RNN-based Methods for Identifying Adverse Drug-Drug Interactions using Electronic Health Records Data

By: Morgan Sun

Supervisor: Michael Guerzhoy

April 2022

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Algorithms to identify adverse drug-drug interactions (DDIs) from data relying on experimentation or experts are of value to medical practitioners and researchers. Most existing research on such algorithms focuses on supervised training techniques trained using labelled DDI data. However, generating labelled DDI data requires novel medical research, making scaling labelled DDI datasets challenging. On the other hand, electronic health record (EHR) data is comparatively easy to collect at scale. This thesis develops and evaluates models for identifying DDIs that can be trained exclusively on EHR data, without the need for labelled DDI data. These models are based on RETAIN, an interpretable LSTM architecture for medical sequence classification. The behavior of RETAIN when given specially generated synthetic patient data is used to quantify DDI likelihood for a given pair of drugs; sampling-based and attention-LSTM-based methods of generating synthetic patient data are evaluated. The models are trained on the MIMIC-III intensive care dataset and evaluated using AP score on the BIOSNAP Chemical-Chemical DDI dataset. The trained models are capable of achieving substantially better-than-random performance and are capable of identifying DDIs with precision=0.8 and recall=0.2. Additionally, the models are able to continue achieving better-than-random performance when evaluated on drug interactions which were not explicitly represented in the MIMIC-III training data. Overall, evaluation results demonstrate that this framework for identifying DDIs has the potential to offer unique clinical value in a highly scalable manner.

Acknowledgements

I'd like to thank my supervisor, Professor Michael Guerzhoy, for his guidance, advice, and accomodation throughout this research project.

I'd also like to thank my friends Nou and Jeff, as well as my parents, for their invaluable support during this challenging year.

Contents

List of Figures

List of Tables

1 Introduction

The adoption of Electronic Health Records (EHR) systems in hospitals has enabled the collection of relatively large datasets of patient health records, such as the MIMIC-III dataset [1]–[3]. In the past, conventional machine learning methods such as the logistic-regression-based APACHE model [4] have been used to analyze such datasets. Modern machine learning has introduced neural networks, computational models capable of learning nonlinear and high-dimensional functions given large quantities of data. The combination of the availability of EHR data and the maturity of modern machine learning techniques presents an opportunity to develop systems whose aim to extract general clinical information. More specifically, there is an opportunity to develop systems capable of identifying drug-drug-interactions (DDIs).

Although data-driven methods of identifying potential DDIs would not provide direct replacements for human trials in investigating DDIs, such methods would nonetheless present substantial clinical value, as they are purely computational methods that do not rely on any form of laboratory or human experimentation. This allows these methods to offer scalability and convenience compared to conventional drug interaction experiments, which are time-consuming and costly [5]. More specifically, data-driven methods of identifying DDIs can (1) inform the design and prioritization of conventional experimentation, (2) scale and generalize to new contexts or newly adopted treatments, and (3) sidestep logistical and ethical barriers to human experimentation.

Thus far, almost all research on machine learning based methods of identifying DDIs employ supervised machine learning in some form [6]–[9]. More specifically, datasets consisting of pairs of drugs labelled with information regarding interaction severity are used to provide the targets for supervised learning; models are developed to directly optimize for the prediction of these targets. The aim of this research is to explore alternative approaches that do not require such labelled datasets in the model training process. In this research, models will be trained using only patient health records (i.e. EHR data from the MIMIC-III dataset); labelled DDI data will be used only for model evaluation purposes.

This constraint increases the challenge posed by the DDI identification problem, since a straightforward supervised learning approach is no longer possible. However, developing models that comply with this constraint has unique benefits. The scale of available training data has been a strong factor in dictating the success of modern machine learning models in other domains such as image recognition. Therefore, when evaluating machine learning approaches, the scale of training data available both today and in the future is an important consideration. Current datasets of known DDIs are derived from drug-specific research and expert knowledge. As such, these datasets are limited in scale, and increasing the scale of these datasets requires performing costly novel medical research. The aforementioned existing research on DDI identification models employs these types of datasets as training data, which poses a potential threat to the future applicability of these models. On the other hand, a DDI identification model which can be trained using only routinely-collected patient records would benefit from continuously-increasing availability of training data as data accumulates over time.

The models which will be developed and discussed in this thesis provide another benefit over existing DDI identification models. Many of these existing DDI identification models rely on features derived from the chemical or physiological properties of drugs [6], [8], [9]. These features cannot be computed for certain classes of drugs (e.g. antibody-based drugs), as well as for non-drug clinical interventions, and therefore these

models can only be used to identify adverse interactions between pairs of drugs for which these features can be computed. The models discussed in this thesis, on the other hand, do not rely on such features, and therefore could potentially be extended to identify interactions involving non-pharmaceutical medical interventions such as surgeries. Although this application will not be extensively explored in this thesis, it is nonetheless worth noting as another characteristic which differentiates this research from past work.

The DDI identification models developed in this thesis build upon a previous project completed by Chae et. al [10], and are based on the following ideas:

1. RNN neural networks can be trained to forecast and interpret patient treatment behavior and outcomes.
2. Synthetic patient data corresponding to specific drug combinations can be algorithmically generated.
3. RNN neural networks will behave differently when given patients exhibiting presence of a DDI.
4. This difference can be used to derive a quantitative measure of DDI likelihood.

Two methods of algorithmically generating synthetic patient data, as well as two methods of quantifying difference in RNN behaviour, will be developed and evaluated, for a total of $2 \times 2 = 4$ models. Given a pair of drugs, a DDI identification model will first generate a set of synthetic patients, then analyze the synthetic patients using a RNN, then finally, based on the RNN's behavior, compute a metric representing the likelihood of the given pair of drugs resulting in an adverse DDI.

This thesis document will first discuss existing literature on existing DDI identification models and RNN medical modelling. A description of the dataset sources and preprocessing will then be given, followed by a discussion of two RNN models which were selected and trained for use as components in the DDI identification models developed in this work. The architecture of the four candidate DDI identification models will then be presented. Finally, evaluation methods and results will be presented and discussed.

2 Literature Review

This section will be consist of three subsections. The first will provide a general overview of data relevant to this research. The second will discuss existing work on machine learning approaches to DDI identification. The third will look at trends in machine learning based medical modelling, with a focus on RNN-based techniques.

2.1 Datasets and Preparation

This subsection will provide an general overview of the structure of EHR and DDI datasets, as well as a brief overview of preparation steps that are typically necessary for these datasets.

2.1.1 EHR Data and Preparation

The adoption of EHR systems has become increasingly commonplace, with 84% of hospitals and 86% of office-based physicians adopting some form of EHR by 2017 [11], [12]. This has led to many datasets of patient medical history; these datasets consist of static features such as patient demographic information, time series of quantitative data such as vitals, and sequences of non-quantitative events such as diagnosis/intervention

codes and physician notes [13]. Commonly used features for medical models can be broadly categorized into several types, which are summarized in Table 1.

Feature	**Type**	**Representation**
Demographics	Static	Quantitative and Categorical
Medication	Discrete Event	Categorical (GPI codes)
Diagnoses	Discrete Event	Categorical (ICD-9/10 codes)
Procedures	Discrete Event	Categorical (CPT codes)
Lab results	Discrete Event	Quantitative
Physician notes	Discrete Event	Text
Vitals	Continuous	Quantitative

Table 1: Types of features commonly used for medical modelling.

In EHR datasets, sequential event data can often be grouped into discrete admissions, as hospital admissions are often short-term events. A common data-cleaning practice is therefore to filter the dataset based on number of admissions. For short-term prediction tasks, authors may retain only the first admission per patient, such done by Suresh et al. [14]. Conversely, for longer-term prediction tasks, only patients who have at least some minimum number of admissions (typically 2) are considered [15], [16]. Individual ICU stays may also be subject to length constraints. For example, Kaji et al. [17] removed insufficiently long (under 2 days) ICU stays, and truncated stays longer than a maximum (14 days). Suresh et al. similarly considered only stays lasting 12-240 hours.

As with most real-world datasets, EHR datasets, including MIMIC-III, suffer from missing and erroneous values. Pham et al. [15] remove all patients with missing categorical demographic information. Models using quantitative features depend more heavily on imputation of missing values. Kaji et al. choose to entirely exclude features with $\leq$75% missing values, and imputes remaining missing values using variable medians. Kaji et al. furthermore note the presence of incorrectly recorded values in MIMIC-III, and corrects for this by replacing values above a 95% threshold with the variable median. Lipton et al. [18] impute missing values via forward- and back-filling, while noting that back-filling would be inappropriate for forecasting tasks. Clinically normal values are used in cases where a variable was not measured for a patient, since physicians are less likely to order lab measurements for variables that they believe to be within a normal range.

Another common data-cleaning step involves dealing with extremely rare clinical events with insufficient associated data. For example, Farhan et al. [19] excludes all clinical events not present in at least 1% of all sequences.

2.1.2 DDI Data and Preparation

Generally, DDI datasets can be considered to represent a graph, where each node is a drug, and each edge represents an interaction between a pair of drugs. Typically this data is provided as a list of edges.

Different datasets may differ greatly in the amount of information pertaining to interactions (edges). For example, the Stanford BIOSNAP Chemical-Chemical dataset [20] provides only the edge list, while the ONH interaction datasets [21] sorts interactions into two classes of severity. The DrugBank database [22] provides the highest level of edge information, classifying DDIs into 177 categories of interactions.

DDI datasets are often used in conjunction with other datasets which provide more robust features pertaining to individual drugs (nodes). The DrugBank database is one of the most popular, providing information regarding drug structure, target genes, and target gene functionalities. As different databases employ different standards for labelling drugs, additional mapping/translation steps using supplementary "thesaurus" databases may be necessary when using multiple databases together.

The BIOSNAP and DDI datasets are derived by aggregating research literature, while the ONH dataset is derived from a panel of experts.

2.2 DDI identification using Machine Learning

The majority of existing literature on machine learning models for DDI identification and classification frame the problem as a supervised machine learning problem where the input is some representation of a pair of drugs; however, there is some variation in both the representation of the input used as well as the desired output.

Lee et al. [6], Kastrin et al. [8], and Cheng et al. [9] all employ a similar approach of using similarity metrics between a pair of drugs to represent said pair. These similarity metrics are derived by first representing characteristics of each drug as a categorical multi-hot vector, then calculating the similarity between the two vectors using measures such as cosine similarity or Tanimoto coefficient [23]. Most authors compute multiple similarity metrics for multiple types of similarity. More specifically, Lee et al. employ measures of simlarity in structure, targeted genes, and functionality of targeted genes; Kastrin et al. uses measures of similarity in structure, drug category, therapeutic effects, and adverse effects; Cheng et al. uses measures of similarity in structure, drug category, therapeutic effects, and targeted genes. Liu et al. [7] employs a substantially different approach to feature construction, computing the correlations between types of adverse events reported to the FDA and presence of associated drugs.

Kastrin et al., Cheng et al., and Liu et al. frame the DDI identification problem as a binary classification problem, targeting the prediction of presence/absence of a severe drug interaction. These authors use a range of classical machine learning models, with a particular focus on support vector machines [24]. Lee et al. instead targets prediction of the category of drug interaction using a more complex feedforward neural network.

All four aforementioned DDI identification models source their known DDI drug pairs (at least partially) from DrugBank [22]. Because Kastrin et al., Cheng et al., and Liu et al. aimed to develop binary classifiers, they additionally required pairs of drugs without severe interactions to serve as negative examples for training and evaluation of their models. Liu et al. derived these non-interacting pairs from the ONH dataset [21], which was created based on expert opinion. Kastrin et al. and Cheng et al. derive non-interacting pairs through simple randomly sampling.

2.3 RNN-based Medical Modelling

RNNs have been applied extensively to modelling and forecasting of a variety of medical and clinical time series. This subsection will first provide an explanation of the mathematics behind the most popular RNN model variants. It will then endeavor to identify some key challenges/decisions involved in medical modelling

using RNNs, as well as summarize the range of approaches various authors have taken to address these challenges/decisions.

This thesis will employ two RNNs with the respective purposes of (1) forecasting medication prescriptions and (2) predicting and interpreting patient mortality. However, the works presented in this review include long- and short-term prediction of a variety of medical events, and are not limited to modelling of patient mortality and medications. Nonetheless, there will be several relevant themes in the approaches taken by these works.

2.3.1 RNN Preliminaries

Each layer of a basic perceptron neural network can be represented as a pair of parameters: a weight matrix W and a bias vector b. The output of the layer is

$$\hat{y} = f(Wx + b)$$

where f is some nonlinear activation function. The parameters can be optimized to minimize some loss function defined on $(\hat{y}, y)$ via gradient descent. This structure allows perceptron neural networks to handle one-to-one regression problems, but is not ideal for problems involving sequences as regression inputs and/or targets, as it does not attempt to model temporal relationships between elements of a sequence.

RNNs were introduced in 1985 by Rumelhart et al. [25], and were applied to a supervised machine learning task in 1990 by Elman [26]. The RNN structure is designed to address regression problems involving sequence inputs $\{x_t\}_{t=1}^n$ and targets $\{y_t\}_{t=1}^n$. In the simplest form, a recurrent layer consists of three trainable weight matrices W_h, U_h, and W_y and two trainable bias vectors b_h and b_y. The layer furthermore maintains a state vector h_t which is updated for every timestep $t \in \{1, 2, 3, ..., n\}$. For each timestep, the layer performs the operations

$$h_t = f(W_h x_t + U_h h_{t-1} + b_h)$$
$$\hat{y}_t = g(W_y h_t + b_y)$$

where f and g are some nonlinear activation functions. For sequence-to-sequence tasks, all of the $\hat{y}$ values can be taken as the predicted output sequence; for sequence classification or regression tasks, the $\hat{y}$ corresponding to the last timestep is typically used. Like perceptrons, RNNs are trained through gradient descent, however, simple RNNs like the one described above struggle to model relationships between timesteps not in close proximity to each other. This is because backpropagated gradients tend to exponentially decay or grow in magnitude over the course of multiple timesteps [27]. The "vanishing/exploding gradients" problem has driven several key structural innovations in recurrent models, including gated recurrent unit (GRU) and long short-term memory (LSTM) models.

GRU models consist of trainable parameters W_r, U_r, b_r, W_z, U_z, b_z, W_h, U_h, b_h, W_y, and b_y. For each timestep, a GRU layer first computes the "reset gate" r_t and "update gate" z_t:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

Note that the sigmoid function σ clamps the elements of both r_t and z_t to $(0, 1)$. The reset gate r_t is used to attenuate the values in the hidden state through elementwise multiplication (denoted $\odot$) before a new candidate hidden state $\hat{h}_t$ is computed. The elements of the update gate z_t are then used to control how much to update the hidden state with $\hat{h}_t$:

$$\hat{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_t)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

The computation of $\hat{y}_t$ is identical to that used in simple RNNs. The structural changes introduced by GRUs make it easier for the model to propagate a hidden state through multiple timesteps with minimal changes, thus mitigating the problem of vanishing/exploding gradients.

LSTMs are conceptually similar to GRUs, but employ three gates: "forget", "input", and "output", as opposed to the two used by GRUs. They also introduce a "memory state" c_t which, like h_t, is passed through timesteps. Their operations can be described as follows; as with GRUs, W, U, and b represent trainable parameters:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \tanh(c_t)$$

LSTM models have achieved remarkable success in a diverse range of sequence tasks, such translation [28], speech recognition [29], and image caption generation [30]. This success has established them as the preferred variety of RNN in many applications.

2.3.2 Feature Engineering and Embedding

It is often desirable to reduce the dimensionality of input vectors. This is motivated by the extreme granularity of many medical coding schemes, which results in large categorical input vectors which can easily lead to overfitting. For instance, at the most granular level ICD-9 defines a total of 17849 distinct medical codes for diagnoses, while the more recent ICD-10 raises this by nearly an order of magnitude to 141747 codes [31]. A similar problem exists for other types of medical events, such as medications and procedures. To reduce the number of target classes, Choi, Bahadori, Schuetz, et al. [16] cluster ICD-9 codes by the first three characters, and GPI drug codes by GPI Drug Group. In a later work, Choi, Bahadori, Kulas, et al. [32] employ the same method for clustering GPI drug codes, but switch to using Clinical Classifications Software [33], [34] for clustering ICD-9 diagnosis and CPT procedure codes; they furthermore apply this clustering to not only the target, but also the input data. Pham et al. [15] similarly cluster ICD-10 codes by the first two characters. These less granular classes are still of value to physicians, and vastly reduce the number of classes to the order of hundreds.

Clustering of medical codes is not the only method of input dimensionality reduction. Choi, Schuetz, et al.

[35] compared medical code clustering to another approach in which they used skip-grams [36], a technique commonly used in NLP to generate embeddings for word/character tokens, to generate embeddings for unclustered medical codes. They found that this approach yielded both better model performance and lower input dimensionality.

Although the use of skip-grams to generate embeddings for categorical inputs is a novel approach in this domain, many authors developing medical RNN models use some form of embedding for categorical inputs. This is most often done using an embedding layer $x_i^{emb} = Ax_i^{cat}$, where A is a trainable embedding matrix and x_i^{cat} is a categorical input vector; This method has been applied to a range of model applications and architectures [15], [32], [37]–[39]. Additionally, Choi, Bahadori, Schuetz, et al. investigated the use of skip-gram embeddings to initialize the embedding matrix, and reported improved performance over random initialization.

Due to the extensive leveraging of embedding mechanisms in this domain, it is often desirable to convert quantitative variables to a categorical representation. Esteban, Staeck, et al. [37] performed this conversion by quantizing numerical laboratory results into three categorical bins (within, above, or below ± 1 standard deviation from the mean). This also removed the need to perform imputation of missing quantitative values, as missing values can simply be represented by zeroing out all three categorical features; the authors found that this approach actually increased model performance over directly providing normalized numerical features.

Other data processing and feature construction steps which are of less relevance to our research include application of NLP to generate embeddings of physician notes [14], as well as manual selection of features with known relevance to prediction targets [14], [17]. These steps are useful for training models exclusively for forecasting of specific medical events, but restrict the ability of the model to generalize to applications involving wide ranges of medical events of interest.

2.3.3 Handling Static Features

As RNNs are designed to deal with sequential inputs, incorporating static information such as demographic features can require some modification of either the data representation or the model architecture. Suresh et al. [14] take the simple approach of concatenating static features onto each sequential input vector, allowing an unmodified RNN model to be used. Esteban, Staeck, et al. [37] chose to augment their RNN architecture instead. Sequential features are input to an RNN, while static features are input to an independent multi-layer perceptron (MLP). The hidden states from these two models are concatenated prior to a dense layer which outputs the final prediction.

2.3.4 Handling Variable Time Intervals Between Events

The highly variable time intervals between discrete medical events such as diagnoses, medication administration, and procedures pose a significant challenge for the development of RNN medical models [13]. The solutions to this challenge can be divided into three categories:

1. Grouping and pooling of medical events by fixed-size quanta of time
2. Grouping and pooling of medical events by admission
3. Representing each individual medical event by one input vector (i.e. no grouping/pooling).

Grouping and pooling of medical events by fixed-size quanta of time eliminates the problem of RNN inputs being unevenly spaced in time, but introduces the problem of designing a method to perform said pooling. Kaji et al. [17] target daily prediction of sepsis, myocardial infarction, and vancomycin administration. As they primarily use quantitative measurements, pooling of multiple measurements per day is done by using the minimum, maximum, average, and standard deviation of each measurement per day. Suresh et al. [14] target forecasting of ICU interventions with a short-term (6-10 hours) prediction horizon, and quantize time by the hour. Multiple quantitative values for a quantitative feature within the same hour are averaged, while multiple incidences of discrete medical events within the same hour are dealt with by constructing a categorical feature for each event corresponding to presence/absence of one or more incidences of said event within the hour. Rajkomar et al. [38] seek to predict longer-term prediction targets such as mortality, diagnosis, and length-of-stay (LOS) by using a patient's entire medical history as input. They quantize time into 12-hour segments, and take a significantly different approach to pooling of multiple discrete medical events: every event corresponds to an embedding vector, and a weighted average of the embeddings for each event occurring within a 12-hour period is used to represent said period; both embeddings and weights are optimized during model training.

Grouping and pooling of medical events by admission, as opposed to using fixed-size timesteps, is a more common approach, especially for longer-term forecasting where a patient may have a history of many admissions. Almost all such approaches involve using a trainable embedding layer for individual events, then summing all embedding vectors for events occurring during a admissions to construct a representation of that admissions [16], [32], [37], [39]. Grouping by ICU admissions introduces the problem of variable durations between consecutive admissions. Esteban, Staeck, et al. [37] choose to simply ignore this issue, and are able to achieve reasonably competitive results in spite of it. Choi, Bahadori, Schuetz, et al. [16] and Rasmy et al. [39] address this issue by appending the log-time since the previous admissions to each input vector after embedding. In a later work, Choi, Bahadori, Kulas, et al. [32] switch to the slightly different approach of using the log-time since the patient's first admissions. Pham et al. [15] take a radically different approach to both pooling of events and representation of time intervals. Rather than summing embeddings for events during a admissions, they evaluate three alternative techniques: element-wise maximum, normalized vector sum, and vector mean. Instead of concatenating time since last admissions with the pooled event vectors, it is used as a separate input which modulates only the forget gate of a modified LSTM model.

Representation of each individual medical event as its own input vector is a significantly less commonly-used approach, although Choi, Schuetz, et al. [35] take this approach. As is commonly done with the vector-per-admissions approach, they first use an embedding layer before concatenating the log-time since the last event. The viability of this approach depends on knowing the sequence of medical events within each admissions.

2.3.5 Model Architectures

Although there are many examples of successful direct application of "vanilla" RNN models, many authors implement changes to improve predictive performance, account for domain-specific dynamics, or augment model capabilities. For concision, a selection of the most significant changes and the reasoning behind them will be summarized in Table 2.

Publication	Modification	Authors' Justification
Choi, Bahadori, Schuetz, et al. [16]	Two output heads, one for predicting what the next medical event will be, the other for predicting when it happens.	Enables the model to predict duration until next medical event.
Choi, Bahadori, Kulas, et al. [32]	Two LSTMs used to generate admission- and event-level attention weights. Attention weights used to pool all sequential inputs into a context vector. Dense layer uses context vector to make final prediction.	Allows for admission- and event-level importance interpretability.
Esteban, Staeck, et al. [37]	Two parallel networks - A MLP for static features, and a RNN for sequential features.	Novel method of combining both static and sequential features.
Rajkomar et al. [38]	Ensembling of a "vanilla" LSTM with two other non-recurrent networks.	Improved AUROC.
Pham et al. [15]	Monotonic decay function applied to LSTM forget gate.	Less recent events have less clinical relevance for short-term predictions.
	Time since last admission is used to modulate LSTM forget gate through a learned weight matrix.	Enables learning of complex disease progression dynamics.
	Diagnoses and treatments used as separate LSTM inputs and modulate different gates.	Helps separate confounding interactions between diagnoses and treatments.
	LSTM input gate scaled by a factor depending on whether a admission is planned or unplanned.	Unplanned admissions are typically more severe than planned admissions.
	All historical LSTM hidden states pooled and used to make extreme-long-term predictions.	Global dynamics across multiple admissions helpful for extreme-long-term predictions.
Kaji et al. [17]	Generation of feature-level attention weights.	Allows for admission- and feature-level importance interpretability.
Rafiei et al. [40]	Stacking of 1-D convolutional layers after LSTM layers.	Convolutional layers are effective and noise-resistant feature extractors.

Table 2: Summary of modifications to RNN architecture for medical modelling.

2.3.6 Training Methodologies

Many medical RNN models employ well-established techniques to mitigate overfitting. For instance, L2 regularization [41] is a technique in which a penalty term $||W||_2^2$ is added to the loss function to discourage excessively large values in a weight matrix W; this is a versatile technique which can be applied to a variety of machine learning models with little modification. L2 regularization is employed by Suresh et al. [14], Rajkomar et al. [38], and Rafiei et al. [40].

Dropout [42] is a technique in which neurons within a neural network are randomly masked (i.e. their outputs are zeroed out) during training; this improves model robustness to overfitting by essentially training an ensemble of random sub-networks. Dropout was initially applied to feedforward networks [43], but has been adapted for use in RNNs in a variety of ways. The most straightforward method of doing so is to apply dropout only to the inputs and outputs of a recurrent layer, but not to recurrent connections [44]. This is the approach taken by Suresh et al. [14] and Rafiei et al. [40]. Rajkomar et al. [38] also apply this type of recurrent dropout, but combine it with two others: "variational dropout" [45] and "zoneout" [46], both of which are methods of generalizing the idea of dropout to recurrent connections. They additionally apply

standard feedforward dropout to the output of their embedding layer. Newer forms of recurrent dropout include the AWD-LSTM proposed by Merity et al. [47], which has not been applied to medical modelling yet, but has seen great success in natural language modelling.

2.3.7 Interpretability

Interpretability is an important characteristic for many model applications, especially in medicine, where physicians want to know what factors induced a model to provide a particular recommendation or prediction. Classical regression models such as APACHE [4] are relatively easy to interpret, but more modern machine learning models involve transforming data into embeddings spaces which are not easily interpretable, leading them to be seen as black-box models. Interpretation of RNNs is made even more challenging due to the the propagation of the hidden state through multiple timesteps.

Suresh et al. [14] apply feature occlusion, a method previously used to interpret CNN models [48], to quantify the relative importance of each input feature. The results are consistent with clinical knowledge; however, their approach does not provide insight into the relative importance of individual timesteps. Kaji et al. [17] employ an attention mechanism to weight input features, and are thus able to quantify timestep-level and feature-level importance in a sequence-to-sequence task.

Choi, Bahadori, Kulas, et al. [32] introduce a novel model architecture called RETAIN with the goal of predicting patient heart failure based on patient history in an interpretable manner. The RETAIN architecture is one of the two RNNs which will be used as components for the DDI identification models developed in this thesis, therefore a more comprehensive discussion of RETAIN will be presented in a subsequent section.

2.3.8 Models for Next Medical Event Forecasting

The majority of works presented in previous sections target prediction of individual or small sets of clinical events or outcomes. Since the goal of this thesis is to develop methods that generalize to arbitrary medications, such specialized models are less relevant than models capable of more general medical modelling. More specifically, this section will discuss existing work on models targeting prediction of the next medical event out of a large and general hypothesis space of possible events. The methods and results presented here are of interest, because the second of the two RNNs which will be used for our DDI identification models is a RNN which aims to forecast the next medication prescription for a given patient out of a large and general space of possible medications. This is a very similar task to the existing work presented here, and therefore the results discussed here can be used to inform and benchmark our medication forecaster RNN.

Farhan et al. [19] target next-diagnosis prediction on the MIMIC-III dataset. Their approach does not directly train a neural network model to output predictions, but instead leverages cosine similarities between embedding vectors. They use a variant of skip-grams to train embeddings for all medical events, and generate representations of patients by using a time-weighted sum medical event embeddings. They then investigate two approaches for modelling the likelihood that a given patient will receive some diagnosis upon their next admission, one based on collaborative filtering and one based on patient-diagnosis projection similarity. These approaches are able to outperform a logistical regression baseline.

Esteban, Schmidt, et al. [49] similarly seek to predict which medical events will occur during a patient's

next admission, however, they predict medical events in general, as opposed to only diagnoses. They use a dataset of patients who are awaiting or have received a kidney transplant. They do not use an RNN, but instead use a MLP which ingests embeddings of the last k admissions in addition to static patient information. The authors notably observe that, due to the extreme sparseness of most medical events, even a constant-predictions model is capable of achieving an AUROC of 0.964; the authors therefore suggest that AUPRC be used to evaluate next-medical-event models instead; their model achieves an AUPRC of 0.574. In a later publication, Esteban, Staeck, et al. [37] follow up on this work by adding their modified RNN architecture to this evaluation; the best version of this architecture achieves an AUPRC of 0.571, failing to outperform their earlier MLP model.

Choi, Bahadori, Schuetz, et al. [16] endeavor to predict not only the medical events during a patient's next admission, but also the duration until said admission. Aspects of their work have been discussed in previous sections; in summary, they:

1. Using skip-grams to generate embeddings for medical events,
2. Representing admissions by summing constituent event embeddings,
3. Appending log-time since last admission to these admission representations, and
4. Using a 2-layer GRU with two outputs to predict both the events in and time until the next admission.

The authors note that the model does not perform well in predicting the time until next admission. However, it performs competitively in terms of predicting the medical events during the next admission. More specifically, the authors propose the use of top-k recall as a metric for evaluating next-medical-event models. Top-k recall is defined as the number of true positives in the top k predictions divided by the total number of true positives. They justify this choice by arguing that real-world doctors generally do not make a single diagnosis, but instead identify multiple likely diagnoses. Their model achieves top-10, -20, and -30 recalls of 68.31, 79.77, and 85.53, respectively. This compares favorably to a naive strategy of constantly guessing the most common k diagnoses, which yields top-10, -20, and -30 recalls of 62.99, 69.02, and 70.07, respectively.

3 Datasets and Preprocessing

Two datasets were used in this work. EHR data for model training was sourced from the MIMIC-III dataset [1], [2], a collection of ICU care records collected from a large tertiary-care hospital over the course of about a decade. Known DDI pairs were sourced from the Stanford BIOSNAP Chemical-Chemical dataset [20]. The following subsections will discuss the properties of these datasets and preprocessing steps taken.

3.1 MIMIC-III Dataset Description and Preprocessing

The MIMIC-III dataset consists of a wide range of patient health records. We will only consider records of medication prescriptions, therefore there will be no need for correction or imputation of quantitative data.

The dataset consists of a set of patients $\{X_1, X_2, X_3, ..., X_N\}$ and their respective outcomes (mortality or survival) $\{y_1, y_2, y_3, ..., y_N\}$. Each patient consists of a sequence of admissions $X_n = [x_{n,1}, x_{n,2}, x_{n,3}, ..., x_{n,i}]$; each admission contains a sequence of medication prescriptions $x_{n,k} = [m_{n,k,1}, m_{n,k,2}, m_{n,k,3}, ..., m_{n,k,j}]$. The MIMIC-III dataset contains a total of $N = 46520$ patients. Patients with less than 50 medication

prescriptions across all admissions were excluded, reducing the dataset size to $N = 23828$ patients with an average mortality rate of 27.15%. Patients with more than 100 medications across all admissions were truncated down to the 100 most recent medications. These preprocessing steps are very similar to the measures taken by authors of similar works, as discussed in section 2.1.1. After these steps, patients were randomly divided into training, validation, and test sets by a 80%/10%/10% split.

Medications in MIMIC-III are represented using a variety of standards, including National Drug Code (NDC), General Sequence Number (GSN) and a list of common/chemical name synonyms. Out of these options, NDC codes are the only standard that uses a hierarchical format of any kind. A heirarchical format allows for dimensionality reduction of codes through truncation, as discussed in section 2.3.2. The NDC format was therefore used. There are 4203 unique 11-digit NDC codes in the dataset; the last two digits of each code encode packaging type, which is irrelevant to drug interactions and can thus be truncated; truncation yielded 3916 unique medication types.

As can be seen in Figure 1, the vast majority of prescriptions are concentrated in a small number of common medication types, with half of all prescriptions being for one of only 76 medication types. Therefore, the number of medication types can be further reduced by only considering the most common types, similar to the approach taken by Farhan et al. [19]. Only the 606 most common medication types were used; this subset corresponds to 90% of the prescriptions present in MIMIC-III.

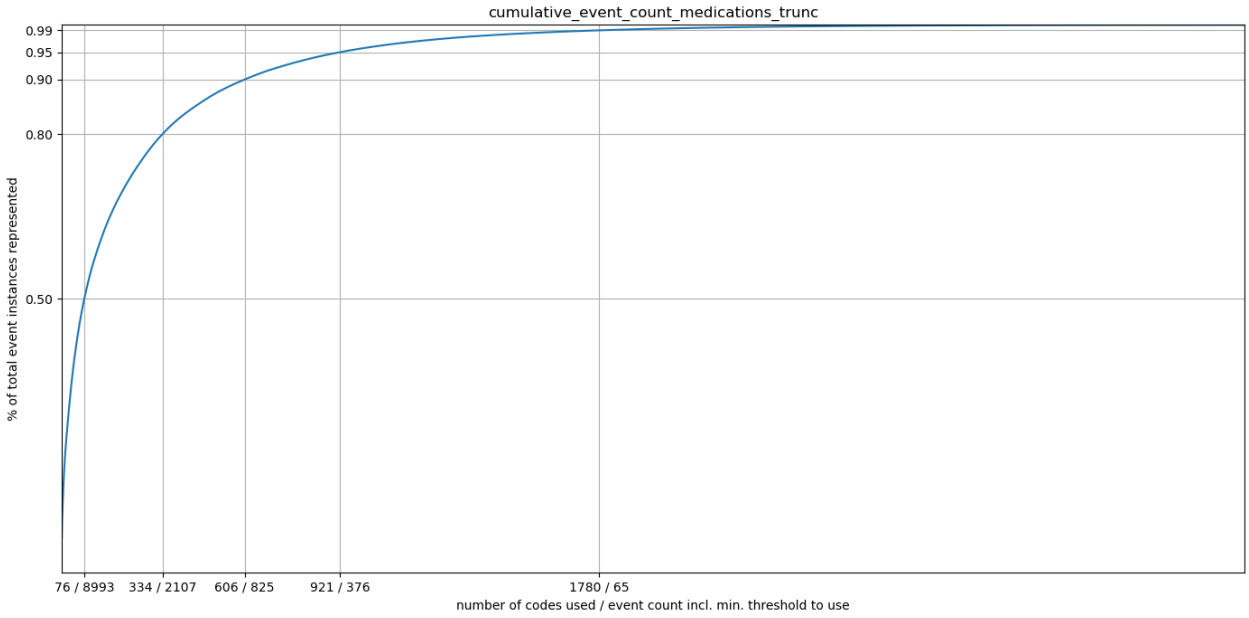


Figure 1: Cumulative % of prescriptions in MIMIC-III as a fcn. of number of medications considered.

3.2 BIOSNAP Dataset Description and Preprocessing

DDI pairs were sourced from the BIOSNAP Chemical-Chemical dataset [20]. As mentioned in section 2.1.2, this dataset consists of a set of pairs of drugs $\{(m_a, m_b), (m_c, m_d), ...\}$ that are known to result in adverse interactions. The dataset represents each drug using its DrugBank ID. These representations must be converted to a NDC codes. DrugBank offers a vocabulary database which maps DrugBank IDs to

corresponding common/chemical names. As MIMIC-III also provides common/chemical names, these names were used to map DrugBank IDs to NDC codes. As most of the the drugs in the BIOSNAP dataset were either relatively rare or not present at all in MIMIC-III, this mapping reduced the number of known DDI pairs from 48514 to 711.

The 711 known DDI pairs form the set of true positive (TP) pairs. A set of true negative (TN) pairs is necessary to complement this set for evaluation; following the methods used by Kastrin et al. [8] and Cheng et al. [9], 711 unique pairs were randomly sampled, with sampling probabilities weighted by the relative frequencies of drugs in MIMIC-III.

4 RNN Architectures, Validation, and Training

Two RNN neural networks were trained on the preprocessed MIMIC-III dataset for use as components in DDI identification models. It is important to reiterate that these RNNs do not directly output predictions of drug interactions. Instead, these neural networks are used as components in the larger DDI identification models. Therefore, usage of the term "model" will not, in the context of this report, refer to these RNNs.

4.1 Forecaster LSTM

The forecaster LSTM is a simple attention-LSTM intended to predict the next medication a given patient will receive. More specifically, the forecaster LSTM takes as input a sequence of $j-1$ medications the patient has been prescribed, represented as a sequence of tokens $[m_1, m_2, m_3, ..., m_{j-1}]$, where each token is a one-hot vector. The LSTM is trained to predict the next token in the sequence m_j, although we are actually more interested in the predicted probability distribution for the next token $\hat{m}_j$.

After training, the forecaster LSTM will be used to generate synthetic patient data in a manner analogous to the methods used to generate novel text in a natural language processing context [50]. More specifically, a sequence of tokens of length $j-1$ will be extended by iteratively by:

1. Using the forecaster LSTM to compute a probability distribution over the next token $\hat{m}_j$
2. Randomly sampling a token m_j from this probability distribution
3. Appending the sampled token to the original sequence

This translates to a structure of one timestep per medication prescription, which, as mentioned in section 2.3.4, is a less commonly used approach as compared to using one timestep per patient admission. However, as MIMIC-III provides the order in which medications were prescribed during a patient's admission, using one timestep per medication allows this ordering information to be retained and used by the forecaster LSTM, while using an approach of one timestep per admission would discard this information. In order to represent the delimitation between consecutive admissions, an additional special token type is used as a separator. Two more special token types are used to represent patient survival (to date) and patient mortality, and one more special token type is used for padding. These four special token types, combined with the tokens types representing the 606 most common medications, give a total of 610 token types.

4.1.1 Forecaster LSTM Architecture

The forecaster LSTM architecture used was developed by Chae et al. [10]. The architecture was updated to allow for arbitrary-length input sequences, but is otherwise unchanged.

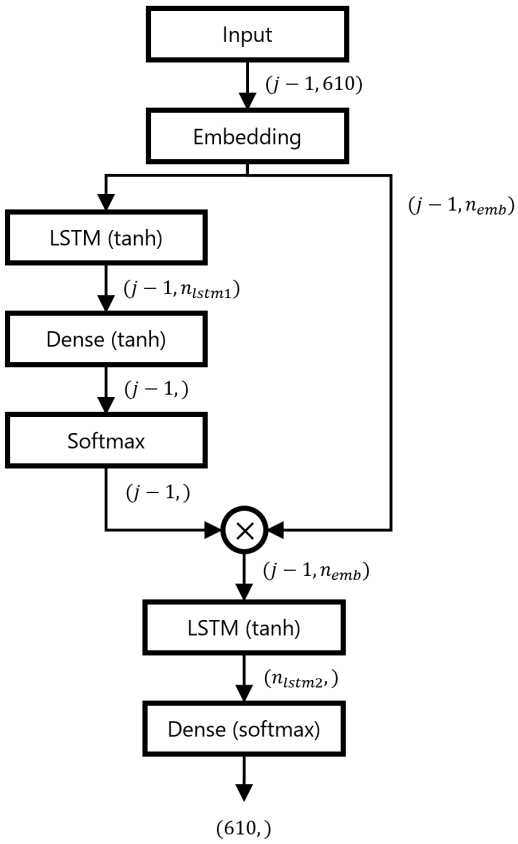


Figure 2: Architecture of next-medical-code forecaster LSTM.

The structure of the forecaster LSTM is shown in Figure 2. The input sequence $[m_1, m_2, m_3, ..., m_{j-1}]$, $m_k \in \{\text{one-hot}\}^{610} \ \forall k$ is first passed through an embedding layer which converts each token to a trained vector embedding representation, resulting in a sequence of embeddings $[u_1, u_2, u_3, ..., u_{j-1}]$, $u_k \in \mathbb{R}^{n_{emb}} \ \forall k$. This sequence is then given to the first of two LSTM blocks, which calculates a scalar attention weight for each timestep $[w_1, w_2, w_3, ..., w_{j-1}]$, $0<w_k<1 \ \forall k$. The attention-weighted embedding sequence $[w_1 \cdot u_1, w_2 \cdot u_2, w_3 \cdot u_3, ..., w_{j-1} \cdot u_{j-1}]$ is then used as the input to the second LSTM block, which outputs a softmax prediction for the next medication $\hat{m}_j \in \mathbb{R}^{610}$, which can be interpreted as a categorical probability distribution. Based on Chae et al.'s previous work, an embedding size of $n_{emb} = 40$ and LSTM block sizes of $n_{lstm1} = n_{lstm2} = 128$ were selected.

4.1.2 Forecaster LSTM Validation Experiments

One major way in which the usage of this forecaster LSTM differs from Chae et al.'s previous work [10] is in the addition of special separator tokens to mark the boundaries between consecutive patient admissions. In order to investigate the effects of this change, as well as to test the forecaster LSTM code before training

on real patient data, a small experiment on toy data was designed.

The toy data consisted of four token types arranged in a simple endlessly-repeating periodic pattern, with each token having a different period. An subsequence of this periodic pattern is shown in Table 3, with $\times$ marking the presence of a token.

		Admission Index																	
Token	Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
A	4			×	×			×	×			×	×			×	×		...
B	6				×	×	×				×	×	×				×	×	...
C	10						×	×	×	×	×						×	×	...
D	14								×	×	×	×	×	×	×				...

Table 3: Example of periodic toy data used to validate forecaster LSTM.

This toy data was converted into two sequence of tokens, once using separator tokens and once without using separator tokens. For both cases, the forecaster LSTM was able to reconstruct the toy data sequence with near-perfect accuracy, although the forecaster was slightly more accurate when the separator tokens were used. This experiment demonstrated that the forecaster LSTM architecture was functioning correctly, and that separator tokens improved the performance of the forecaster.

4.1.3 Forecaster LSTM Training

The forecaster LSTM was trained using the the MIMIC-III training set with a categorical crossentropy loss function. RMSProp with a learning rate of 0.01 was used. Figure 3 shows the convergence of accuracy and loss for both the training and validation sets over 60 epochs. Since slight overfitting was observed before the end of 60 epochs, the weights from the epoch with the best validation accuracy (19.28%) were used as the trained weights.

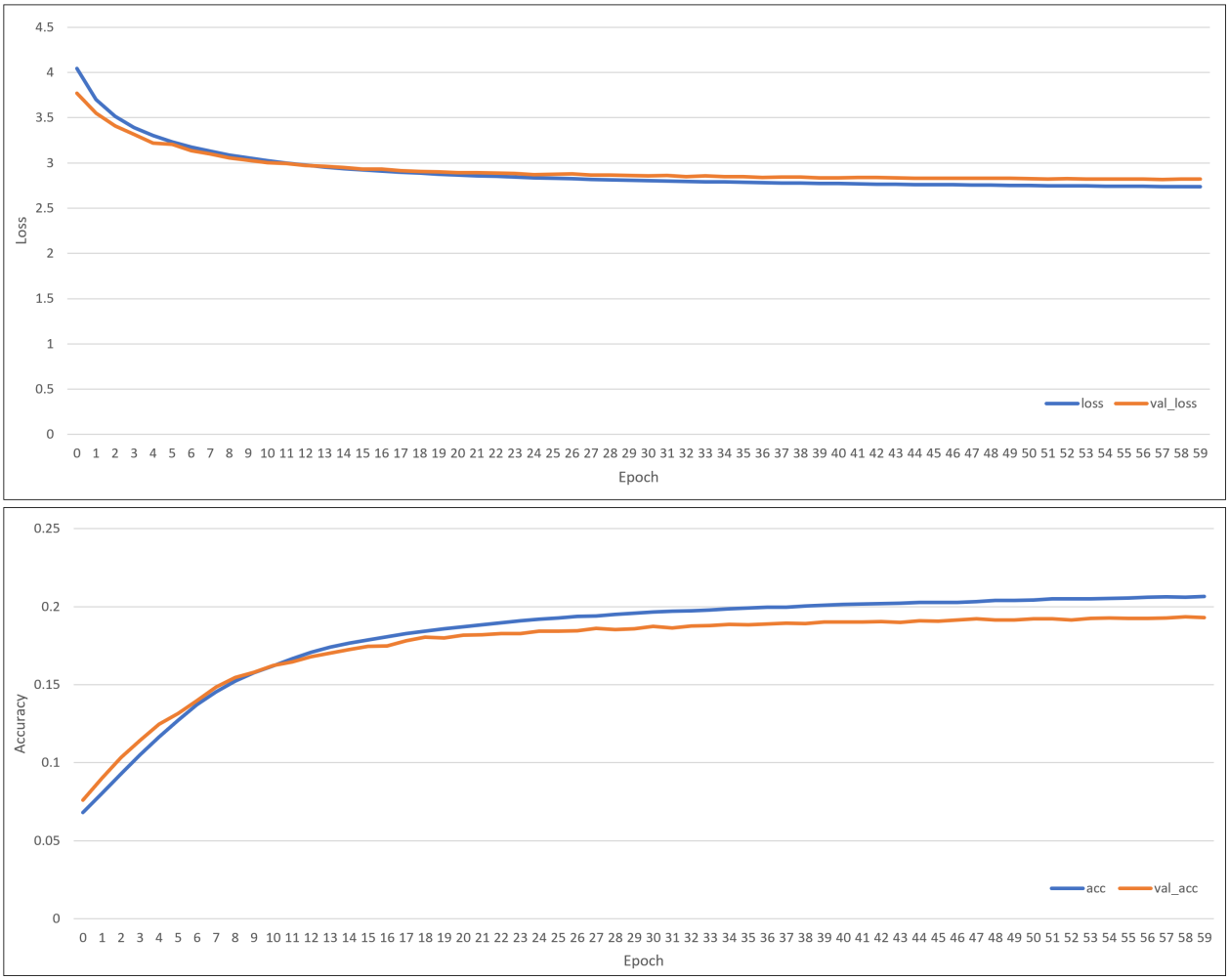


Figure 3: Forecaster LSTM training convergence (loss and accuracy).

The validation accuracy of 19.28% does not seem overly optimistic. However, this metric is based on the accuracy of the single top prediction. We plan to use the predicted probability distribution as opposed to just the top prediction, therefore conventional (i.e. top-1) accuracy is not a good metric to use in evaluating the training results. Instead, we use top-k accuracy, a similar metric to the top-k recall used by Choi, Bahadori, Schuetz, et al. [16]. Accuracy results are shown in Table 4 for different values of k. The top-k accuracies achieved by the forecaster LSTM show substantial improvement over those achieved by a strategy of always predicting the most common k tokens, demonstrating that the forecaster LSTM is an effective model of the probability distribution over a patient's next medication.

	Top-1 acc.	Top-10 acc.	Top-20 acc.	Top-30 acc.
Predicting most common k tokens	6.32%	22.48%	30.88%	37.20%
Forecaster LSTM	19.28%	52.73%	65.20%	72.33%

Table 4: Top-k accuracies for the forecaster LSTM compared to a naive strategy.

4.2 RETAIN

As briefly mentioned previously, RETAIN is a LSTM-based architecture developed by Choi, Bahadori, Kulas, et al. [32]. RETAIN was originally developed to predict patient heart failure. However, for this thesis their architecture will be trained to predict patient mortality. As both heart failure and mortality prediction are binary classification tasks, the architecture does not need to be modified for our purposes.

We will use RETAIN to both predict the likelihood of mortality for a given patient, as well as quantify how much each medication in the patient's history contributed to this prediction. Both the mortality prediction and the contribution values given by RETAIN will be used in different ways to compute metrics that quantify the likelihood of two drugs combining to yield a DDI; this will be discussed in more detail in a later section.

RETAIN takes as input a sequence of i admission representations $[x_1, x_2, x_3, ..., x_i]$, where each admission is represented by a set of the medications the patient was prescribed, that is, $x_k = \{m_{k,1}, m_{k,2}, m_{k,3}, ...\}$. RETAIN has two outputs. First, it outputs a scalar outcome prediction $\hat{y}$; in our case this can be interpreted as the predicted probability of patient mortality. The second output provides interpretability for the outcome prediction and is the main feature of RETAIN. This output assigns a contribution value $\omega_{k,l}$ to each medication $m_{k,l}$. Notably, as a consequence of the architecture of RETAIN, the sum of all contribution values, plus a scalar bias term b, yields the logit for the outcome prediction, that is, $\sigma(\Sigma[\omega_{k,l}] + b) = \hat{y}$.

4.2.1 RETAIN Architecture

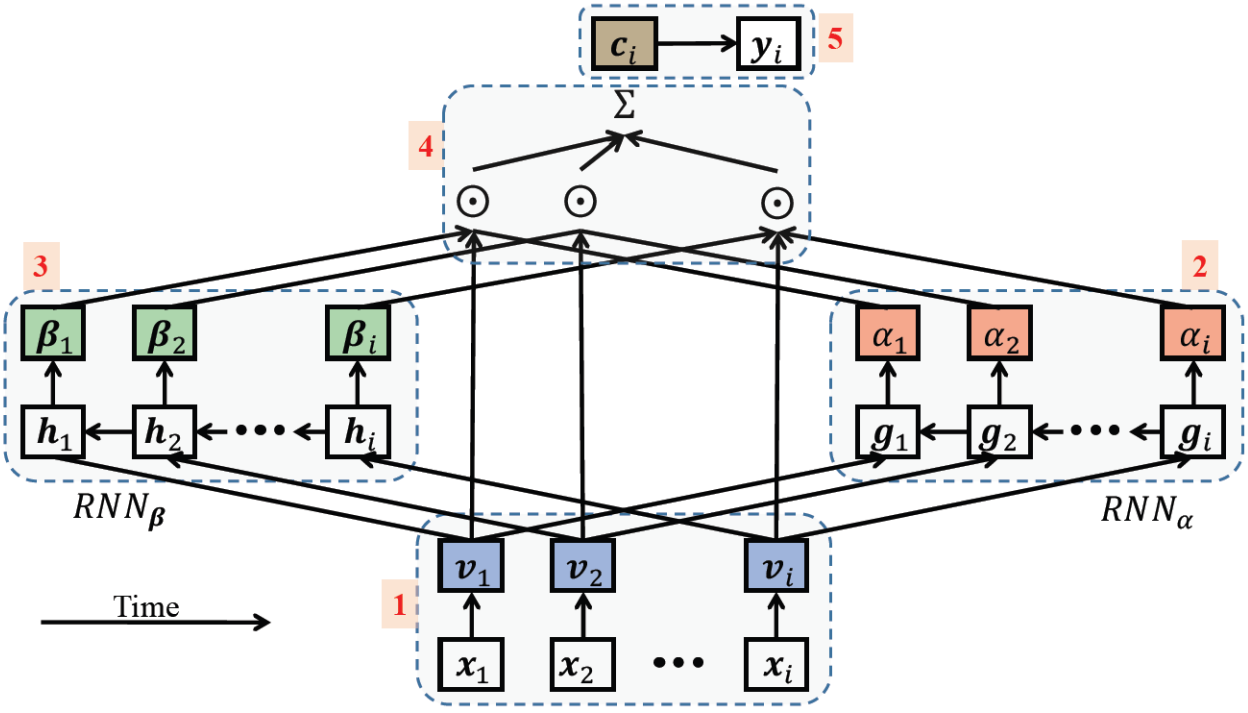


Figure 4: Architecture diagram of RETAIN.

The architecture diagram of RETAIN, taken from the original RETAIN paper [32], is shown in figure 4. This subsection will provide an overview of RETAIN's architecture and attention mechanism. Refer to the original RETAIN publication for more details [32].

As shown in figure 4, there are five main steps involved in RETAIN's architecture.

Step 1 involves converting the input sequence $[x_1, x_2, x_3, ..., x_i]$ to a sequence of admission embedding vectors $[v_1, v_2, v_3, ..., v_i], v_k \in \mathbb{R}^{n_{emb}} \ \forall k$. More specifically, a medication embedding matrix W_{emb} containing an embedding vector for each medication is trained; the admission embedding vector v_k is derived by summing the medication embedding vectors for all medications in x_k.

Steps 2 and 3 involve inputting the admission embedding vector sequence to two parallel LSTM blocks. The first LSTM block generates timestep-level attention weights $[\alpha_1, \alpha_2, \alpha_3, ..., \alpha_i], \alpha_k \in \mathbb{R} \ \forall k$ while the second LSTM block generates variable-level attention weights $[\beta_1, \beta_2, \beta_3, ..., \beta_i], \beta_k \in \mathbb{R}^{n_{emb}} \ \forall k$.

Step 4 first applies the timestep- and variable-level attention weights to the embedding sequence, resulting in the sequence $[\alpha_1\beta_1 \odot v_1, \alpha_2\beta_2 \odot v_2, \alpha_3\beta_3 \odot v_3, ..., \alpha_i\beta_i \odot v_i]$. The sum of the elements of this sequence yields the context vector $c \in \mathbb{R}^{n_{emb}}$.

Finally, step 5 involves passing c to a dense layer followed by sigmoid activation, yielding the final prediction $\sigma(Wc + b) = \hat{y}$.

RETAIN is more interpretable than traditional RNNs because LSTM blocks are not directly used to predict the final result; they instead only generate the attention weights. This formulation allows Wc to be decomposed into the sum of individual contributions from each medication in a patient's history. The exact formula is $\omega_{k,l} = \alpha_k W(\beta_k \odot W_{emb}[:, l])$, which is derived in the original RETAIN paper. A higher $\omega_{k,l}$ value means that RETAIN attributes a larger increase in mortality risk to medication $m_{k,l}$.

4.2.2 RETAIN Validation Experiments

We did not feel the need to validate the functionality of the RETAIN implementation, as the code used was from the original RETAIN research by Choi, Bahadori, Kulas, et al. [32]. However, our quantitative usage of RETAIN's interpretability outputs was not something that was validated by the original research, which intended for the ω values to be qualitatively interpreted by an end-user such as a doctor. Therefore a small experiment with toy data was carried out.

The toy data for this experiment involved an extremely simplified model of patient health and medications, and was based on similar experiments performed by Chae et al. [10]. Each "patient" was modelled using a single latent health variable h which was initialized at zero and clamped between two randomly generated bounds; at each timestep the patient had a probability $\sigma(h)$ of surviving to the next timestep. Two "medications" were modelled: m_a and m_b. m_a had minimal impact on h. m_b was beneficial and increased h unless it was administered alongside m_a, in which case both "medications" had minimal impact. At each timestep, the "patient" would randomly receive neither, one, or both medications. This algorithm was used to simulate training and test sets of "patients" and their outcomes; it was found that most of these simulated "patients" either died within the first few timesteps or survived indefinitely.

RETAIN was trained on the training "patient" set, achieving a near-perfect accuracy; the trained model was used to generate ω values for the test set of "patients". Table 5 shows the mean of ω values calculated

across various subsets of "medication" administrations in the test set:

Subset of simulated "patients"	"Medication"	
	m_a	m_b
Surviving "patients"	0.007	0.004
All "patients"	0.040	0.024
Non-surviving "patients"	0.239	0.169

Table 5: Means of ω values assigned by RETAIN for various subsets of toy data.

The mean ω values in "patients" who died is greater than that in "patients" who survived, which is consistent with expectations, as RETAIN should, on average, predict a larger value of $\hat{y}$ for patients who died, and $\hat{y} = \sigma(\Sigma[\omega] + b)$. Furthermore, m_b improves "patient" health, while m_a potentially prevents improvements to "patient" health. Therefore, the mean ω value for administrations of m_a is expected to be greater than that of m_b, which is once again consistent with these results. These findings confirmed that the ω values output by RETAIN were behaving as expected.

There were, however, some concerning observations regarding these results. The differences in mean ω between m_a and m_b were substantially smaller than the differences between surviving and non-surviving "patients". This showed that the same medication could be assigned vastly different ω values in different contexts, and that variation in ω due to context can potentially be much greater than variation due to differences in actual medication effects. This suggested that ω values assigned by RETAIN to a given patient are somehow strongly influenced by the characteristics of the entirety of the patient's history, as opposed to being independently interpretable outside of the context of the whole patient.

In the case of this toy data experiment, it was suspected that RETAIN had learned to use the number of "medications" in "patient" history to predict outcome, rather than learning the effects of the simulated "medications". This was a plausible hypothesis, since the populations of surviving and non-surviving "patients" differed greatly in mean number of "medications" per "patient". To test this, the toy data generation algorithm was modified such that neither m_a nor m_b would have any impact on "patient" health under any circumstances. New toy data was generated and RETAIN was retrained. Despite the modification, RETAIN achieved the same near-perfect accuracy as in the prior experiment, which confirmed that RETAIN was counting the number of "medications" to predict outcome, as opposed to modelling the effects of the "medications".

Overall, these toy data experiments on RETAIN provided conclusions which greatly informed the design of subsequent development of DDI identification models and experiments on real patient data. We found that directly comparing or aggregating ω values assigned to completely different patients is inadvisable, since a high degree of uncontrolled variation in the properties of different patients may exist. Instead, in order to compare or aggregate ω values, the differences between patients must be somehow controlled. The number of medications per patient was a specific example of a property which could confound RETAIN's ω values; this finding partially motivated the MIMIC-III preprocessing step of truncating patients to a maximum of 100 medications, as this truncation greatly decreases variation in the number of timesteps and medications per patient in the dataset.

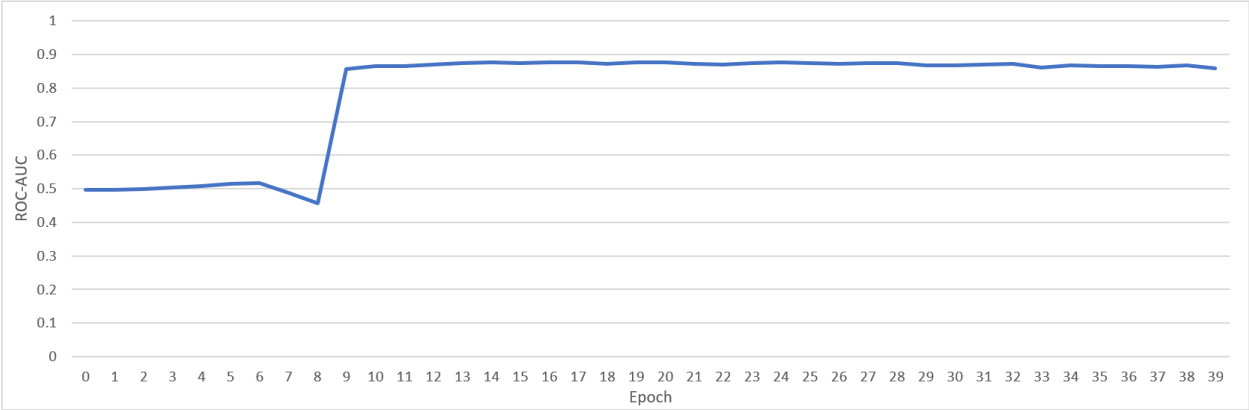
4.2.3 RETAIN Training

RETAIN was trained on the MIMIC-III training set with a binary crossentropy loss function. As per the original implementation, the AdaMax optimizer with L2 regularization was used. A two-stage grid search was performed to optimize hyperparameters for the MIMIC-III dataset; the results are shown in Table 6. Dropout in the dense layer, which was used by the original authors of RETAIN, was tried, but was found to have a detrimental effect for MIMIC-III data; optimal embedding and LSTM sizes were found to be 256.

n_{emb}	LSTM size	Dropout Rate	Val. PR-AUC
256	256	**0.0**	**0.799791**
256	256	0.2	0.798606
256	256	0.4	0.792274
512	512	0.0	0.796986
512	256	0.0	0.795547
256	512	0.0	0.791528
256	**256**	0.0	**0.799791**
256	128	0.0	0.796881
128	256	0.0	0.797035
128	128	0.0	0.796271

Table 6: Grid search results for RETAIN hyperparameters.

Figure 5 shows the convergence of validation ROC-AUC and PR-AUC over 40 epochs for the final training run with optimized hyperparameters. As was done with the forecaster LSTM early stopping was employed; the weights from the epoch with the best validation PR-AUC were used as the final weights.



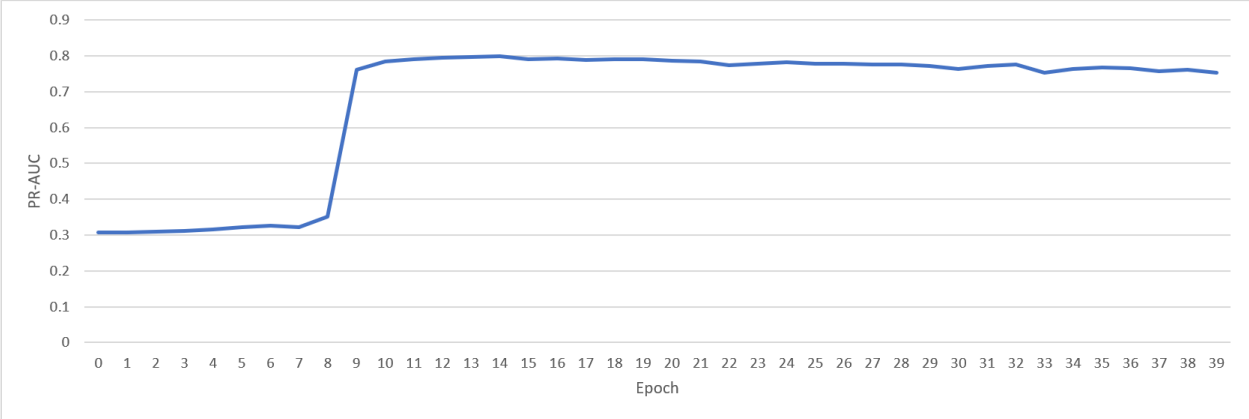


Figure 5: RETAIN training convergence (ROC-AUC and PR-AUC).

5 DDI Identification Models

As mentioned in the introduction, the DDI identification models developed in this thesis are based on the following ideas:

1. RNN neural networks can be trained to forecast and interpret patient treatment behavior and outcomes.
2. Synthetic patient data corresponding to specific drug combinations can be algorithmically generated.
3. RNN neural networks will behave differently when given patients exhibiting presence of a DDI.
4. This difference can be used to derive a quantitative measure of DDI likelihood.

All DDI identification models developed in this thesis will follow the same general structure for predicting interactions. The model will be given a pair of medications (m_a, m_b). The model will then (1) generate a set of synthetic patients and (2) use RETAIN to perform inference on the set of synthetic patients to compute a score $S_{a,b}$ representing likelihood of an adverse DDI resulting from the combination of m_a and m_b. Therefore, in order to design DDI identification models, we must (1) design algorithms to generate synthetic patient data corresponding to specific drug combinations and (2) design metrics based on RETAIN's outputs that quantify likelihood of a DDI. This thesis will explore two algorithms for generating synthetic patient data, as well as two metrics for quantifying DDI likelihood, yielding a total of four combinations to be evaluated.

In addition to these four models, two simple baseline strategies are considered for the sake of comparison.

Like the models described above, they assign a score $S_{a,b}$ to each pair of medications (m_a, m_b). These baselines will be briefly discussed at the end of this section.

5.1 Algorithms to Generate Synthetic Patient Data

Instead of generating individual synthetic patients, we plan to generate tuples of three variations of each synthetic patient. For each patient, the first variant will exhibit only drug m_a, the second variant will exhibit only drug m_b, and the third will exhibit both. Therefore, when evaluating a pair of drugs (m_a, m_b), the set of synthetic patients generated will have the form $\{(X_{1,a}, X_{1,b}, X_{1,ab}), (X_{2,a}, X_{2,b}, X_{2,ab}), (X_{3,a}, X_{3,b}, X_{3,ab}), ..., (X_{N_{syn},a}, X_{N_{syn},b}, X_{N_{syn},ab})\}$. For the experiments performed in this thesis, $N_{syn} = 128$ was selected.

As each (X_a, X_b, X_{ab}) tuple contains three variants on the same synthetic patient, it will be easier to isolate any effects of the drug interaction (m_a, m_b) from not only the effects of the individual drugs m_a and m_b, but also from the variation across different and unrelated patients. This is expected to allow us to develop DDI-likelihood metrics that mitigate the concerns discussed in section 4.2.2.

The two algorithms discussed here will be referred to as "sample-and-swap" and "virtual-experiments".

5.1.1 Sample-and-Swap Generation of Synthetic Patients

The sample-and-swap algorithm is the more straightforward out of the two.

To generate each tuple of synthetic patient variants (X_a, X_b, X_{ab}), a real patient is first sampled from the MIMIC-III test set. After this patient is sampled, the drugs prescribed during the patient's most recent admission are inspected; this set of drugs might contain both, one, or neither of m_a and m_b.

If both m_a and m_b was prescribed during the last admission, the unmodified patient can be used as the "both" variant X_{ab}. To generate the X_a variant, the prescription of m_b during the last admission is replaced with some other randomly selected medication. To generate the X_b variant, the prescription of m_a during the last admission is replaced instead. Random selection of replacement medications is done by sampling one of the 604 medications which are not m_a or m_b, with probabilities weighted by frequency in MIMIC-III.

If only m_a was prescribed during the last admission, the unmodified patient can be used as the X_a variant. To generate the X_{ab} variant, a random non-m_a medication during the last admission is randomly selected and replaced with m_b. m_a can then be replaced with some other randomly selected medication to yield the X_b variant. An analogous procedure is used if only m_b was prescribed during the last admission.

If neither medications of interest were prescribed during the last admission, an ordered pair of medications in the last admission are randomly selected. To generate X_a or X_b, the first or second selected medication is swapped with m_a or m_b, respectively. To generate X_{ab}, both are swapped.

The advantage of this approach is that it tightly controls the differences between the three synthetic patient variants. The procedure is designed such that all admissions prior to the last are completely identical across the X_a, X_b, X_{ab} variants; the last admissions differ only in the surgical swapping of one or two medications. Furthermore, by basing the synthetic patient variants on real patient data from the MIMIC-III test set, we expect distribution of synthetic patients to be extremely similar to the distribution of real

patients RETAIN was trained on; this may allow RETAIN's analysis of these synthetic patients to be more meaningful.

5.1.2 Virtual-Experiments Generation of Synthetic Patients

The virtual-experiments algorithm is slightly more complex than sample-and-swap, and leverages the trained forecaster LSTM to generate completely novel synthetic patient data. The idea is to use the forecaster LSTM to simulate the future treatment trajectory and outcome resulting from some initial condition, thus running "virtual" experimental trials.

To generate each tuple of synthetic patient variants (X_a, X_b, X_{ab}), a single short "probe" sequence of length $n_{probe} = 10$ is first generated. Each of the n_{probe} medications in this probe sequence is selected by randomly sampling one medication from the 604 medications which are not m_a or m_b, with probabilities weighted by frequency in MIMIC-III. Three variants of this probe sequence, corresponding to X_a, X_b, and X_{ab}, are then generated. This is done by first randomly selecting an ordered pair of the n_{probe} medications. The first, second, or both selected medication(s) are then swapped with m_a, m_b, or both, respectively, to yield the three probe variations.

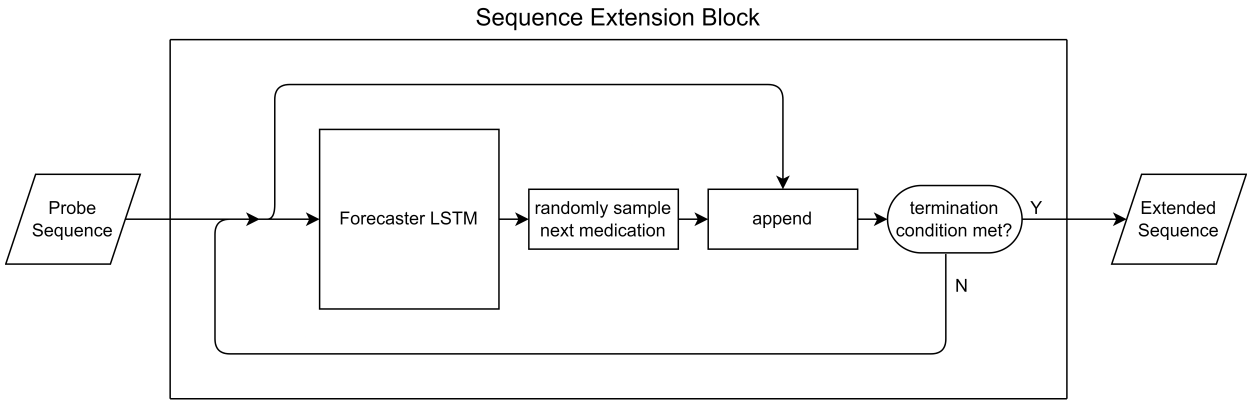


Figure 6: Block diagram of algorithm used to generate virtual-experiments data.

Each of the three probe variations are then independently extended using the forecaster LSTM. As mentioned in section 4.1, the technique used is that used for novel text generation in natural language processing applications [50]. Figure 6 shows a block diagram of this process. To extend the probe sequence $[m_1, m_2, m_3, ..., m_{n_{probe}}]$ by a single medication, the sequence is input into the forecaster LSTM, which outputs a probability distribution over the next token $\hat{m}_{n_{probe}+1} \in \mathbb{R}^{610}$. This distribution is adjusted to zero out the probabilities associated with m_a and m_b; a random token $m_{n_{probe}+1}$ is then sampled and appended to the original sequence to yield $[m_1, m_2, m_3, ..., m_{n_{probe}}, m_{n_{probe}+1}]$. This process is repeated until one of two stop conditions is reached: either the forecaster predicts a patient outcome token, or a maximum length of 100 tokens is reached. Finally, a postprocessing step ensures that each admission contains no more than one of each medication type by removing duplicates.

Whereas the sample-and-swap algorithm generates tuples of extremely similar synthetic patient variants, the virtual-experiments algorithm may create tuples of dissimilar patient variants; although the starting probe sequences are nearly identical, the independent extension of the probe sequences using a random

sampling process allows the three sequences to diverge. However, this process of simulating "virtual" patients creates synthetic patients which may model longer-term "butterfly effects" which may result from a drug interaction earlier in their treatment. The virtual-experiments method of generating synthetic patient variants is therefore somewhat complementary to the sample-and-swap algorithm: whereas the sample-and-swap algorithm seeks to emphasize acute effects of drug interactions by placing the interacting drugs in the patient's final admission, the virtual-experiments algorithm seeks to model long-term effects by placing the interacting drugs in the patient's first admission and simulating the resulting long-term progression.

5.2 Metrics for Quantifying DDI Likelihood

Regardless of whether the sample-and-swap or the virtual-experiments algorithm is used, synthetic patient variant tuples $\{(X_{1,a}, X_{1,b}, X_{1,ab}), (X_{2,a}, X_{2,b}, X_{2,ab}), (X_{3,a}, X_{3,b}, X_{3,ab}), ..., (X_{N_{syn},a}, X_{N_{syn},b}, X_{N_{syn},ab})\}$ will be generated to test the interaction between m_a and m_b. Two metrics will be developed and evaluated. Both metrics will reduce a single tuple of synthetic patient variants to a scalar score; the mean of these scores across all $N_{syn} = 128$ tuples will yield a final aggregated score $S_{a,b}$ representing the likelihood of a DDI.

Both metrics require running RETAIN inference on the synthetic patient variants; one metric is derived from RETAIN's ω contribution value outputs assigned to m_a and m_b, while the other will be derived from RETAIN's $\hat{y}$ mortality probability outputs.

5.2.1 Contribution-Value DDI Metric

To compute the contribution-value based metric for a given tuple (X_a, X_b, X_{ab}), specific ω values assigned by RETAIN will be extracted. More specifically, we are interested the ω values assigned to the instances of m_a and/or m_b inserted during the generation of synthetic patient variants. For data generated by the sample-and-swap algorithm, this corresponds to the instances of m_a and/or m_b in the last admission; for data from the virtual-experiments algorithm, this corresponds to the instances of m_a and/or m_b in the first admission. Regardless, four specific ω values will be extracted: ω_{only_a} will be extracted from the ω values assigned to X_a; ω_{only_b} will be extracted from the ω values assigned to X_b; ω_{both_a} and ω_{both_b} will be extracted from the ω values assigned to X_{ab}.

We wish to design a metric to quantify the amount by which combining m_a and m_b results in a "nonlinear" interaction. More specifically, we wish for the metric to be minimized in cases where $\omega_{only\_a} = \omega_{both\_a}$ and $\omega_{only\_b} = \omega_{both\_b}$. This represents a case where the presence or absence of m_b has no effect on the contribution of m_a, and vice versa. The metric developed to fit this critera is:

$$|(\omega_{both\_a} + \omega_{both\_b}) - (\omega_{only\_a} + \omega_{only\_b})|$$

5.2.2 Mortality-Probability Based Metric

The predicted mortality probabilities $\hat{y}$ can also be used to measure the severity of a drug interaction. For a given tuple (X_a, X_b, X_{ab}), RETAIN gives three mortality probability predictions $\hat{y}_a$, $\hat{y}_b$, and $\hat{y}_{ab}$.

A drug interaction between m_a and m_b is more severe if the likelihood of mortality is higher when both

drugs are given, as opposed to the worst-case scenario where only one drug is given. Therefore, we use:

$$\hat{y}_{ab} - \max(\hat{y}_a, \hat{y}_b)$$

5.3 Baseline Models

For the sake of comparison, two extremely simple baseline strategies are considered. The first and simpler baseline is a random guessing strategy; for each given pair of medications (m_a, m_b) this baseline simply assigns a random score $S_{a,b}$ between 0 and 1 with uniform probability.

The second baseline strategy employs ideas similar to those used in the mortality-probability based metric. For a given pair of medications (m_a, m_b), the patients in MIMIC-III are sorted into four sets based on the medications administered during their most recent admission. One set consists of all patients who were prescribed both m_a and m_b during their most recent admission; one set consists of all patients who were prescribed m_a but not m_b; one set consists of all patients who were prescribed m_b but not m_a; and the final set consists of all remaining patients. Average mortality rates for the patients in the former three sets are calculated to yield $\bar{y}_{ab}$, $\bar{y}_a$, and $\bar{y}_b$. The score assigned is calculated as $S_{ab} = \bar{y}_{ab} - \max(\bar{y}_a, \bar{y}_b)$.

6 Model Performance Evaluation Methods

This section will explain and justify the methods and metrics which will be used to evaluate and compare the performance of the DDI identification models.

6.1 Evaluation Metrics for DDI Identification Models

As mentioned in section 3.2, 711 TP drug interaction pairs and 711 TN drug interaction pairs are available for evaluation purposes. Each model will be given all 1422 pairs and will generate a score for each pair, with higher scores ideally being assigned to TP pairs.

The primary benefit of a DDI identification model is the ability to prioritize pairs of drugs in terms of DDI likelihood. We are therefore primarily concerned with the ability of the models to output scores which, when used to sort drug pairs, results in an effective partitioning of TP and TN interaction pairs; the absolute magnitude of scores output by the models are of less concern. The precision-recall curve is only affected by the relative order of scores, and is therefore an appropriate tool to determine the performance of our models. A quantitative performance score can be derived by integrating the area under the curve; this is a popular way to measure model performance which is employed by two of the existing DDI identification papers discussed in section 2.2 [6], [8]. Area Under Precision Recall Curve (AU-PRC) and Average Precision (AP) are metrics commonly used for this purpose. AU-PRC is computed using linear interpolation between points on the precision-recall curve, and can therefore lead to a slightly inflated measure of the area under the curve; we will therefore use AP instead, although the difference between these two metrics is likely to be minimal in practice.

For each model, the AP score will be computed across all 1422 drug pairs; the AP score across all 1422 pairs will be referred to as AP_{ALL} and will serve as the main measure of each model's performance.

We are additionally interested in whether or not our models are capable of generalizing to drug interactions which were not explicitly represented in MIMIC-III. Therefore, an "out-of-sample" (OOS) subset of the 1422 drug pairs was constructed. This subset contains all pairs of drugs where no patients in the MIMIC-III training set ever received both drugs during the same admission. This subset contains 44 TP pairs and 69 TN pairs. An additional score AP_{OOS} will be computed across only these 113 pairs and used as a secondary measure of the model's ability to generalize to drug interactions which have not been explicitly observed in the training data. Note that AP_{OOS} cannot be calculated for the statistical-mortality-rate baseline.

6.2 Confidence Testing on Average Precision Scores

With only 1422 total drug pairs, and only 113 pairs in the OOS subset, there was a degree of concern regarding models achieving a better-than-random performance by chance. Therefore, an informal confidence test was used to identify which models, if any, might have outperformed a random baseline by chance.

Monte-Carlo simulation ($n = 100,000$) was used to determine the distributions of AP scores achieved by the random baseline strategy for the all-pairs and OOS-pairs sets. The resulting distributions of AP scores are shown in Figure 7. For each model except the random baseline, these distributions were used to calculate the probabilities p_{ALL} and p_{OOS} of the random baseline matching or beating the model's AP_{ALL} and AP_{OOS} scores.

It is important to note that the only purpose of performing confidence testing was to identify any models which could have outperformed the random baseline by chance. The results of this confidence test were not interpreted as any meaningful validation of a model's performance or utility. In addition, since the distributions of random baseline AP scores do not model all sources of randomness or noise (e.g. from datasets) involved in this experiment, the results of this confidence test were not interpreted as having the capability to prove that any given model did not achieve its level of performance by chance.

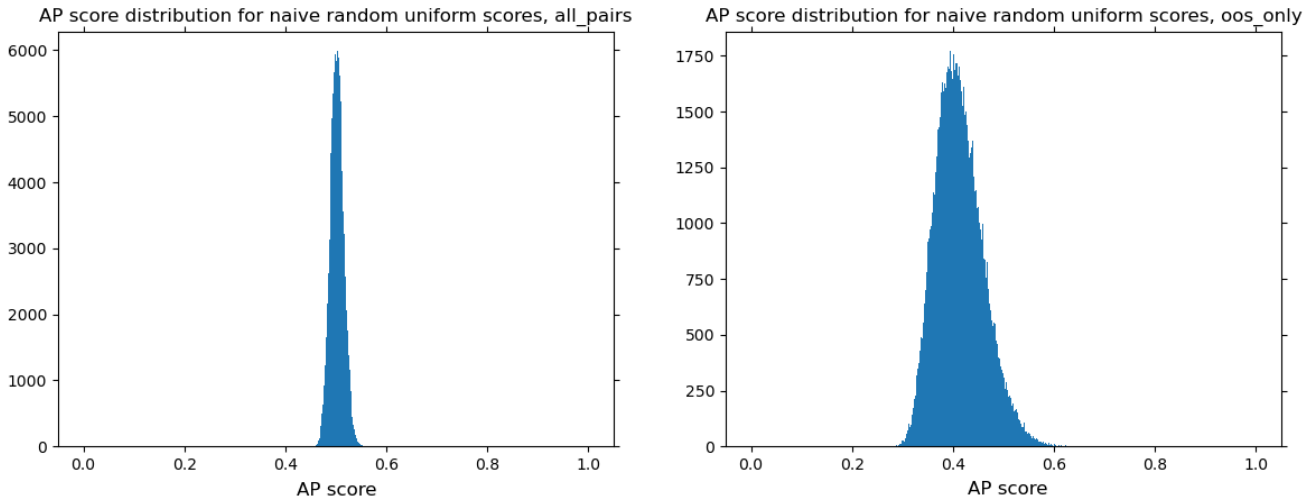


Figure 7: Distributions of AP_{ALL} and AP_{OOS} scores for a random baseline.

7 Results, Limitations, and Further Work

This section will begin by presenting and discussing the results of model evaluations. This will be followed by a discussion of the limitations and qualifications relevant to these results, and finally ideas regarding possible directions for future research will be presented.

7.1 Model Evaluation Results and Discussion

AP scores for models evaluated on both the full set of 1422 drug pairs and on the set of 113 OOS drug pairs are provided in table 7. Corresponding precision-recall curves will also be presented in this section.

Model	AP_{ALL}	p_{ALL}	AP_{OOS}	p_{OOS}
sample-and-swap + RETAIN contribution values	0.638028	0.000	0.532855	0.013
virtual-experiments + RETAIN contribution values	0.609331	0.000	0.690493	0.000
sample-and-swap + RETAIN mortality probs.	0.494431	0.720	0.563441	0.003
virtual-experiments + RETAIN mortality probs.	0.489683	0.832	0.596903	0.001
statistical mortality rate baseline	0.532982	0.013	-	-
random baseline	0.500000	-	0.389380	-

Table 7: AP scores for evaluated models.

It was found that all models which outperformed the random baseline did so with significance.

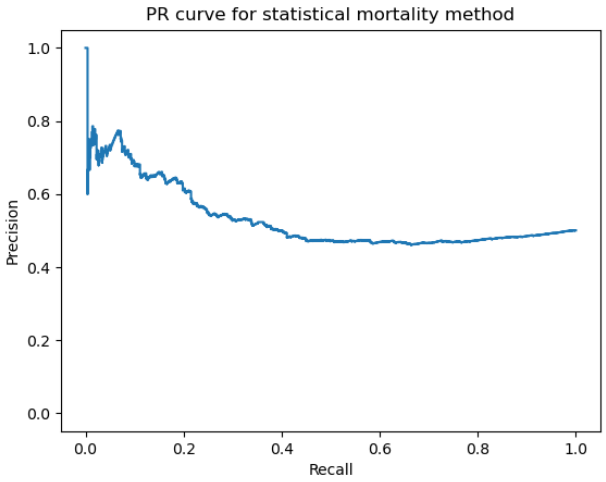
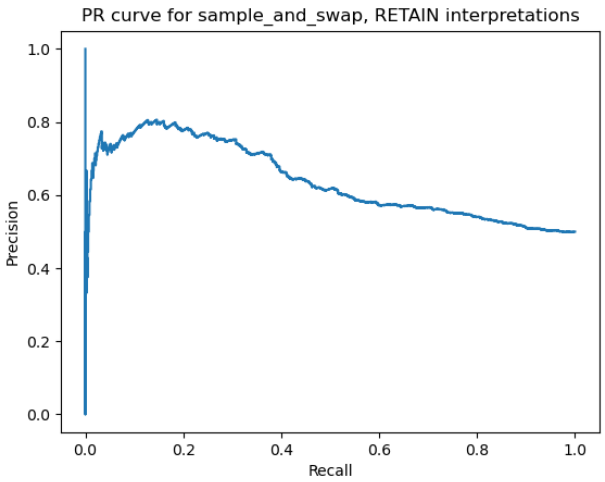


Figure 8: PR curve across all drug pairs for the statistical mortality rate baseline.

Figure 8 shows the precision-recall curve for the statistical mortality rate baseline. This baseline was able to outperform the random baseline as measured by AP_{ALL}, but as the margin of outperformance is extremely slim, this baseline model provides highly questionable real clinical utility. The precision of this model rapidly degrades to that of random guessing (0.5) beyond the most conservative of recall thresholds.



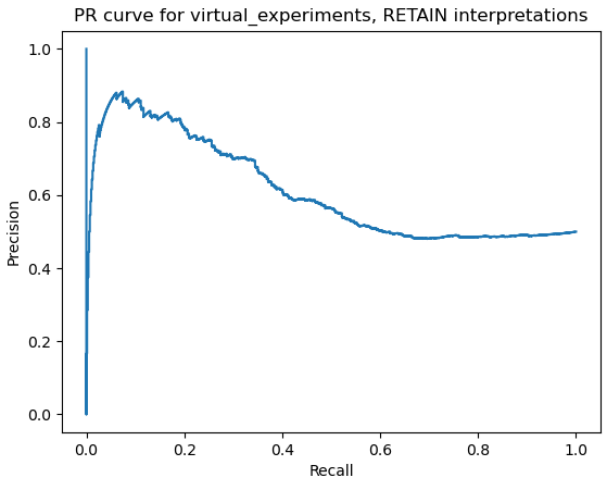
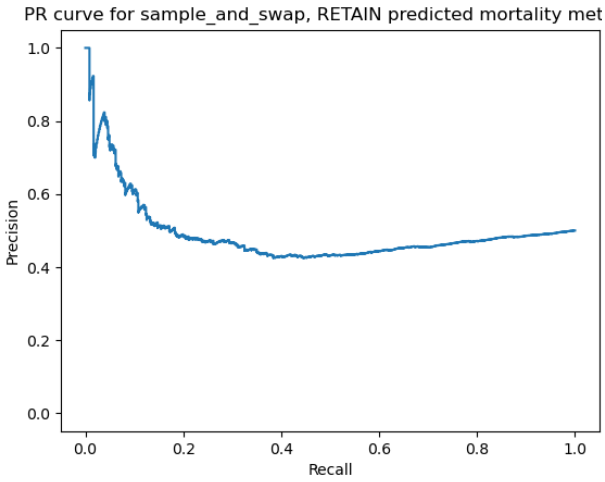


Figure 9: PR curve across all drug pairs for (sample-and-swap, RETAIN contribution values).

Figure 10: PR curve across all drug pairs for (virtual-experiments, RETAIN contribution values).

Figures 9 and 10 show the all-pairs precision-recall curves for the models based on RETAIN contribution value metrics. These models performed the best as measured by AP_{ALL}. Both models are able to achieve a precision of 0.8 with a recall of around 0.2, and thus might be able to provide some degree of clinical utility by identifying a small fraction of DDIs with some degree of reliability. However, neither of these models is capable of comprehensively identifying DDIs, as beyond a recall threshold of 0.5, precision drops to roughly that of random guessing.



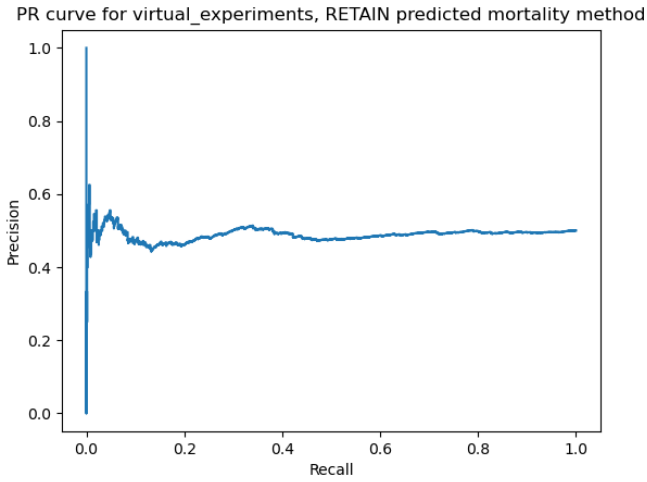


Figure 11: PR curve across all drug pairs for (sample-and-swap, RETAIN mortality predictions).

Figure 12: PR curve across all drug pairs for (virtual-experiments, RETAIN mortality predictions).

Figures 11 and 12 show the all-pairs precision-recall curves for the models based on RETAIN mortality predictions. Neither of these models are able to outperform the random baseline in terms of AP_{ALL}, with the precision-recall curve for (virtual-experiments, RETAIN mortality predictions) closely resembling that of a random guessing strategy (i.e. flat line at precision = 0.5), while the curve (sample-and-swap, RETAIN mortality predictions) actually shows precision which is inferior to random guessing for the majority of recall thresholds. One possible interpretation for this poor performance is as follows: Mortality probabilities predicted by RETAIN cannot be expected to be substantially closer to real-world values than those derived statistically from RETAIN's training data, since RETAIN simply learns to model the distribution of mortalities in its training data; therefore the DDI identification models based on RETAIN's predicted mortality rates are unlikely to perform better than statistical mortality rate baseline.

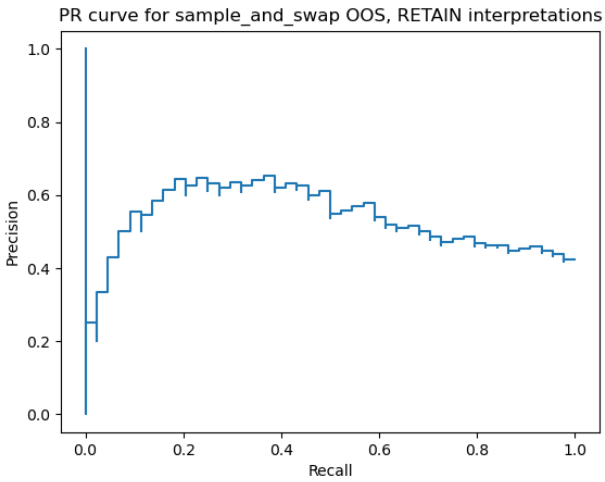


Figure 13: PR curve across OOS drug pairs for (sample-and-swap, RETAIN contribution values).

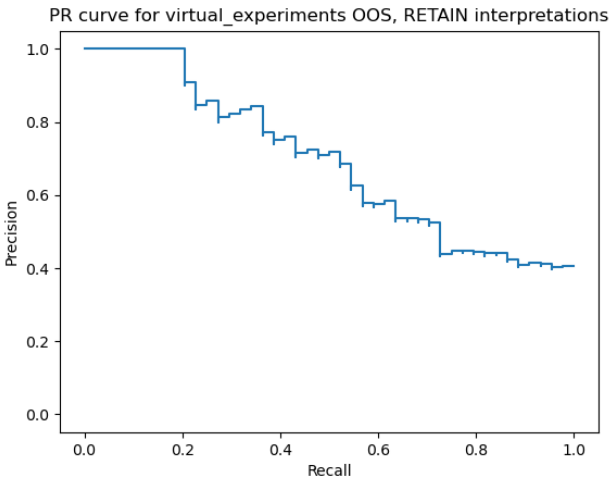


Figure 14: PR curve across OOS drug pairs for (virtual-experiments, RETAIN contribution values).

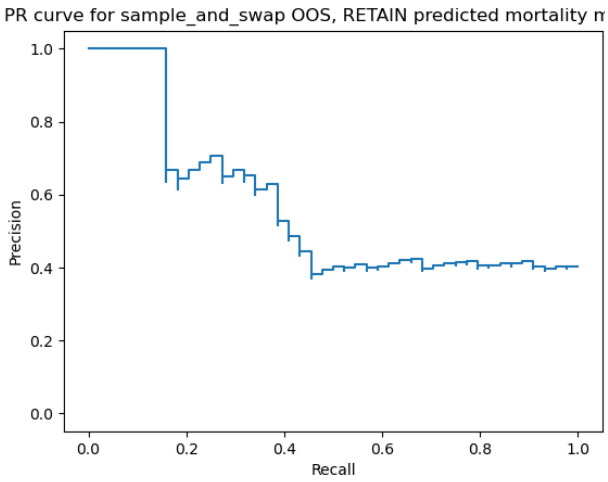


Figure 15: PR curve across OOS drug pairs for (sample-and-swap, RETAIN mortality predictions).

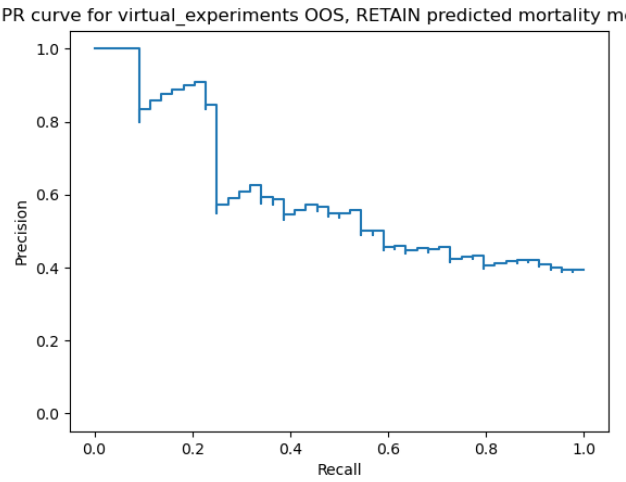


Figure 16: PR curve across OOS drug pairs for (virtual-experiments, RETAIN mortality predictions).

Figures 13 through 16 show the precision-recall curves computed over only out-of-sample drug pairs. It is highly surprising that, despite not performing well in terms of AP_{ALL}, the models based on RETAIN's predicted mortality rates substantially outperform the random baseline in terms of AP_{OOS}. Overall, all four models are able to outperform the random baseline in terms of AP_{OOS}, with the (virtual-experiments, RETAIN contribution values) model achieving the highest margin.

The DDI identification models based on RETAIN's contribution values were able to outperform a random baseline in terms of both AP_{ALL} and AP_{OOS}. An optimistic interpretation of these results might assert that they demonstrate the ability of these DDI models to generalize to previously-unseen drug interactions. However, especially in light of the unexpected AP_{OOS} scores achieved by the models based on RETAIN's predicted mortality rates, a more grounded interpretation would need to take into account the many factors which could have biased these results. These factors will be discussed in the next subsection.

7.2 Limitations and Qualifications

Many, but not all, of the the limitations of this work stem primarily from the characteristics of the two datasets used. The following subsections will discuss some of the characteristics of these datasets which should be taken into account when interpreting the results of this research, as well as some other more general flaws and limitations.

7.2.1 Limitations of the MIMIC-III EHR Dataset

The MIMIC-III dataset, while extremely useful, is not flawless. More specifically, there are at least five characteristics of this dataset which are worth recognizing as potential problems in the context of the experiments discussed.

First, the prescriptions records in MIMIC-III are exactly that - prescriptions. The MIMIC-III documentation [1] acknowledges that not all prescribed medications may have been actually administered to the patient. This introduces a form of noise in the MIMIC-III data which is very difficult to account for.

Second, medication prescriptions are timestamped to only the nearest day. Since many medications are typically prescribed per day, there is a substantial degree of ambiguity with respect to the true order of medication prescriptions, especially in the case of admissions which are shorter in duration.

Third, prescriptions in MIMIC-III may either be short-term or long-term prescriptions, with both a starting and ending date given. For the purposes of these experiments, the ending dates were disregarded, which potentially discards meaningful information regarding a patient's treatment.

Fourth, survival and mortality in the dataset cannot be completely ascribed to the medications prescribed: the healthcare of a patient is a complex system of which medication is just one component. Furthermore, the long-term nature of the dataset means that the death of a patient may occur some time after the patient's last admission, further obfuscating any potential causal relations.

Finally, the majority of patients in MIMIC-III are admitted only a few times. For some number of admissions n, figure 17 shows the number of patients with at least n admissions. It can be observed that the vast majority of patients were only admitted once, and the proportion of patients who have more than three or four admissions is miniscule. This property of the MIMIC-III dataset may work to counteract the

effectiveness of modelling it with RNNs; this is especially a concern with RETAIN, which uses one timestep per admission, and was originally developed to operate on much longer sequences than are available in MIMIC-III. It was observed by Chae et al. [10] that RETAIN could not be trained to as high a degree of performance on MIMIC-III as compared to that reported by the original RETAIN authors [32]; this observation was confirmed during the training of RETAIN for this thesis. The smaller number of admissions per patient in MIMIC-III as compared to the data used by the original RETAIN authors may at least partially account for this difference.

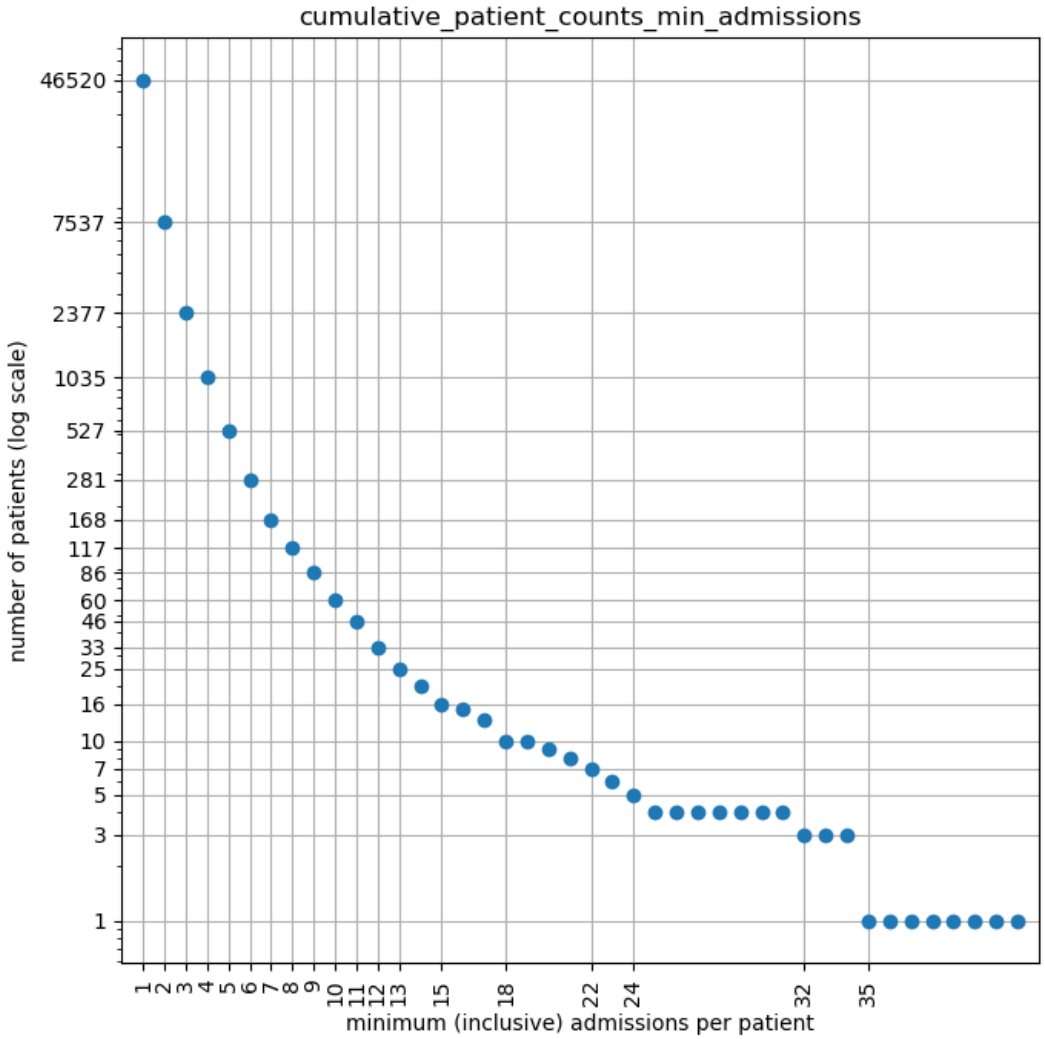


Figure 17: Number of patients with at least some number of admissions in MIMIC-III.

7.2.2 Limitations of the BIOSNAP Chem-Chem Dataset

The BIOSNAP Chem-Chem dataset is also subject to some limitations in the context of this research.

First, the DDI pairs in the dataset are derived by algorithmically processing medical research literature, and are not validated by experts. This introduces some degree of risk of spurious pairs being erroneously

introduced into the dataset.

Second, although the original BIOSNAP DDI pair list is fairly sizable, most of these pairs involved drugs that could not be mapped to a representation compatible with MIMIC-III, either due to different naming conventions or absence/rarity of the drugs in MIMIC-III. This resulted in a fairly small set of 711 successfully mapped pairs, and an even smaller set of 113 OOS drug pairs. These small datasets introduce the risk of random chance skewing evaluating results, especially in the case of the smaller set of OOS drug pairs.

Third, the BIOSNAP DDI pair list does not provide any information about interaction severity. The objective of this research is to develop a model capable of identifying high-severity DDIs. However, since no severity information is provided for the DDI pairs from BIOSNAP, it is possible that the dataset contains interactions which are only mildly harmful, or that do not result in patient mortality. This makes it more difficult to interpret and trust the evaluation results of this research.

7.2.3 Other Potential Issues

Beyond issues with the specific datasets used in this research, there are some other flaws in methodology which are worth acknowledging.

First, as the BIOSNAP dataset does not provide examples of drug pairs that do not result in severe adverse interactions, we needed to randomly sample drug pairs to serve as TN examples for evaluation. The sampling process was designed to prevent sampling TP pairs present in the BIOSNAP dataset. However, it would be impossible to prevent the random sampling of a hitherto-unknown DDI pair. Additionally, as the TP pairs were taken from the BIOSNAP dataset while the TN pairs came from random sampling, the difference in the underlying distribution of the TP and TN sets may have biased into the evaluation results.

Second, the usage of only one EHR dataset and one DDI dataset means that the ability of the models and results discussed in this thesis to generalize across different datasets was not evaluated in any way.

Finally, as a much of the Python code used for this research was developed by a single individual and has not undergone any form of review, the existence of at least some bugs escaping detection is a near-certainty. (Code is available; see Appendix A).

7.3 Potential Future Improvements

Many of the issues and limitations in the preceding subsection can be addressed, at least to some degree. For example, the limitations of MIMIC-III can be overcome by simply experimenting with alternative EHR datasets; this would also serve the purpose of testing generalization across different datasets. Some alternative DDI datasets were discussed in section 2.1.2; some of these datasets provide interaction severity, interaction category, or examples of benign interactions. Experimenting with these datasets could provide several benefits, such as increased scale of evaluation data, more contextual information about evaluated interactions to help with interpreting evaluation results, and availability of expert-validated TP and TN interaction pairs.

Beyond these corrective measures, it seems as if the general structure of the DDI identification models developed in this thesis have some degree of promise, especially when using RETAIN's contribution score outputs. This structure could be further explored by investigating additional metrics to quantify interaction likelihood. Additionally, as discussed in section 4.1.3, the forecaster LSTM used for this research, despite

being relatively simple in terms of size and architecture, is able to provide competitive predictions for a patient's next medication. This thesis did not explore the potential of using more complex forecaster models; this is another area of potential future progress.

It would also be useful to gain a better understanding of the factors which affect the performance of the proposed DDI identification models. To this end, for each individual drug or drug of drugs pairs, quantitative properties based on the patients administered said drug(s) could be evaluated through statistical methods on the EHR dataset. These measures might include:

- Average mortality rate of patients administered the drug(s)
- Number of times the drug(s) were administered
- Number of patients who were administered the drug(s)
- Mean number of admissions for patients administered the drug(s)
- Mean number of prescriptions for patients administered the drug(s)
- Average demographic properties of patients administered the drug(s)

The correlation of these quantitative properties with DDI identification performance could be evaluated. Such an experiment could provide data which informs the development of improved models or selection of more appropriate datasets.

8 Conclusion

This thesis research project has built open previous work from Chae et al. [10] on using EHR data to identify adverse drug-drug interactions. More specifically, key novel contributions include:

1. Modification and refactoring of the forecaster LSTM developed by Chae et al., as well as training and evaluation of the modified forecaster LSTM on the MIMIC-III dataset, and demonstration of competitive prediction accuracy.
2. Design of a method to effectively aggregate RETAIN's contribution values across multiple patients.
3. Design of a general framework for identifying DDIs without directly training on labelled DDI data, as well as development and evaluation of four specific DDI identification models within this framework based upon RETAIN and the forecaster LSTM.

The DDI identification framework, and components thereof, developed in this thesis utilize EHR data from MIMIC-III for training. Existing research targeting algorithmic DDI identification utilizes models that directly train on labelled DDI data; this framework provides unique value, as EHR data is potentially easier to collect at scale compared to labeled DDI data. The results of quantitative evaluation of the DDI models developed show that there is potential for this framework to provide clinical utility; specifically, the models based on contribution scores assigned by RETAIN were found to be capable of achieving substantially better-than-random performance in identifying adverse drug interactions, including drug interactions not represented in training data. However, much additional work is required to both address the limitations of this research and further explore additional methods of using this DDI identification framework. Key future steps include validating the models developed in this research on larger EHR and DDI datasets, as well as experimenting with additional synthetic-patient-data generation methods and DDI-likelihood metrics.

References

[1] J. A., P. T., and M. R., "Mimic-iii clinical database (version 1.4)," *PhysioNet*, 2016. DOI: https://doi.org/10.13026/C2XW26.

[2] J. A. E. W., P. T. J., S. L., *et al.*, "Mimic-iii, a freely accessible critical care database.," *Scientific Data*, vol. 3, no. 160035, 2016.

[3] G. A., A. L., G. L., *et al.*, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," vol. 101, no. 23, e215–e220, 2000.

[4] J. E. Zimmerman, A. A. Kramer, D. S. McNair, and F. M. Malila, "Acute physiology and chronic health evaluation (apache) iv: Hospital mortality assessment for today's critically ill patients," *Crit Care Med*, vol. 34, no. 5, pp. 1297–310, 2006. DOI: http://dx.doi.org/10.1097/01.CCM.0000215112.84523.F0.

[5] H.-B. Fang, X. Chen, X.-Y. Pei, S. Grant, and M. Tan, "Experimental design and statistical analysis for three-drug combination studies," *Statistical Methods in Medical Research*, vol. 26, no. 3, pp. 1261–1280, 2017, PMID: 25744107. DOI: 10.1177/0962280215574320. eprint: https://doi.org/10.1177/0962280215574320. [Online]. Available: https://doi.org/10.1177/0962280215574320.

[6] G. Lee, C. Park, and J. Ahn, "Novel deep learning model for more accurate prediction of drug-drug interaction effects," *BMC Bioinformatics*, vol. 20, no. 415, 2019. DOI: https://doi.org/10.1186/s12859-019-3013-0.

[7] N. Liu, C.-B. Chen, and S. Kumara, "Semi-supervised learning algorithm for identifying high-priority drug–drug interactions through adverse event reports," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 1, pp. 57–68, 2020. DOI: 10.1109/JBHI.2019.2932740.

[8] A. Kastrin, P. Ferk, and B. Leskošek, "Predicting potential drug-drug interactions on topological and semantic similarity features using statistical learning," *PLOS ONE*, vol. 13, no. 5, pp. 1–23, May 2018. DOI: 10.1371/journal.pone.0196865. [Online]. Available: https://doi.org/10.1371/journal.pone.0196865.

[9] C. F. and Z. Z., "Machine learning-based prediction of drug-drug interactions by integrating drug phenotypic, therapeutic, chemical, and genomic properties," *J Am Med Inform Assoc.*, vol. 21, e278–e286, 2014. DOI: doi:10.1136/amiajnl-2013-002512.

[10] D. Chae and M. Guerzhoy, "A systematic formulation of complex hypotheses from an icu mortality prediction model based on recurrent neural networks with lstm units," 2020.

[11] D. Charles, M. Gabriel, and M. F. Furukawa, "Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2012," Tech. Rep., 2013.

[12] Y. N., J. E., and H. E., "Office-based physician electronic health record adoption," Tech. Rep., 2016.

[13] T. Tran, W. Luo, D. Phung, *et al.*, "A framework for feature extraction from hospital medical data with applications in risk prediction," *BMC bioinformatics*, vol. 15, p. 6596, Dec. 2014. DOI: 10.1186/s12859-014-0425-8.

[14] H. Suresh, N. Hunt, A. Johnson, L. A. Celi, P. Szolovits, and M. Ghassemi, "Clinical event prediction and understanding with deep neural networks," *Proceedings of Machine Learning for Healthcare*, vol. 68, Jan. 2017.

[15] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach," *Journal of Biomedical Informatics*, vol. 69, pp. 218–229, 2017, ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2017.04.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046417300710.

[16] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor AI: predicting clinical events via recurrent neural networks," *CoRR*, vol. abs/1511.05942, 2015. arXiv: 1511.05942. [Online]. Available: http://arxiv.org/abs/1511.05942.

[17] D. A. Kaji, J. R. Zech, J. S. Kim, *et al.*, "An attention based deep learning model of clinical events in the intensive care unit," *PLoS One*, vol. 14, 2019. DOI: https://doi.org/10.1371/journal.pone.0211057. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6373907/#pone.0211057.s003.

[18] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with lstm recurrent neural networks," 2017. arXiv: 1511.03677 [cs.LG].

[19] W. Farhan, Z. Wang, Y. Huang, S. Wang, F. Wang, and X. Jiang, "A predictive model for medical events based on contextual embedding of temporal sequences," *JMIR Med Inform*, vol. 4, 2016. DOI: https://doi.org/10.2196/medinform.5977. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5148810/#app1.

[20] M. Zitnik, R. Sosič, S. Maheshwari, and J. Leskovec, *BioSNAP Datasets: Stanford biomedical network dataset collection*, http://snap.stanford.edu/biodata, Aug. 2018.

[21] P. S., van der Sijs H., T. A. D., *et al.*, "Drug-drug interactions that should be non-interruptive in order to reduce alert fatigue in electronic health records," *J Am Med Inform Assoc.*, vol. 20, pp. 489–93, 2013. DOI: 10.1136/amiajnl-2012-001089.

[22] D. Wishart, Y. Djoumbou, A. C. Guo, *et al.*, "Drugbank 5.0: A major update to the drugbank database for 2018," *Nucleic acids research*, vol. 46, Nov. 2017. DOI: 10.1093/nar/gkx1037.

[23] D. J. Rogers and T. T. Tanimoto, "A computer program for classifying plants," *Science*, vol. 132, no. 3434, pp. 1115–1118, 1960. DOI: 10.1126/science.132.3434.1115. eprint: https://www.science.org/doi/pdf/10.1126/science.132.3434.1115. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.132.3434.1115.

[24] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152, ISBN: 089791497X. DOI: 10.1145/130385.130401. [Online]. Available: https://doi.org/10.1145/130385.130401.

[25] D. E. Rumelhart and J. L. McClelland, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362.

[26] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, DOI: https://doi.org/10.1207/s15516709cog1402_1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1.

[27] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 5, pp. 157–66, Feb. 1994. DOI: 10.1109/72.279181.

[28] K. M. Hermann, T. Kociský, E. Grefenstette, *et al.*, "Teaching machines to read and comprehend," *CoRR*, vol. abs/1506.03340, 2015. arXiv: 1506.03340. [Online]. Available: http://arxiv.org/abs/1506.03340.

[29] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *CoRR*, vol. abs/1506.07503, 2015. arXiv: 1506.07503. [Online]. Available: http://arxiv.org/abs/1506.07503.

[30] K. Xu, J. Ba, R. Kiros, *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," *Proceedings of Machine Learning Research*, vol. 37, pp. 2048–2057, 2015.

[31] "International classification of diseases,tenth revision (icd-10)." (), [Online]. Available: https://www.cdc.gov/nchs/icd/icd10.htm.

[32] E. Choi, M. T. Bahadori, J. A. Kulas, A. Schuetz, W. F. Stewart, and J. Sun, "RETAIN: interpretable predictive model in healthcare using reverse time attention mechanism," *CoRR*, vol. abs/1608.05745, 2016. arXiv: 1608.05745. [Online]. Available: http://arxiv.org/abs/1608.05745.

[33] "Clinical classifications software (ccs) for icd-9-cm." (), [Online]. Available: https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp.

[34] "Clinical classifications software for services and procedures." (), [Online]. Available: https://www.hcup-us.ahrq.gov/toolssoftware/ccs_svcsproc/ccssvcproc.jsp.

[35] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Using recurrent neural network models for early detection of heart failure onset," *J Am Med Inform Assoc*, vol. 24, pp. 361–370, 2017. DOI: https://doi.org/10.1093/jamia/ocw112. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5391725/#ocw112-B20.

[36] 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*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[37] C. Esteban, O. Staeck, Y. Yang, and V. Tresp, "Predicting clinical events by combining static and dynamic information using recurrent neural networks," *CoRR*, vol. abs/1602.02685, 2016. arXiv: 1602.02685. [Online]. Available: http://arxiv.org/abs/1602.02685.

[38] A. Rajkomar, E. Oren, K. Chen, *et al.*, "Scalable and accurate deep learning for electronic health records," *CoRR*, vol. abs/1801.07860, 2018. arXiv: 1801.07860. [Online]. Available: http://arxiv.org/abs/1801.07860.

[39] L. Rasmy, M. Nigo, B. S. Kannadath, *et al.*, "Covrnn—a recurrent neural network model for predicting outcomes of covid-19 patients: Model development and validation using ehr data," *medRxiv*, 2021. DOI: 10.1101/2021.09.27.21264121. eprint: https://www.medrxiv.org/content/early/2021/09/29/2021.09.27.21264121.full.pdf. [Online]. Available: https://www.medrxiv.org/content/early/2021/09/29/2021.09.27.21264121.

[40] A. Rafiei, A. Rezaee, F. Hajati, S. Gheisari, and M. Golzan, "Ssp: Early prediction of sepsis using fully connected lstm-cnn model," *Computers in Biology and Medicine*, vol. 128, p. 104 110, 2021, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2020.104110. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482520304418.

[41] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *In ICML*, 2004.

[42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. arXiv: 1207.0580. [Online]. Available: http://arxiv.org/abs/1207.0580.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jun. 2014.

[44] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. arXiv: 1409.2329. [Online]. Available: http://arxiv.org/abs/1409.2329.

[45] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," 2016. arXiv: 1506.02142 [stat.ML].

[46] D. Krueger, T. Maharaj, J. Kramár, *et al.*, "Zoneout: Regularizing rnns by randomly preserving hidden activations," *CoRR*, vol. abs/1606.01305, 2016. arXiv: 1606.01305. [Online]. Available: http://arxiv.org/abs/1606.01305.

[47] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *CoRR*, vol. abs/1708.02182, 2017. arXiv: 1708.02182. [Online]. Available: http://arxiv.org/abs/1708.02182.

[48] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. arXiv: 1311.2901. [Online]. Available: http://arxiv.org/abs/1311.2901.

[49] C. Esteban, D. Schmidt, D. Krompaß, and V. Tresp, "Predicting sequences of clinical events by using a personalized temporal latent embedding model," in *2015 International Conference on Healthcare Informatics*, 2015, pp. 130–139. DOI: 10.1109/ICHI.2015.23.

[50] D. Pawade, A. Sakhapara, M. Jain, N. Jain, and K. Gada, "Story scrambler - automatic text generation using word level rnn-lstm," *International Journal of Information Technology and Computer Science*, vol. 10, pp. 44–53, Jun. 2018. DOI: 10.5815/ijitcs.2018.06.05.

Appendix A: Code Availability

Python code used for this research, as well as the .yaml file for prerequisite setup using Anaconda, is available on Github from the following link: `https://github.com/morgannewellsun/icu_hypotheses`.

The data processing and experiments discussed in this thesis can be replicated by running the scripts in the "refactored" directory as follows. All arguments are simply data/output paths. More comprehensive documentation will be available in the Github README in the future.

1. preprocess_and_profile_mimic.py (unpack MIMIC-III data)
2. map_drugbank_to_ndc.py (map BIOSNAP dataset to use NDC codes)
3. train_lstm_on_mimic.py (forecaster LSTM training)
4. evaluate_lstm_topk_accuracy.py (LSTM top-k accuracy)
5. train_retain_on_mimic.py (RETAIN training)
6. run_virtual_experiments.py (generate virtual-experiments synthetic patient variant tuples)
7. generate_retain_interpretations.py (first run; get RETAIN outputs for virtual-experiments)
8. sample_and_swap.py (generate sample-and-swap synthetic patient variant tuples)
9. generate_retain_interpretations.py (second run; get RETAIN outputs for sample-and-swap)
10. find_interactions_*.py (run and evaluate various DDI identification models)

This page is intentionally left blank.