Background Research
2024 Spring
Week 1: 02/06-02/13
I initially became interested in antibody-drug conjugates when I came across a research paper discussing their potential for use in cancer treatment, so I decided to look into them further.

- antibody drug conjugates (ADCs)
    - target antigen
        - must be higher expression in tumor + specific
        - generally surface receptors on tumor cells, tumor stem cells, or found within tumor vasculature (develop as tumor grows larger) and microenvironment (ecosystem surrounding a tumor in a cell)
        - expressed homogeneously → conjugation in specific sites
    - monoclonal antibody
        - size
            - generally quite large, to make smaller use Fab-drug conjugates, scFv-drug conjugates, and diabody-drug conjugates
                - smaller would result in faster clearance
        - internalization
            - necessary component for drug to be delivered
            - looking into targeting structural components of tumor environment
                - helps overcome penetration barriers of solid tumors
                - target tumor stroma
        - composed of two main parts:
            - antigen binding fragment (Fab)
                - creates specificity in the target
            - crystallizable fragment (Fc)
                - drives biological function of the antibody
            - these two affect specificity, durability, and the outcome of the antibody
        - bispecific antibodies (bsAbs)
            - composed of a Fab which binds to two different targets
            - current problems: design/compatibility → different parts of the antibody interfere with each other
        - generally use IgG1 because of higher delivery capability and effector functions
            - would use IgG2 and IgG4 isotopes depending on the target characteristics and proposed mechanism of action (MOA)
            - isotope differences affects conjugation
        - sources
    - cytotoxic payloads
        - only 1-2% reach the intracellular target so typically require high potency
        - use tubulin-polymerization disruptors or induce DNA damage
        - one existing ADC has lower potency which allows for a higher DAR → more stable
        - tubulin polymerization
            - targets the beta subunits of tubulin dimers

                - use MMAE/MMAF as promoters → increase stability of cell division
                - use DM1/DM4 as inhibitors → reduce cell division by preventing mitosis
            - DNA damaging agents
                - reach the picomolar level which is smaller than others which are nanomolar
                - can be more effective and work against lower antigen expression cells as well
                - ex. calicheamicin used in some ADCs
            - immunomodulators
                - immune stimulating antibody conjugates (ISACs)
                - newer concept, currently in clinical trials mostly
                - payloads include toll like receptor (TLR) agonists and stimulator of interferon genes (STING) agonists
                    - pattern recognition receptors in innate immunity
    - linkers
        - cleavable
            - in response to the environment, proteases, or reducing agents → can lead to premature releases sometimes
            - non-internalizing ADCs rely on glutathione and proteases shed from tumor cell death
        - non cleavable
            - resistant to degradation, the full degradation of the antibody is required which can overcome drug resistance
        - can balance hydrophobicity between mAb and payload to prevent aggregation, ensures stability of the ADC
    - sources
      https://www.tandfonline.com/doi/pdf/10.1080/19420862.2021.1951427

Specifically, I thought the linker was particularly interesting and significant in its affect in ADC toxicity, so I looked into that further.

Week 2: 02/13-02/20

- **Choosing the Best Linker**
- conjugation chemistry, length, and steric hindrance are important factors affecting PK and efficacy
    - steric hindrance: slowing of chemical reactions due to steric bulk
    - affected by spatial arrangement of atoms → induces nonbonding interactions that influence shape/reactivity of ions and molecules
- linker decoupling and degradation → important biotransformation pathways of the ADC in the body
- my project: not really focusing on conjugation site since only focused on developing a novel linker, not the entire drug
    - can look at potential conjugation sites still as a way to evaluate the linker's success
- adjusting conjugation site, linker length, and linker steric hindrance → effective methods for developing intact/stable ADCs

- studies show: **shorter linker = increased ADC stability**
- maleimide & disulfide bonds → main types
    - generated linkers need chemistry to make these bonds
    - steric hindrance near cleavage site is critical, stability of conjugate can be adjusted by chemical modification
    - disulfide → new strategy of cleavable linkers
        - **need to be able to self degrade** after the linking group is broken
        - something I should take into account for my model and the linkers it produces
- conclusion:
    - depends on the drug, should find a good balance of the important parameters
        - conjugation site, conjugation chemistry, linker length, cleavable vs non-cleavable, local steric hindrance
        - also consider dosage design
- Linker: source (https://pmc.ncbi.nlm.nih.gov/articles/PMC8262647/)
    - acts as bridge between antibody + payload - mostly contributes to stability and efficacy of an ADC
    - payload release also depends on the type and nature of linker
    - ideal linker: should be highly soluble in water - prevents formation of aggregation and premature release of payload in the systemic circulation
        - **conjugation site** - look into this aspect further
    - shorter linker is responsible for better stability
        - tie up payload closer to steric shield provided by antibody
- characteristics of linkers
    - molecular weight
        - small → premature cleavage potential
        - large → more stable, could interfere with ADC functionality, can affect circulation time
        - therefore, both have pros/cons and there's a general range of acceptability set out by GPT ⇒ 200 - 1000 Daltons
            - also says depends on what's common with ***acceptable standards*** - they all fall within a 200-600 range
        - affect stability and cleavage
            - h-bonds stabilize the linker by forming interactions that prevent degradation and help maintain conformation integrity
        - can increase hydrophilicity - prevents aggregation through the bloodstream
        - can facilitate the payload release - ex. low pH, specific enzymes, etc.
        - following Lipinski's rule of 5 (not necessarily relevant for linkers)
            - should be less than 5 h-bond donors (too many decreases permeability) and less than 10 h-bond acceptors (too many affects lipophilicity)
            - however, values vary depending on the ADC and its overall properties, so don't need to take into too much account, as long as its within reasonable range it's acceptable and would need further testing to explore more

- TPSA (surface area)
  varying values
    - higher = hydrophilic → can have difficulty crossing lipophilic cell membranes
    - generally moderate values are ideal, overall better performance
    - allows ADC to stay soluble in bloodstream and effectively reach cancer cells
  - can just go with average/range of values in approved linkers
- logP values
  - negative = more hydrophilic, positive = higher concentration in lipid phase
  - should be generally negative, differs from different sources
  - mainly should probably aim to have lower, more hydrophilic is better in general

Types of Linkers:

- Different types of linkers
  - Cleavable
  - hydrazone
    - hydrolysis occurs in acidic endosomes
    - common to cause toxicities and has low tolerability
    - found in Mylotarg, BR96-DOX, and others
    - undergoes slow hydrolysis under physiological conditions which leads to premature payload release
  - cathepsin-B responsive
    - cathepsin-B is a lysosomal protease that is over-expressed in cancer cells
    - it has a broad scope of substrate
      - generally prefers Phe-Lys, Val-Cit, and others
      - tends to cleave the peptide bond on the C-terminal side
    - common existing ones, Val-Cit-PABC and Val-Ala-PABC
      - they are cleaved and released tracelessly
    - PABC
      - functions as a spacer between Val-Cit and the payload
    - found in Acetris
  - glutathione sensitive
    - common cleavable linker
    - higher concentration of glutathione in the cytoplasm compared to extracellular environment
    - disulfide bond embedded in the linker
      - resists reductive cleavage in circulation
        - upon internalization, abundant glutathione reductively cleaves the bond
        - enhance stability → methyl groups next to disulfide bond
    - found in Mylotarg and maytansine-based clinical trials
  - pyrophosphate diester linker
    - novel cleavable linker structure
    - greater aqueous solubility and good circulatory stability

        - promptly cleaved through endosomal-lysosomal pathway to liberate unmodified payload
            - two step cleavage
            - enzymes involved are not yet identified
    - non-cleavable linkers
    - stable bonds resistant to proteolytic degradation → greater stability in the bloodstream
    - antibody is completely degraded by proteases and liberates the payload molecule
        - payload molecule ends up linked to amino acid residue
        - payload structure carefully selected and designed so payload can exert comparable/better anti-tumor potency in modified form
        - PK/PD and toxicity profiles of possible metabolites of ADCs need to be examined
        - found in T-DM1

Ultimately, I found several issues with linker generation related to their toxicity problems, as well as some examples of ADCs that didn't work because of their linkers, so I thought about how linker generation could be optimized. Considering the rapid increase in use of generative AI for tasks nowadays, I saw the potential to apply that concept to linker development.

Week 3: 02/20-02/28

General machine learning in drug discovery notes:

- https://link.springer.com/article/10.1007/s10462-021-10058-4
- steps:
    - finding materials, consumables
    - chemical synthesis
    - data mining, ontology, biology research (target validation)
    - lead discovery
    - preclinical development
    - biomarker discovery
    - clinical development
    - pharmacovigilance
    - drug repurposing
    - virtual lab assistants
- Section 2: Application of AI in drug design
    - protein structure
        - deep learning and feature extraction tools
        - gains information on connection among structure and sequence through feature extraction
        - goal: predict 3D-protein structure by utilizing deep-learning techniques to improve accuracy
    - drug-protein interactions
        - FINDSITEcomb
        - database focuses primarily on therapeutic protein targets
        - computational techniques fill knowledge gap for predicting protein targets
- Section 3: Machine learning methods

    - Hidden Markov model, hierarchial clustering, K-means, neural networks (DAENs and autoencoders)
        - de novo molecular design
        - deep feature selection for biomarkers
        - reduction in single-cell data to identify cell types
        - cell types and biomarkers from single-cell RNA data
        - unsupervised
    - deep learning accelerates features → available initialized data
        - multi-layer feature extraction techniques
        - ex. CNN, RNN, Auto Encoder, DNN, and RBN
    - multilayer perception
        - feed-forward neural network
        - provides an outcome based on a set of input sources
        - training information → uses backpropagation approach
        - multiple layers are essentially input nodes and output nodes
            - data processed, then perceptron can fluctuate with connected weight in the network (weight determines the connections between neutrons in adjacent layers)
        - ex. MLP used in prediction action between drugs
            - does not require structural information → uses experimental data to predict accuracy
            - utilized to generate a de-novo drug design
        - greater success in classification success → secondary structure of proteins = greater advantage in determining protein function
            - secondary structure = more difficult/expensive
    - deep learning
        - neural network architecture with several layers, data is transformed between these layers
        - can be developed through greedy layer-by-layer strategy
        - pooling layer → structure that hinders neural networks
            - capacity of pooling layer reduces spatial size of representation → reduces boundary measurement and system computations, work independently on each feature map (channel)
- Section 4: Drug design applications
    - precise training, validation, and application of ML algorithms required
    - QSAR → huge data collection and training of datasets → rate-limiting steps in defining ligand-based virtual screening protocols
        - replaced by Denovo design techniques
    - predicting protein 3D structures and protein-protein structures…
        - fragment docking
        - protein-protein interactions

This general research led me to diffusion models, a relatively new type of generative model. I thought their use in image generation was really interesting and it worked well, so I looked into them further.

- **Diffusion Models**

- https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/
- generative models
    - generate data similar to data trained on
    - destroy the training data by adding Gaussian noise, learn to recover the data by reversing noising process
    - after training, use diffusion model to generate data → pass randomly sampled noise through learned denoising process
    - Gaussian noise
        - statistical noise with pdf (probability density function) equal to Gaussian distribution
    - latent variable model → maps latent space using fixed Markov chain
        - chain gradually adds noise to data, obtains approximate posterior q(x1:T|x0), where x1, …., xT are latent variables with same dimensionality as x0
        - latent variable model
            - contains latent (unobserved) variables
        - Markov chain
            - process consisting of a finite number of states with Markovian property and some transition probabilities pij (probability of moving from state i to state j)
            - can be used to describe traffic, something inherently unpredictable
                - also, how network will perform
    - image is asymptotically transformed to Gaussian noise
        - learn reverse process (training p0(xt-1 | xt)
    - advantages:
        - no adversarial training required
            - adversarial training
                - apply small perturbations to original input data
                - crafted to make model produce incorrect/undesired outputs
                - used to fool/misguide a model, such that algorithm learns from mistakes and adjusts parameters
        - scalability and parallelizability as well

Other diffusion model notes:

Week 4: 02/20-02/28

https://arxiv.org/pdf/2302.10907

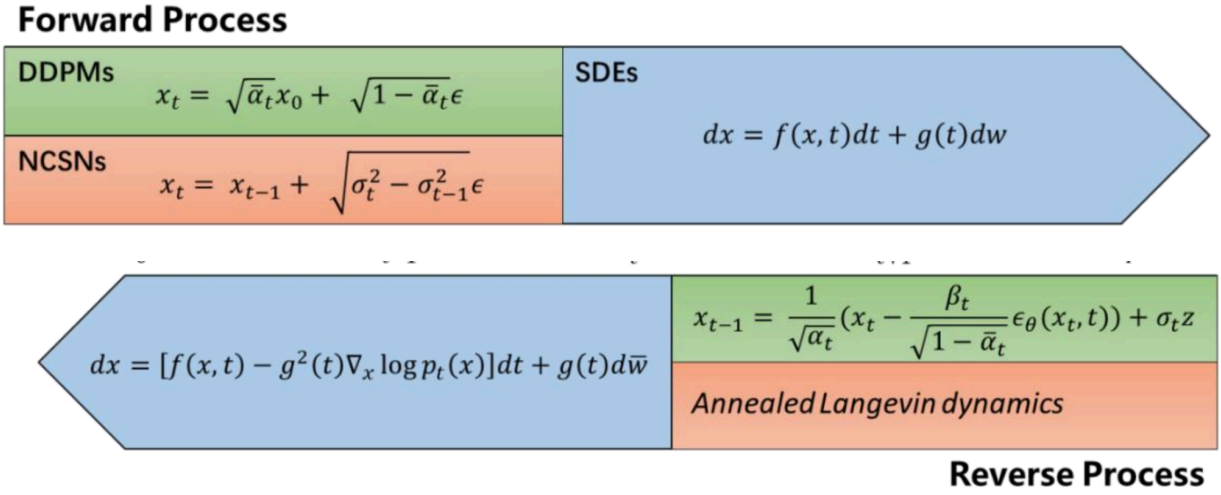
- denoising diffusion models have been used in computer vision, NLP, and bioinformatics
    - lack of overview of applications in bioinformatics

introduction:

- diffusion models generate artificial yet realistic data from input parameters
- advantages over other generative models → learn complex distributions smoothly, handle high-dimensional data, generate extremely diverse data
- address bioinformatics problems
    - denoising cryo-EM data, single-cell gene expression analysis, protein design,drug and small molecule design, and protein ligand interaction modeling

concept + foundation of diffusion models

- reverse process of data destruction/corruption for generation of realistic, clean data samples
- based mainly on three predominant formulations: denoising diffusion probabilistic models (DDPMs), noise-conditioned score networks (NCSNs), and stochastic differential equations (SDEs)



- DDPM has two Markov chains
    - Forward gradually adds noise, reverse removes noise to recover original
    - … math stuff
- NCSNs … math stuff
    - Main difficulty of training -> trained score functions were unreliable in low-dimension manifold because data usually lied in a low-dimension manifold embedded in a high-dimensional space
        - Solved by introducing Gaussian noise to the data at various scales, improves data distribution's suitability for score-based generative modeling
- Stochastic differential equations (SDEs)
    - Gradually transforming data into noise
    - Forward process of formulation score SDE uses stochastic differential equations and requires estimated score function of noisy data distribution

Applications of diffusion models in bioinformatics

- Cryo-em data analysis
    - cryo-EM (single-particle cryo-electron microscopy) is a key imaging technique
        - Can determine and visualize 3D conformation of large biomolecular complexes
        - Can then reconstruct 3D conformation represented by 3D density maps
- Single cell image and gene expression data analysis
    - Reconstructing 3D shape of single cell from individual single-cell 2D microscopy image
    - Uses 2D image of individual cell as inductive bias
        - Constraints model with 2D image to predict 3D cell shapes
        - Predict infinite number of probable reconstructions

        - Single cell RNA sequencing can measure expression of genes in individual cells
            - Low RNA quantities in individual cells -> highly noisy measurements of gene expression, especially missing values
            - Important to denoise and impute missing values
            - Uses KNN-G (K-Nearest Neighbor Graph) to select denoising hyperparameters via noise2self self supervision method
- Protein design and generation
    - Designing protein sequences that can fold into structures is important for designing novel proteins to carry out specific functions more effectively than natural proteins
    - Many types…
- Small molecule generation and drug design
    - Designing small molecules to mediate protein structure + function
    - Fragment based drug design, strategy for discovering new small molecules (such as drug candidates) in 3D space
    - Design linkers consisting of atoms to connect fragments into complete, viable molecule
    - DiffLinker generates molecular linkers to connect molecular fragments
        - Graph neural network predicts size of linker
        - Equivariant diffusion model generates linker between input fragments
        - Use idea, consider more biological standpoints
            - Small molecule drug design, linkers connect two molecular fragments
                - Short + rigid, maintains desired orientation of connected fragments
                - Simple chemical bonds, small in size
            - ADC, linkers ensure drug is released at right time and place (more constraints)
            - Must be stable + cleavable, long enough for drug to remain functional + active after being released, not too long though
- Protein-ligand interaction modeling
    - Predicting conformation of ligand bound with protein
        - Studying protein ligand interaction, protein function, and discovering drugs
        - Prediction ligands: protein conditioned generation and ligand inpainting generation
        - First produces new ligands given fixed protein pocket, second learns joint distribution of ligand and protein pocket first, and then ligand is inpainted during interference time

DiffLinker https://github.com/igashov/DiffLinker/tree/main

- ReadMe
    - Places missing atoms in between and designs a molecule incorporating all the initial fragments

Week 5: 02/28-03/06

- started feature selection for the ADC components

Feature selection

NLP code

```
from rdkit import Chem

n_gram_list = [1, 2, 3, 4, 5, 6, 7, 8]
SMILES = ('C(C(=O)NCC(=O)NCC(=O)NCC(=O)NCC(=O)O)N')

def getKmers1(sequence, size=6):
  if sequence != None:
    lst = []
    X = None
    for i in [1]:
      for x in range(len(sequence) - i + 1):
        try:
          X = Chem.MolFromSmiles(sequence[x:x + i])
        except SyntaxError:
          pass
        if X is None:
          continue
        else:
          lst.append(sequence[x:x + i])
    return lst
  else:
    return None

kmers = getKmers1(SMILES, size = 6)
print(kmers)
```

- this was mostly just copied off of how the instructions were set to get the embeddings, works pretty easily
- I also explored tools like FG-BERT, but they didn't seem to work as well and not as user friendly, this went into pretty high dimensions as well which captures lots of information so I went with this

ESM-2:

- pretrained language models for proteins
- hugging face → code and pre-trained weights
  - weights → connection management between two basic units within a neural network
  - ESMFold and ESM-2, previously released ESM-1b and ESM-1v
  - read about ESM-2 (one used in ADCNet)
  - ESM-2 outperforms all tested single-sequence protein language models and enables atomic resolution structure prediction

    - ESM → trained with masked language modeling objective
        - **masked language modeling**
            - mask some words and predict what replaces these masks → gets a statistical understanding of the language the model was trained in

in designing the diffusion model, i looked at diffusers by hugging face a lot, since it had designed diffusion models as well as tutorials and general packages that could be borrowed and applied in my diffusion model

1. diffusers by hugging face
    - usability over performance, customizability over abstractions
        - can work for 3d molecules
    - three components of the library
        - diffusion pipeline
            - high level end to end class that rapidly generates samples from pretrained models
        - popular pretrained model architectures and modules → building blocks to create diffusion systems
        - schedulers
            - algorithms controlling how noise is added for training, how to generate denoised images during inference
    - tutorial (need to do the tutorial, find out how to work with 3d molecules)

Week 6: 03/06-03/13

- started to look at how to represent the linkers in lower dimensions, can get more information

UMAP

- UMAP parameters
    - **n_neighbors**
        - controls how UMAP balances local vs global structures
        - constrains size of local neighborhood when learning manifold structure of data
        - low values → concentrates on very local structures, can be detrimental to big picture
        - large values → looks at larger neighborhoods, loses fine detail structure
        - n_neighbors = 4 works well for MFP plot
            - shape changes significantly when the n_neighbors changes, is this normal
        - default is above 15
    - **min_dist**
        - controls how tightly UMAP packs points together
        - lower values → clumpier embeddings
        - useful if interested in clustering/finer topological structure
        - larger values prevents packing points together, focuses on preservation of broad topological structure

Hyperparameters:

- Hyperparameters
    - https://symbl.ai/developers/blog/a-guide-to-llm-hyperparameters/
    - num_epochs

        - epochs are the number of iterations of all the training data in one cycle to train the ML model
            - frequency at which training data passes through algorithm
            - num_epochs = 1 means all training data used at once
        - find ideal number
            - too many = overfitting
            - too little = poor performance
    - gradient_accumulation_steps
        - technique to train on bigger batch sizes than model can normally fit in memory
            - accumulates gradients over several batches, doesn't need to update model weights after each batch
            - gradient measures change in all weights with regard to change in error
                - weights → connection managements between two basic units in a neural network
            - higher gradient = faster model learning
    - learning_rate
        - fundamental hyperparameter controlling how quickly the model updates in response to calculated loss function
            - moving towards minimum of loss function
            - lower increases stability and improves generalisation → lengthens training time
            - higher expedites training process → has some instability and overfitting
    - batch_size
        - determines how much data the model processes each epoch
            - often restricted to hardware capabilities
            - small batches require less memory and compute power
                - each data point processed more thoroughly
        - probably not that important for this because only 162 linkers
    - max_output_tokens
        - max_sequence_length
            - max tokens LLM can generate as output
            - typically, higher = more coherent and contextually relevant response
                - longer also = more inference which increases computational and memory demands
                - lower may result in incoherence and errors
                - benefits of lower:
                    - boosts other aspects of LLM's performance (ex. throughput or latency) and expedite process by lowering inference time
                    - control inference costs
                    - constrain amount of generated text so it conforms to specific format (**probably constraints on size for linkers in ADCs)
    - decoding type

        - decoding is when selected output is converted from vector embeddings into tokens then presented to the user as a response
            - greedy and sampling decoding
            - greedy → choose token with highest probability at each step
            - sampling → chooses a subset of potential tokens and selects one at random → creates more variability/randomness, desirable in creative applications of language models

Week 7: 03/13-03/20
Coding the diffusion model:

- diffusion model
    - neural network with two main components
        - context encoder - encodes combined heavy, light, and payload embeddings
        - linker generator - predicts new linker embeddings based on encoded context and a time step
            - time embedding - encodes the time step into a feature vector for use in the generation process
- training
    - diffusion_loss_fn function defines training loss
        - Mean Squared Error between predicted noise and actual noise
            - diffusion process: noise is gradually added and then removed from the data to generate new embeddings
            - computes the loss using diffusion_loss_fn - applies gradient clipping to avoid exploding gradients, parameters updated using Adam optimizer
- generation
    - reverse diffusion process to generate novel linker embeddings - conditioned on heavy, light, and payload embeddings
- potential problems:
    - vanishing/exploding gradient
        - occurs when gradients are too small for the model to learn effectively
            - exploding gradients can cause values to grow uncontrollably

After my first version of the diffusion model, there were some problems, there are some of the changes I made:

- changed the model architecture, added an attention layer
- loss values stay stagnant mostly, relatively low
- values are about the same for embedding, slightly improved
- training dataset only 80%, validation 10%, testing 10%
- added validation into it

Week 8: 03/20-03/26
Decoder
Problems + potential fixes

- model architecture
    - add a bidirectional LSTM (backward and forward dependencies)

    - add attention mechanism, help model focus on relevant parts of input
    - increase latent dimension: current is 256, can try increasing to learn complex patterns
    - embedding layer: add embedding layer to project input embeddings into more suitable space for processing
- model training
    - teacher forcing: use true token as next input instead of predicted token (helps model learn correct sequences faster)
    - curriculum learning: start training with shorter sequences, gradually increase sequence length (don't think I can do this)
    - hyperparameter tuning (use AI tools to find best learning rates, optimizers, batch sizes, etc.)
    - gradient clipping (avoid exploding gradients), stabilizes training can improve performance
- handling overfitting
    - techniques such as dropout in encoder and decoder, can add L2 regularization to dense layer to prevent overfitting
    - data augmentation (probably can't do)
- evaluation metrics
    - use validation set (monitor additional metrics like BLEU score or Levenshtein distance to evaluate similarity)
    - SMILES validity check, after generating validate them which gives additional feedback in training
- post training improvements
    - beam search decoding (replace greedy decoding)
        - considers more sequences
    - identify common errors and fine tune model accordingly

coding the decoder:

- tokenization refinement
    - ensure ring closures, bond types, and stereochemistry are properly captured
- penalty for invalid characters and structures
    - penalize invalid characters or unmatched parentheses
    - *have tried before, might not work, can try again
- ring closure management
    - usually doesn't work that well
    - ensure that when a number appears (ring opening) the corresponding closure is also predicted
- modify beam search
    - avoid sequences where SMILES tokens cause parsing errors early
- increase training data by augmenting SMILES dataset
- pretrained models for embeddings
    - chemBERTa during training to generate more accurate SMILES embeddings (hasn't worked in the past)
- sequence validator in training
    - adjust training weights - can try this

- more epochs?
- post processing rule based approach

Week 9: 03/26-04/02

- After generating the linkers, I need to evaluate them so I looked into several small molecule drug evaluation tools, some of the simplest as well as accurate models were chosen -> ProTox and SwissADME

Pro-Tox

- toxicity evaluation - important step in drug and chemical design
- green toxicology approach reduces number of animal tests and chemical usage during toxicity testing
    - green toxicology promotes and encourages use of new and innovative techniques/strategies reducing animal use in testing, amounts of chemicals used/disposed of, and increasing consideration of toxicity in synthesis, use, and regulation of chemicals
- specify the SMILES string of the input
- results:
    - information on acute toxicity class dose
        - LD50, toxicity class, and prediction accuracy
        - includes three structures of three most similar compounds in the dataset
    - distribution of physicochemical properties
    - prediction for organ toxicity, toxicological endpoints, toxicological pathways, molecular initating events, and metabolism with respective predicted class and confidence score
    - predicted toxicity targets with info on target's name, average fit, and similarity of input compound concerning pharmacophore and known ligands
- models based on Random Forest machine learning and deep neural network algorithm
    - avoids overfitting, performs better than deep learning-based models
- validated using 10-fold cross-validation

SwissADME

- important values
    - physicochemical properties
    - lipophilicity - consensus value of five models
    - LogS scale (3 different values)
        - insoluble < -10 < poorly < -6 < moderately < -4 < soluble < -2 < very < 0 < highly
    - GI absorption, BBB permeant, P-gp substrate, CYP450 inhibitors, Log Kp (skin permeation)
        - variety of models - GI - high/low, permeant/substrate/inhibitor = no, log Kp - value of cm/s
    - druglikeness
        - Lipinski, Ghose, Veber, Egan, Muegge - have specific requirements - yes/no for violations
        - Abbott bioavailability score - probability F > 10 % in rat
    - synthetic accessibility score - 1-10 (1 is very easy)

comparing different dimensionality reduction techniques (ending up selecting UMAP because it aligned with my project the best)

1. linear vs non-linear techniques
    1. PCA is linear
        - transforms data into lower dimensional space by finding new axes that maximize the variance in the data
        - suitable for datasets where relationships between variables are approximately linear
    2. t-SNE and FIt-SNE are non-linear
        - designed to preserve the local structures of data
        - excellent for visualizing complex datasets with non-linear relationships
        - t-SNE works by minimizing divergence between two distributions: a distribution that measures pairwise similarities in high dimensinoal space and similar distribution in low-dimensional space
    3. UMAP and LargeVis are non-linear
        - focus on preserving both the global and local structure of data
        - UMAP - constructs a high dimensional graph and then optimizes a low dimensional graph to be as structurally similar as possible
        - LargeVis constructs a k-nearest neighbor graph and optimizing it to map the structure of data
    4. Laplacian Eigenmaps is nonlinear
        - aims to preserve local structure by constructing a graph of data points and computing the eigenvectors of the graph Laplacian
        - useful where local relationships are more important than global structure
    5. global vs local structure
        - PCA focuses on preserving global structure
            - emphasizes maintaining overall variance and relationships across the entire dataset
            - makes it suitable for tasks where global variance is more important than local details
        - t-SNE, FIt-SNE and Laplacian Eigenmaps focus on preserving local structure
            - maintain proximity of datapoints within clusters
                - visualize the intrinsic grouping in the data
                - uncover patterns not visible when only considering global structure
        - UMAP and LargeVis
            - strike a balance, effective for visualizing and understanding overall distribution of data
    6. Hyperparameters
        - PCA - no hyperparameters, straightforward to implement and interpret
            - main decision is # of principal components to retain (determined by explained variance)
        - t-SNE and FIt-SNE involve several hyperparameters

        - perplexity, learning rate, number of iterations - may require careful tuning
    - UMAP
        - number of neighbors (controls local structure) and minimum distance (which influences tightness of clustering)
        - adjusted to emphasize different aspects of data
    - LargeVis
        - number of neighbors, number of negative samples, affects construction and optimization of the graph
    - Laplacian Eigenmaps
        - require setting number of nearest neighbors
7. Suitability for Data Types
    - PCA - linearly separable datasets
        - relationships are approximately linear
    - t-SNE, FIt-SNE, UMAP, and LargeVis
        - non-linearly separable datasets
        - capture complex patterns and relationships
    - Laplacian Eigenmaps
        - preserving local structure is essential