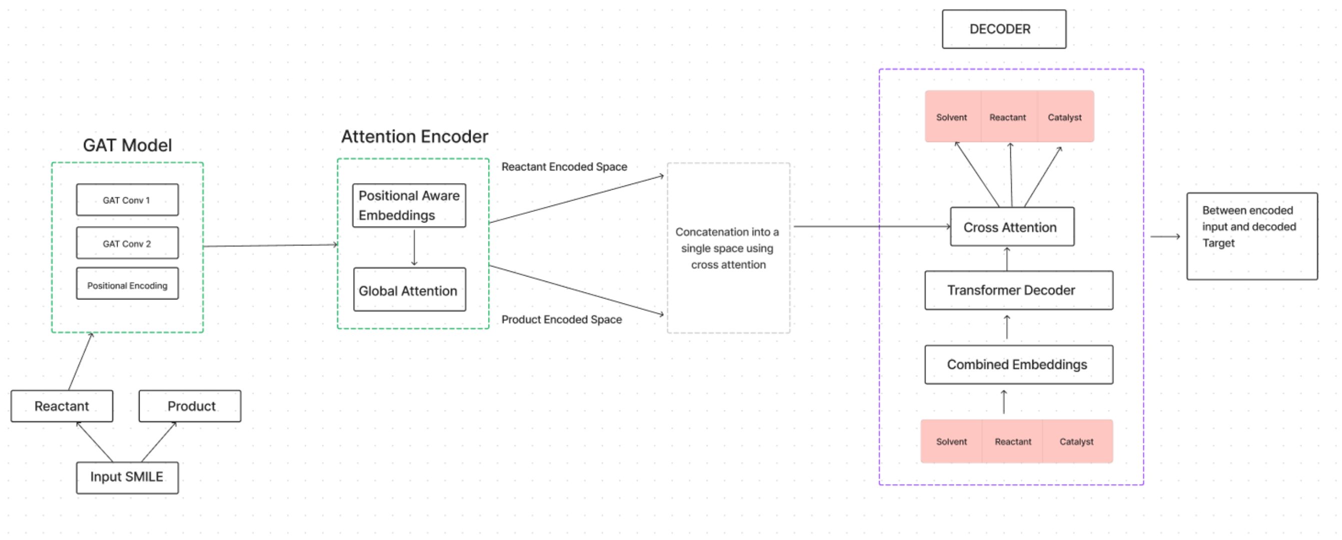
- The model incorporates non-linearity through ReLU activations in the final output layers, enhancing its ability to capture intricate patterns in the input data. The use of a Multihead Attention mechanism and the TransformerDecoder architecture enables the model to intelligently process and predict reaction conditions based on the given input sequences.

Flow of the Variables:



The flow of variables is happening as the source smiles are being converted into their respective Molecular graphs and then Rdkit featurization for both the smileys and encoding into one common space using cross attention mechanism which is parsed to the decoder which provides the probability outputs for the corresponding target sequence. During the time of inference, we initialize the target sequence as zeros and then predict the three classes based on input sequence.

Model Architecture:



This is the model architecture which comprises three major components: GAT model, Attention Encoder and Decoder.

Training

The train_model function uses a provided dataloader and optimizer to coordinate the training of a given sequence-to-sequence model. It starts by transferring the model to the assigned device and turning on its training mode. The method loops through batches of data from the supplied dataloader throughout each of the several epochs that iterates over. The Adam optimizer zeroes its gradients before processing each batch to avoid buildup. After doing a forward pass, the model produces predictions for every one of the three scenarios.

Then, between these predictions and the ground truth targets, the Mean Squared Error (MSE) loss is computed. The total loss for the epoch is updated by adding the current batch's loss. After processing all batches in the epoch, the average loss is computed. The Adam optimizer modifies the model's parameters based on the gradients computed by the backward pass, which updates the model's parameters. The function logs the average loss and current epoch and tracks the total loss for the epoch. Finally, the training performance of the sequence-to-sequence model is summarized by returning the average loss over all epochs.

Inferencing

The function find_key_by_value accepts a dictionary and a value as inputs and iteratively searches the dictionary for the key that corresponds to the supplied value. This allows you to map the integer-encoded tokens back to their original values, which is especially helpful for decoding the model's output.

The decode_input function decodes an encoded input sequence by using the find_key_by_value function. It builds a list of the decoded input tokens after iterating through the encoded input and utilizing the identified keys to obtain the relevant tokens from the token_to_int_mapping.

The inference function performs the model inference on a given input encoding. It loads the trained model, sets it to evaluation mode, and passes the input encoding along with a placeholder target sequence to the model. The initialized placeholder target sequence provides a starting point for the autoregressive decoding of the model, enabling the generation of accurate and meaningful predictions for the reagent, solvent, and catalyst conditions in the reaction.

Probabilities are generated by the model for every token for each of the three conditions. The decode_input function is then used to acquire the decoded representations of the conditions after converting the output probabilities to NumPy arrays. The decoded representations of the reagent, solvent, and catalyst conditions are displayed at the end of the function.

The output conditions from the inference process are stored in the o1, o2, and o3 variables, which stand for the predicted reagent, solvent, and catalyst for the specified input reaction, respectively.

Mathematical and Logical Reasoning

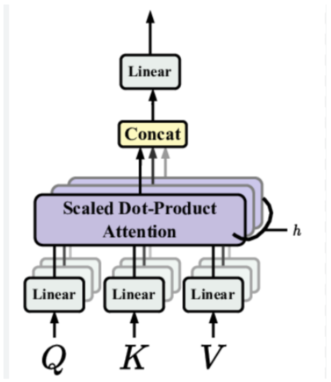
Mathematics for multi-head attention and cross attention.

Multi-Head Attention:

The multihead attention mechanism involves splitting the input into multiple heads, each operating on different linearly transformed projections. Let d_{model} be the dimensionality of the input, h be the number of heads, d_k be the dimension of each head.

1. Linear Projections:
   - $Head_i$ = Linear (xW_i^Q, xW_i^K xW_i^V $)_{where}$ W_i^Q, xW_i^K , W_i^V are learnable weight matrices for the ith head.
2. Scaled Dot-Product Attention:
   - Attention (Q, K, V) =softmax(QK^T)/(sqrt(d_k) x V
3. Concatenation and Linear Projection:
   - Multihead(x)=Concat($head_{1,}$ ....$head_i$)W^o where $W^o \in R^{hxd}{}_{model}$ is the output projection matrix.

Multihead attention allows the model to focus on different parts of the input simultaneously through parallel attention heads. Each head specializes in capturing different aspects or patterns in the input. By using multiple heads, the model can capture complex relationships and dependencies in the data. The capacity of the attention mechanism is enhanced, providing a more expressive representation.

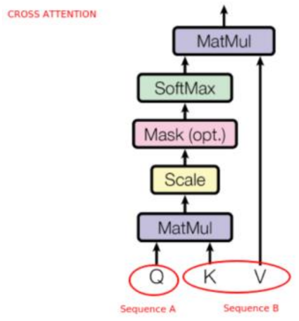


Cross-Head Attention:

Cross attention involves attending to a set of values (V) based on the interaction between a set of queries (Q) and Keys (K). For two sequences X and Y with dimensions d_X and d_Y .

1. Linear Projections for Queries, Keys, and Values:
   - $Q = XW_Q$, $K=YW_K$, $V=YW_V$ where $W_Q \in R^{d_X \times d_k}$, $W_K \in R^{d_Y \times d_k}$, $W_V \in R^{d_Y \times d_V}$
2. Scaled Dot-Product Attention:
   - Cross Attention (Q, K, V) = softmax(QKT)/(sqrt(dk) x V

Cross attention allows the model to capture interactions between different sequences or parts of a sequence. It is useful when generating output sequences based on information from another set of sequences. The attention mechanism facilitates the fusion of information from one set of sequences to another, capturing relevant dependencies.



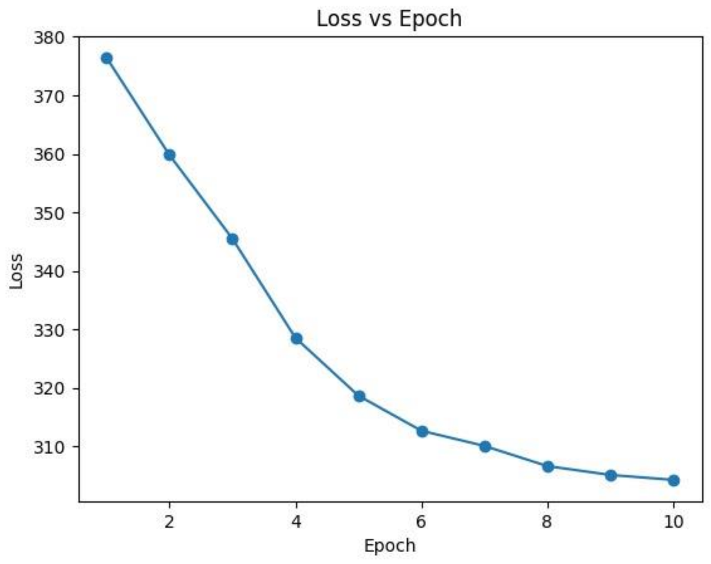
Graph Attention Network (GAT) is chosen for SMILES-encoded reactions because it can represent complex interactions in graph-structured data, which is like chemical structures. Deciphering the spatial arrangements crucial to reaction outcomes is made easier with the help of GAT's dynamic attention allocation to neighboring nodes. Simultaneously, our sequence-to-sequence model extracts intricate chemical reactions from SMILES strings using an encoder. Precise predictions are made possible by the encoder's graph-based learning and attention algorithms, which enable the model to understand complex structural details. By deciphering encoded data intelligently, the TransformerDecoder further refines predictions, resulting in a holistic strategy that accurately anticipates a variety of reaction situations.

Evaluation Metrices

Loss:

Mean Squared Error (MSE) is a commonly used loss function in regression problems, including those related to predictive modeling in machine learning. In the context of your model, MSE is employed to quantify the difference between the predicted values and the actual target values. Specifically, it calculates the average of the squared differences between the predicted and true values for each data point.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

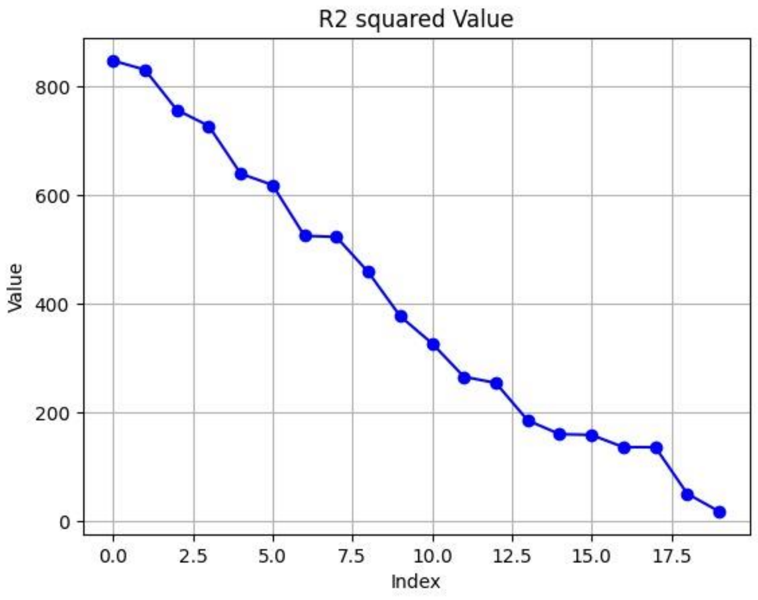


Our model was trained across ten epochs, and as you can see, the loss keeps getting smaller as the number of epochs increases. In general, IT ranges from 300 to 400. It can be kept to 100 with some nice effects if we train additional epochs.

R2 Score:

The R2 Score or the coefficient of determination, is a metric commonly used to assess the goodness of fit of a regression model. It provides a measure of how well the predicted values from the model match the actual values. The R2 score is a normalized metric, with values ranging from 0 to 1, where 1 indicates a perfect fit.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

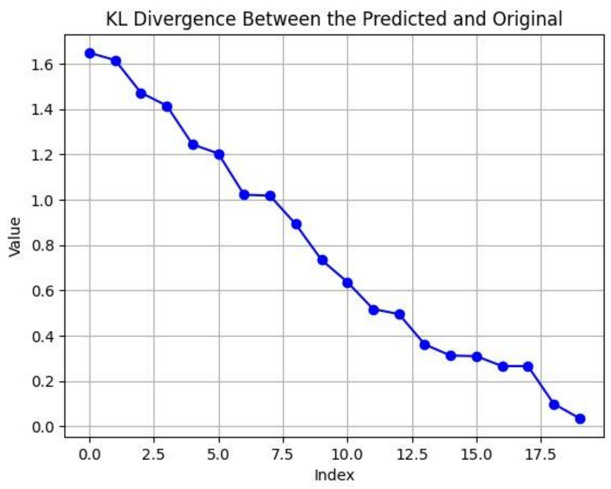


The R2 Score is also being decreases with increasing steps, showing the model is learning something and showing much better results than previous steps.

KL Divergence:

Kullback-Leibler (KL) Divergence is a measure of how one probability distribution diverges from a second, expected probability distribution. It is often used to quantify the difference between two probability distributions.

$$D_{\mathrm{KL}}(p(x)\,||\,q(x)) = \sum_{x \in X} p(x) ln \quad \frac{p(x)}{q(x)}$$



KL divergence is used for checking the similarity between two probability distributions, as we have our outputs in form of probabilities, the decreasing values shows that the two probabilities are becoming much closer with increasing steps.

Benchmarking

In benchmarking our model, it is pertinent to acknowledge that existing research and models, particularly those applied to the Choriso dataset, primarily focus on predicting reaction yield and predicting reaction product rather than predicting reaction conditions.

While our problem statement centers on forecasting the precise conditions of a chemical reaction, the existing literature often addresses the broader challenge of estimating reaction yield.

Consequently, our benchmarking process encounters a unique context wherein the available models and approaches cater to a distinct facet of chemical reaction prediction. While leveraging insights and techniques from previous works on the ChEMBL-29 dataset, we must recognize the fundamental differences in the predictive tasks. The available models, though invaluable for predicting overall reaction outcomes, may not directly align with the intricacies of our task—predicting reaction conditions with a focus on reagents, solvents, and catalysts. This distinction underscores the need for a nuanced evaluation, recognizing the specialized nature of our predictive task within the broader landscape of chemical informatics.

Future Scope

In the realm of future enhancements, several promising avenues can be explored to augment the capabilities of our current model. First and foremost, the integration of more advanced attention mechanisms, such as flash attention, holds the potential to further refine the model's ability to capture intricate relationships within chemical reactions.

Additionally, the development of an interactive user interface will not only facilitate user-friendly interactions but also empower chemists and researchers to actively engage with the model, input reactions, visualize predictions, and contribute valuable feedback, thereby fostering a collaborative and iterative model improvement process. Furthermore, considering the dynamic nature of chemical reactions, the adoption of deeper sequence-to-sequence models could enhance the model's capacity to capture nuanced patterns and dependencies in reaction conditions.

These future directions aim to elevate the model's performance, usability, and adaptability, contributing to its effectiveness in predicting reaction conditions across diverse chemical domains.

GitHub Repository Link

https://github.com/vageesh1/BioChem-Hackathon