# Materials and Methods

**Mica Haney, Camrie Johnson**

University of North Texas

## 1 Data

### 1.1 Data Collection

The siRNA sequences were scraped from a variety of sites using Python. The sites include the The MIT/ICBP siRNA Database ("Sirna at MIT: Human sirnas", n.d.) and the NCBI website (National Center for Biotechnology Information, n.d.). The siRNA Activity dataset from Kaggle was also collected, but this did not require a web scraper and was a .csv file download (Toft, 2021). The information in the .csv file was used to collect data from the NCBI website.

The above datasets did not include the target gene DNA/RNA sequences but did include the name of the target gene. Target genes were scraped from the NCBI website (National Center for Biotechnology Information, n.d.). The URLs were collected in two ways. For the data collected from the MIT/ICBP Database the URL to the relevant NCBI page was listed and collected. For the Kaggle siRNA Activity dataset the relevant URL was constructed from the gene name. From the pages that all of these URLs linked to the FASTA page URL for that gene was collected and scraped for the gene’s sequence.

The scraper used to collect the data was a Python Notebook hosted in Google Collaboratory and used the Selenium, BeautifulSoup, and urllib libraries to pull information off of the website pages.

### 1.2 Data Description

Sense siRNA strands are given for all datasets. Antisense strands, which are complementary to the sense strands, are necessary in order to evaluate the efficacy of an siRNA sequence. Antisense structure plays a role in RISC loading (RNA induced silencing complex). All antisense sequences not given are generated by finding the reverse complement of the sense strands.

#### 1.2.1 MIT/ICBP siRNA Database

The MIT/ICBP siRNA Database contains four relevant pieces of information: the name of the target gene, the URL of the target gene (used for data collection purposes only), the sense DNA sequence, and the antisense DNA sequence. There were no indices indicating where in the target gene the sequences come from or target. How this missing information was handled and the conversion of the DNA to RNA sequences is described in (Section 1.3.2). ("Sirna at MIT: Human sirnas", n.d.)

#### 1.2.2 Kaggle: siRNA Activity

The siRNA Activity dataset contains 24 features, all of which were collected as the dataset was downloaded as a .csv file. Of these features only 4 were used: the name of the target gene, the start and end positions within the target gene, and the siRNA sequence. The additional features were considered but ultimately discarded as the data collected from other sources do not contain this information. (Toft, 2021)

#### 1.2.3 NCBI Website

The NCBI website contains only one additional piece of information that was collected, the target gene sequence. These sequences were mapped with the data collected from the other sources to complete the dataset.

### 1.3 Data Preprocessing

#### 1.3.1 Kaggle Feature Selection

Highly effective siRNA sequences have been shown to satisfy the following criteria: G/C at the 5’ end of the sense strand, A/U at the 5’ end of the antisense strand, AU richness in a 7 base pair long region at the 5’ end of the antisense strand, and no long GC stretches exceeding 9 base pairs in length. (Ui-Tei et. al., 2004). Compositional features are outlined in columns 3-5 within the Kaggle data.

The structure of the target region largely affects the potency of siRNAs. A heavily structured site may not be accessible to potential siRNAs. In addition, the binding energy at the 5’ and 3’ ends must be considered when choosing the best siRNA sequence. This parameter is defined by the change in Gibbs Free Energy (∆G) within the Kaggle data. A relatively low energy (∆G) at the the 5’ end of the antisense strand is preferred as this facilitates RISC loading (RNA induced silencing complex). In contrast, a strand with higher ∆G at the 3’ end compared to 5’ end in the antisense strand results in a less effective siRNA. (Ghosal et. al., 2012; Shabalina et. al., 2006). Energy differences in 5’ and 3’ end are found with column 16 of the Kaggle data set.

#### 1.3.2 Kaggle Sense and Antisense Processing

The siRNA strands in the Kaggle data are unlabeled sense and antisense sequences. Whether a sequence is sense or antisense can be determined by the first nucleotide. In order to generate the missing sequence each record was duplicated and the sequence was replaced with its complement.

#### 1.3.3 MIT/ICBP siRNA Database

The sense and antisense sequences given in the database are of the DNA that the siRNA is targeting. In order to generate the required target siRNA sequences, each sequence has all T values replaced with U values and the labels swapped so that sense becomes anti-sense and vice versa. This is done because the mRNA complement of the sense DNA sequence is the antisense siRNA sequence.

#### 1.3.4 Dataset Versions

The dataset was split into four versions, two of which work as a pair. The first dataset contains the target gene, sense sequence, and antisense sequence from all data sources.

The second dataset is a copy of the Kaggle dataset where only sequences calculated to be effective with an activity score above 70% are kept and the rest discarded. This dataset was included in order to evaluate the models’ ability to generate siRNA sequences that are biologically effective.

The third dataset contains only information from the Kaggle dataset with the extra features described in (Section 1.3.1) included. All numerical values were normalized to a range of 0 to 1. The fourth dataset is a copy of the second with the extra features removed to evaluate the importance of the inclusion of the extra features.

A hold-out test set was collected from the first dataset. All five of the datasets were cleaned of these records so that they did not appear in any of the datasets. The purpose of this dataset is described in (Section 3.2).

## 2 Model Architectures

The models trained to generate siRNA sequences can be classified as either shallow learning machine learning (ML) models or deep learning models. The deep learning models contain computational units that work in concert and in layers to produce an output, while the shallow learning models do not have layers and in many cases don’t contain separate units for computations. An additional difference is that deep learning models typically require significantly more training data in order to make successful predictions. Both model types were included in this project to explore if one type of learning works better for the task and data.

### 2.1 Shallow Learning

#### 2.1.1 Naive Bayes

Naive Bayes are classification models used to predict things based on prior probabilities and are used in natural language processing (NLP) tasks. They can be used to generate output but often struggle if the output space is too large without enough data to calculate accurate prior probabilities. This model type was included due to the low output space given the size of the dataset and the model type’s known ability to work with sequences of information. (Gandhi, 2018a; Gupta, 2023)

#### 2.1.2 Markov Chain

Markov Chains are ML models designed to generate a sequence and are based on the concept of state machines ("What is a state machine?", n.d.). Markov Chains build tables of prior probabilities and generate sequences based on the tables, and initial input, and at later positions previously generated output. This model type was included for its known generative ability, particularly in NLP tasks. ("Markov chain", 2021; "Markov chains. Brilliant Math & Science", n.d.; Pernicano, 2021)

### 2.2 Deep Learning

#### 2.2.1 Long Short-Term Memory (LSTM)

LSTM networks are a kind of deep ML model designed to predict or generate output based on a sequence of input. This model type was specifically designed to handle the vanishing gradient problem, which is defined as the problem where inputs at the beginning of a sequence have a vanishing impact on the output as the sequence gets longer. This model was included as a general deep learning generative model to compare deep learning methods to shallow learning. (Dolphin, 2021)

#### 2.2.2 Transformer

The Transformer is a deep learning model primarily used in NLP tasks and is designed to efficiently work with long sequences of text and handle long-range dependencies. This model type was included because it is very good at working with sequences of text and is included in most of the currently cutting-edge NLP models. (Vaswani et. al., 2023)

## 3 Training Details

### 3.1 Setup and Implementation

The Naive Bayes model was implemented using the Scikit-Learn Python library ("Machine Learning in Python", n.d). The Markov Chain was written as a custom implementation using ("Markov chains. Brilliant Math & Science", n.d.) and (Pernicano, 2021) as references. The LSTM and Transformer models were constructed using the Tensorflow Python library ("Tensorflow", n.d). All data was stored in a Google Drive folder, and all code was run on a Google Colab Notebook.

At this time there are no details on the number of epochs, batches, batch sizes, hyperparameter values, training time, or any other details that must be tuned at the time of training.

### 3.2 Evaluation

All of the models were evaluated using BLEU (Doshi, 2021) and ROUGE ("Candidate. Evaluate translation or Summarization with ROUGE similarity score", n.d) metrics as well as precision, recall, F1, and accuracy averages. The precision, recall, F1, and accuracy scores were calculated individually for each prediction and the minimum, maximum, average, and standard deviation were reported.

All of the models were evaluated on a master held out test set in order to compare them on the same task. All of the models were also evaluated on test sets unique to the experiment.

## 4 References

AG, itemis. (n.d.). What is a state machine? <https://www.itemis.com/en/products/itemis-create/documentation/user-guide/overview_what_are_state_machines#:~:text=A%20state%20machine%20is%20a,state%20transitions%20and%20produces%20outputs>.

*Candidate*. Evaluate translation or summarization with ROUGE similarity score - MATLAB. (n.d.). <https://www.mathworks.com/help/textanalytics/ref/rougeevaluationscore.html>

Dolphin, R. (2021, December 12). *LSTM networks: A detailed explanation*. Medium. <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9>

Doshi, K. (2021, May 11). *Foundations of NLP explained - bleu score and wer metrics*. Medium. <https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b>

Gandhi, R. (2018a, May 17). *Naive Bayes classifier*. Medium. <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c>

Gandhi, R. (2018, July 5). *Support Vector Machine - introduction to machine learning algorithms*. Medium. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

GeeksforGeeks. (2021, December 3). *Markov chain*. GeeksforGeeks. <https://www.geeksforgeeks.org/markov-chain/>

Ghosal, S., Das, S., & Chakrabarti, J. (2012). Computational Approaches for Designing Efficient and Specific siRNAs. In *Bioinformatics*. InTech. <https://www.intechopen.com/chapters/41244>

Gupta, M. (2023, June 13). *Naive Bayes as a generative model*. Medium. <https://medium.com/data-science-in-your-pocket/naive-bayes-as-a-generative-model-7fcc28787188>

*Machine Learning in Python*. scikit. (n.d.). <https://scikit-learn.org/stable/index.html>

*Markov chains*. Brilliant Math & Science Wiki. (n.d.). <https://brilliant.org/wiki/markov-chains/>

Pernicano, G. (2021, April 22). *Text generation with markov chains: An introduction to using Markovify*. Medium. <https://towardsdatascience.com/text-generation-with-markov-chains-an-introduction-to-using-markovify-742e6680dc33>

Shabalina, S. A., Spiridonov, A. N., & Ogurtsov, A. Y. (2006). Computational models with thermodynamic and composition features improve siRNA design. *BMC bioinformatics*, *7*, 65. <https://doi.org/10.1186/1471-2105-7-65>

Sirna at MIT: Human sirnas. (n.d.-a). <http://web.mit.edu/sirna/index.html>

*Tensorflow*. TensorFlow. (n.d.). <https://www.tensorflow.org/>

Toft, L. (2021, July 21). *Sirna activity*. Kaggle. <https://www.kaggle.com/datasets/livtoft/sirna-activity>

Ui-Tei, K., Naito, Y., Takahashi, F., Haraguchi, T., Ohki-Hamazaki, H., Juni, A., Ueda, R., & Saigo, K. (2004). Guidelines for the selection of highly effective siRNA sequences for mammalian and chick RNA interference. *Nucleic acids research*, *32*(3), 936–948. <https://doi.org/10.1093/nar/gkh247>

U.S. National Library of Medicine. (n.d.). National Center for Biotechnology Information. <https://www.ncbi.nlm.nih.gov>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023, August 2). *Attention is all you need*. arXiv.org. https://arxiv.org/abs/1706.03762 <https://arxiv.org/abs/1706.03762>