Predicting Clinical Events with Machine Learning

Stephen Dove*

December 2018

Abstract

In an age where many health care organizations are implementing electronic health records, we have more access to patient health information in convenient databases. These databases give organizations an opportunity to create reports and find trends in their patients. Naturally, as we accumulate more data, we look to predict and prevent unwellness. This paper aims to predict both 30-day emergency department recidivism and 1-year opioid overdose as a patient is discharged from an emergency department visit.

1 Introduction

Electronic health records systems (EHRs) are collections of digital copies of a patient's medical record. In the past, medical records were physical paper-based records filed in a hospital or a physician's office and as such, following a patient's longitudinal health history even within the same organization was more difficult than it is now. A few simple SQL queries can now return a log of a patient's outpatient (OP), inpatient (IP), and emergency department (ED) visits/admissions in seconds. This information can include data on vital signs, medications, diagnoses, procedures, and much more. Given this ease of access to protected health information (PHI), there are strict security measures that go along with EHRs. We will discuss the security measures and roadblocks in section 3.

Initially, this data was mostly used for static reports by SQL report writers or by clinicians using various tools to look at their patient base. With the amount of data increasing, there is now a more significant push to use machine learning to help analyze this resource. With these predictive models, we will be able to support clinicians and ultimately benefit patient care.

In this paper, we look at two specific clinical events: 30-day emergency department (ED) recidivism and 1-year opioid overdose in the ED. 30-day ED recidivism refers to a patient that is discharged from the ED returning to the ED within 30 days of being discharged. These returns visits are costly in terms of money and resources from insurance companies, time for nurses and ED physicians, and, most importantly, patient care. If a patient needs to return to the ED within 30 day, there may have been a missed indicator that a predictive model could pick up on. If this second visit could be avoided by choosing to send the patient to a case manager, referring them to a specialist, or changing their medication, then a model needs to be developed to assist the physicians.

Similarly, opioid overdose has become a rising issue in the past few decades. Since 2002, there has been a steady increase in opioid related deaths in the United States especially with synthetic

*S. Dove is Master's Student of Computer Science, Brown University, Providence, RI 02912 `stephen_dove at brown.edu` opioids predominantly fentanyl. With synthetic opioids, there were 22 times the number of deaths in 2017 than there were in 2002.[1] The ability to predict whether or not a patient being discharged from the ED will return with an opioid overdose within one year would be beneficial to fight against the opioid epidemic.

One of the main issues when working with EHR data is that there is an abundance of data, but this data is not intuitively sorted into features that we can feed into a machine learning model. Before creating a predictive model, we need to first organize the data into predictive features. For example, diagnoses are represented as ICD-10 codes in an ontology from WHO and this ICD-10 database has over 68,000 diagnosis codes. In order to use this discrete data, we need to bucket these codes into groups of different types of diagnosis codes. We go deeper into this problem in section 3. We also recognize that in order to obtain the maximum information from a patient's medical history, we likely need to add a temporal component to our model. If a patient is prescribed hydrocodone one year ago, they may be more likely to develop an addiction than someone prescribed hydrocodone ten years ago. However, there are many features that we would like to use that are not temporal like demographics.

There has been research in similar areas of healthcare dealing with opioid overdose in general and hospital admissions. Many of these models treat a finite set of clinical events as tokens of a sequenced input.[1, 2] This approach acknowledges the temporal component of the data, but does not distinguish great lengths of time between clinical events. These models are both dealing with mostly single hospital admissions rather than the full life of a patient. As we mentioned above with the prescription example, depicting length of time is just as important as the order of events. These models also do not incorporate the static patient data like demographics.

We believe that both distinguishing length of time between clinical events and static patient data should be included in the model and it will increase our model's performance in predicting both opioid overdose and ED recidivism. Our proposed model encodes clinical events similarly to the models in [1] except that we introduce temporal tokens for larger time periods to the event's vocabulary to capture length of time. The proposed model will also evaluate static patient data and combine the outputs of the static and temporal parts in a weighted average akin to a multiple classifier system.

The resources of this research consist of two main entities: Lifespan Corporation and Brown Center for Biomedical Informatics (BCBI).

2 Related Work

Pre-processing temporal EHR data

An important aspect of creating predictive models is manipulating the data into features that a model can receive. This pre-processing is one of the challenges of machine learning with EHR data. Discretizing temporal data can be difficult due to the variability of time stamps on clinical events. If the discrete time interval is too short, the sequence will be mostly some padding, but if the time interval is too long, more information would be lost. Three approaches to solve this issue have been to create a temporal matrix[2], to tokenize multiple time intervals[1], and to create (clinical event, timestamp) tuples[3]. Each of these approaches tokenize a clinical event and create a vocabulary with the various events like diagnosis assignment, procedures, or medication prescriptions.

The temporal event matrix approach from [2] first creates a sequence of clinical events for a patient at a specific timestamp (or in a range of times). The matrix is then populated with the

[1]according to NIH `https://www.drugabuse.gov/related-topics/trends-statistics/overdose-death-rates`

sequences of the patient where there is a time dimension and clinical sequence dimension. The sequences are grouped appropriately through a fusion algorithm using a CNN. The benefit of this structure is that patterns are then more easily detected by spatial algorithms like CNNs.

The multiple time interval token approach from [1] again creates sequences from a patient's hospital admission. Instead of grouping these by timestamp, there are several timestamp tokens that are added in to represent time with no events. They have tokens for 0-1, 1-3, and 6-12 minutes. These tokens are added into the sequence like:

[dx1, pr4, 1-3m, pr2]

which would represent that the patient was diagnosed with dx1 and had pr4 performed at the same or a similar timestamp, had a 1-3 minute period with no event, and then had pr2 performed.

The tuple approach from [3] takes a clinical event and its timestamp and pairs them together. These pairs are then sequenced for a specific patient to be used in an RNN.

Multiple Classifier Systems

Multiple classifier systems are classifiers that take the output of multiple models and combine them to create a final output. The difficulty with these systems is how to appropriately combine the outputs so that the final classifier is most accurate. There are three different levels of output that each classifier could give[4]:

- **Type I:** Each classifier outputs the single most likely class that it predicts from the input.
- **Type II:** Each classifier returns the rankings of most likely to least likely classes.
- **Type III:** Each classifier returns the confidence measurement of each class.

Since we are dealing with a two class problem and using a two classifier system, we believe that it is most important to focus on Type III. Xu et al.[4] give several techniques to combine confidence scores of different models. The first suggested techniques is to simply take the average of the confidence scores and use the new scores to classify the input. The second technique uses the median of all of the confidence scores which would not work in our two classifier system. The third technique is to come up with some distance classifier and use this to calculate the highest confidence score among the classes.

Neural architectures with EHR data

There are several neural architectures that have been created for EHR data. We will discuss two basic architectures since these are models that use similar types of data as we are. These two architectures are a CNN and an RNN model.

The CNN model from [1] takes in the sequences the clinical events as tokens with additional time lapse tokens added in as discussed above. To avoid high-dimensional sparse weight matrices with one-hot encoding, they create an embedding layer akin to a word embedding layer. The embedding layer is a dense representation of each of the clinical events. The the sequence then goes through a convolutional layer to a pooling layer. The pooling layer accumulates the global information from the convolutional layer and feeds this information into a classifier. The classifier then outputs the predicted class.

The RNN model from [3] first creates the (event, time) tuples. After these tuples are sequenced, they are fed into a Gated Recurrent Unit (GRU)[5] as its recurrent cell. The output of the GRU is then sent into a linear classifier in the form of multiple linear layers and a softmax. The output of the softmax can be used to predict the class.

3 Data Collection

3.1 Accessing real patient data

Our process to work with the data comes in three steps: data collection, deidentification, and organization. First, we need to work with Lifespan to obtain a file with the health records of patients that fit our search criteria. Second, while still on Lifespan servers, we deidentify the patient data. We need to remove all information that can be used to identify a patient like medical record numbers, names, social security numbers, etc. However, the deidentified data still needs to group different medications and encounters of the same patient together, so for this we use a hash map to give each patient and encounter a unique, unidentifiable identification. Lastly, we move the deidentified data to a BCBI server and organize the data in a MySQL database. We move the data to a BCBI server to give us more flexibility and control over the packages that we can install and use for our models.

The process for working with PHI, especially large amounts of PHI, is challenging given the necessary security measures. In order to create accurate machine learning models, we require a significant amount of data on the order of thousands of longitudinal health records. Unfortunately, this is the first project that Lifespan has been a part of that requires this much data and so no current process is in place to deal with a data pull this large. We have been working with Lifespan over the past nine months to obtain this file and deidentify it, but there has been continuous push back and at the time of writing this, we still do not have full access to the data. There were many iterations of going back and forth trying to request an appropriate amount of data. We started with a request of $\sim 385,000$ health records and ended with a random sample of 500 records.

Given that the hunt to obtain real patient data has lasted over the past nine months, we started creating fake patient datasets to create data pipelines for when we have access to patient data. We have taken these fake data generators and looked at various statistics from static reports to try to accurately create random patients. Obviously, this new dataset will not be able to replace real data, but it should help us in creating and testing our models. For the rest of this paper, we will call the real patient data **LifeSet** and the fake generated data **DoveSet**.

3.2 Generating fake patient data

When creating a fake dataset, the first step is to create data that on the surface can mimic the real data. In order to mimic the LifeSet we need to evaluate which EHR fields we will pull from Lifespan. The determination of which fields/variables are necessary is a combination of clinical advice from the physicians that we are working with and scouring the available SQL database fields. However, before we can look through the SQL fields, it is important to know the underlying structure of EHR data to make sure that the generated data will look like that of LifeSet. From researching this data structure, we have created a list of important feature of EHR data. These features *must* be present in DoveSet to be able to compare to LifeSet.

1. **MRNs**: Every patient is uniquely identified by their MRN (medical record number. This identifier will be used for all visits and documentation within an organization.

2. **CSNs**: Every encounter is uniquely identified by a CSN (contact serial number). There is a one-to-many mapping between MRNs and CSNs as each patient can have multiple encounters, but each encounter can only belong to one patient.

3. **Encounters**: Any time there is a change in a patients record, an encounter is created without exception. An encounter in terms of EHR data is therefore slightly different than the intuitive

idea of an encounter. Whether the patient comes in for an outpatient visit, comes in for an ED visit, calls the medical staff for a refill, or a physician documents something on a patient's chart without seeing her/him, an encounter is created. This greatly simplifies data collection since now we only need to look at all of a patient's encounters and we can get the entire longitudinal history.

4. **Diagnoses**: Diagnoses are represented by universal ICD-10 codes.[2] ICD-10 codes are part of hierarchical database where each level of the code gives more specific information. For example, the code F11.2 refers to opioid dependency where all codes that begin with F11 more generally refer to mental and behavioural disorders due to opioid use. Even more generally, all codes starting with F1 are mental and behavioural disorders due to psychoactive substance use and anything beginning with F is a mental or behavioural disorder.

5. **Medications**: Medications are represented by RxNorm codes.[3] RxNorm codes are organized in a relational database, however, they are not as intuitive as the ICD-10 codes since each medication has much more information and differing features. RxNorm codes are organized in a SQL database that can be queried to return different groupings of features like ingredients, medication type (opioid, benzodiazepine), or medication purpose (analgesic, sedative).

The fields in DoveSet mimic pulls from several different SQL tables such as the tables for the overall patient data and encounters. If this data was pulled from SQL, we would need to capture these as separate CSV files for each table. Therefore, the code to write DoveSet will output five CSV files such that:

- There is a file for each table (Patient, Encounter, Diagnosis, Procedure, Medication)
- Each file has a patient MRN and each file excluding the Patient file has a CSN. With this, we do not need to save every timestamp in every table since we can always refer to the Encounter file for these.
- Timestamps within an encounter will all be simplified to the timestamp of the encounter, Since we are dealing with long term data, the ten minute difference between two procedures will not show up in our sequencing anyhow.

3.2.1 Adding real statistics to DoveSet

The code to write these files will not only need to mimic the LifeSet fields' datatypes, but it should also mimic some of the trends in the data. We may not know all of the trends since we only have access to some static reports, but we can add some additional artificial trends to DoveSet to test if the model can learn on this type of data. This subsection will focus on some of the trends that we found. Please note that these numbers are mostly approximations of the actual distributions for ease of computation.

First, we will discuss some demographic model that could effect both models. We will use the probability of each patient's sex to be 50/50 for male and female. For the age distribution, we will use a simplified United States age distribution[4]. For DoveSet, we will use three buckets with an even probability to be any age within that bucket. The three buckets are 0-14 years (20%), 15-64 years (65%) and 65+ years (15%).

[2]these codes are maintained by the World Health Organization (WHO)

[3]these codes are maintained by UMLS, a subsidiary of NIH

[4]This distribution can be found at `https://www.statista.com/statistics/270000/age-distribution-in-the-united-states/`

ED Recidivism: When we look at all encounters in the ED, we find that $\sim 20\%$ of all of these patients return to the Ed within 30 days. There is an exponential decay from patients that return that same day or the next day to patients not returning until 29 or 30 days have passed. Where this is not necessarily a feature that we will explore for our model, it is important to note that there are many patients that are 'repeat offenders' meaning that the same people return to the ED every day or every other day.

In terms of static demographic data, the two trends that stand out the most are in age and sex. When a patient is over the age of 22 years old, there is a fairly consistent 22% 30-day ED return rate. From ages 2 to 22, there is a linear increase in the return rate from 10% to 22% and before 2 years old, the rate is $\sim 15\%$. Interestingly, there is also a discrepancy in patient's sex with 19% of female patients and 22% of male patients returning to the ED.

To include these trends into the dataset, we will calculate the probabilities of each of the return to the ED occurring given age and sex. We will use Bayes theorem given these two independent conditions. R, A, and S are the return event, age variable, and sex variable, respectively. The probability that there is a return given the randomly assigned demographic information is:

$$P(R|A,S) = \frac{P(A,S|R)P(R)}{P(A,S)} = \frac{P(A|R)P(S|R)P(R)}{P(A,S)}$$

$$\text{since } P(A|R) = \frac{P(R|A)P(A)}{P(R)} \text{ , } P(S|R) = \frac{P(R|S)P(S)}{P(R)} \text{ , and } P(A,S) = P(A)P(S)$$

$$P(R|A,S) = \frac{\frac{P(R|A)P(A)}{P(R)}\frac{P(R|S)P(S)}{P(R)}P(R)}{P(A)P(S)} = \boxed{\frac{P(R|S)P(R|A)}{P(R)}}$$

The result gives us a clean probability that we can easily calculate with the above information from our static reports.

Opioid Overdose: When looking at clinical events like opioid overdose, there is some difficulty in labeling these patients appropriately from EHR data. Currently, we use the diagnosis codes between T40.0 to T40.4 as these are poisoning by different variations of opioids. In our reports, we see that of the $385,000$ patients only $1,200$ patients have an encounter with one of these codes. This translates to an extreme class imbalance where a positive label is only given to $\sim 0.3\%$. However, since we are using EHR data, we can only see if a patient overdoses *and* goes to the hospital or ED. If the patient does not seek treatment, then we cannot tell if the patient overdoses or not. As part of this project, we aim to look at RI state death data to help label additional patients.

We also investigate some additional predictors. Similarly to the ED recidivism problem, we find that there are trends in both age and sex for opioid overdose. Of the $1,200$ overdose patients, 70% were male to only 30% female. The age distribution is much less balanced than ED recidivism. For patients from 0-18 years, the number of overdoses recorded is negligible and we will assign this 0.01%. For patients from 19-40 years, the probability jumps to 0.5%. For 41-64 and 65+ years, the numbers fall back to 0.2% and 0.1%, respectively. We can use the same computed probability as we found above to label our data with one tweak since we have $P(S|R)$ and not $P(R|S)$ for this problem. This substitution would give us:

$$P(R|A,S) = \frac{P(S|R)P(R|A)}{P(S)}$$

The static reports unfortunately do not give us much data on the dynamic clinical events that coincide with our labels, however, we add some predictors that seem plausible like patients with an F11 (mental and behavioural disorders due to opioid use) are more likely to have an overdose. We add these predictors not to say that we know that these are trends in the data, but to prove that if these trends exist, our model can identify them.

4 Model Architecture

For these experiments, we create a model that can use both demographic data and other static data points in conjunction with longitudinal, dynamic data. To accomplish the use of both types of data, we use a multiple classifier system with two classifiers. The first classifier is an ensemble random forest classifier that will be used for the static data. We will call this the static classifier. The second classifier is a recurrent neural network (RNN) for the dynamic data. We will call this the dynamic classifier. Figure 1 gives a graphical depiction of the model.

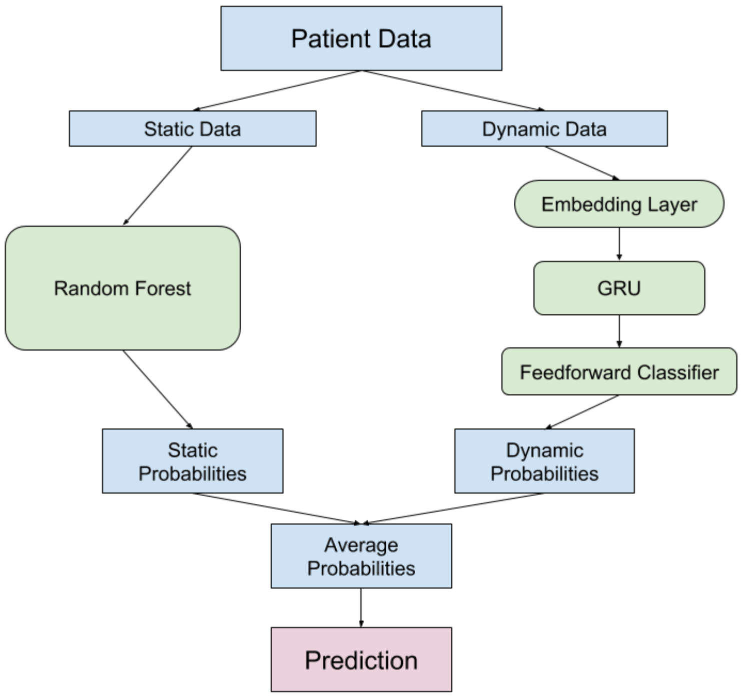

Figure 1: This figure is a graphical depiction of the model used in this paper. The static classifier using the static data is on the left where the dynamic classifier is on the right. The outputs are then averaged to give a final prediction.

4.1 Static Classifier

The static classifier uses both demographic data and cumulative overall data. Age, sex, and race are straight forward to include into the model as their own feature space. We create other features by aggregating the patients dynamic data at any given point in the patient's history. If we are training on a patient's visit in February of 2017, we will only create features based on previous encounters. Some of these aggregated features include the number of previous visits, number of previous visits in the past year, and the number of occurrences for each diagnosis, procedure, or medication bucket. The output of this classifier will be the class probabilities of each sample to act as a confidence score.

4.2 Dynamic Classifier

The RNN used for the dynamic classifier is composed of an embedding layer, a GRU RNN cell, and a basic feed forward linear classifier. The input to the dynamic classifier is an array of clinical event tokens. These tokens represent diagnosis, procedure, and medication buckets and lengths of time between events. We have decided to use time intervals of 1 month, 3 months, and 1 year.

The inputs are sent through an embedding layer to give each token a more robust definition. We initially have the embedding dimension size set to 10, but we plan to revisit this size with the real LifeSet. The embedded input is sent to the GRU cell where we will take the final hidden state of the GRU as our output. We take the final state since this state represents the entire sequence which in our case is the patient's medical history up to the current encounter. This hidden state is then given to a linear neural classifier of two linear layers separated by the RelU activation function. We then use the cross-entropy loss function to calculate our loss.

The output of the classifier when testing will be the softmax of the output of the linear layers so that we can use this as a confidence score to compare with our static classifier.

4.3 Combining the Classifiers

Since both the static and the dynamic classifiers output a confidence score, we can just take the average of these confidence scores to make a prediction. With additional testing with LifeSet we will be able to determine if we need to instead use a weighted average.

5 Experiments and Results

For this model we ran two experiments. The first tests the ability to predict ED recidivism and the second tests the ability to predict opioid overdose. For both of these experiments, we will use the available DoveSet for training and testing purposes. We use the same data for each experiment with different labels. For ED recidivism we label each encounter that has a return visit within 30 days as a positive sample and every encounter as a negative sample. With opioid overdose, we look for an encounter with a diagnosis code that begins with 'T40'. If a patient has one of these codes, then every previous encounter within the past year is labeled as a positive sample.

In order to test our approach of having separate classifiers for both static and dynamic data, we test the accuracy of each classifier on its own and both classifiers averaged together. The results on DoveSet can be found in tables 1 and 2 for our ED recidivism and opioid overdose tasks, respectively.

The results of these initial experiments on DoveSet show that the discrete EHR data can be modeled in both a static and demographic method and a dynamic and temporal method. These experiments do not show that possible predictive power since we are using a synthetic dataset with very few predictors surrounded by noise and useless data points. We believe that with LifeSet, we will find that many more clinical events and demographics will act as additional predictors.

6 Conclusion

In this paper, we look at predicting two different clinical events: 30-day ED recidivism and 1-year opioid overdose. We propose a model that looks at both the static and the dynamic patient data

	Static	Dynamic	Combined
Precision Score	0.274	0.235	0.247
Recall Score	0.033	1.0	0.095
F1 Score	0.059	0.380	0.137
Accuracy	75.3%	23.43%	72.03%

Table 1: The metric scores of the static and dynamic classifiers by themselves and combined for our **ED recidivism** task

	Static	Dynamic	Combined
Precision Score	0.166	0.032	0.333
Recall Score	0.022	0.067	0.044
F1 Score	0.039	0.043	0.078
Accuracy	99.76%	99.34%	99.77%

Table 2: The metric scores of the static and dynamic classifiers by themselves and combined for our **opioid overdose** task.

that is available in an EHR. We were unable to gain access to the data that we initially thought that we would have, so we have created a new dataset called DoveSet. The results of this paper show that it is possible to model these clinical events using discrete EHR data, but without real data, we cannot tell how well these models could work. We hope to continue on this project once LifeSet is available in the future. We also hope that with LifeSet we can extend this model to look at predicting other clinical events such as alcohol abuse or mental health diagnostics.

All code for this project can be found at `https://github.com/stephenrdove/clinic_predict`. The creation of DoveSet is located in all files preceded by 'fake_' and the other files contain the necessary processing for the models. Running main.py assumes that there is a pre-processed CSV file in folder 'data/'.

Acknowledgment

We want to thank all of those that helped with the project which includes (but is not limited to) Dr. Megan Ranney of the Brown Medical School; Paul Stey, Ashley Lee, and Mary McGrath of BCBI; John Brister and Dennis Ferrante of UEMF; and Kristen Morgan and John Patena of Lifespan.

References

[1] P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh (2017). Deepr: A Convolutional Net for Medical Records *IEEE J. Biomed. Health Informat., vol. 21*, 1, 22-30.

[2] Y. Cheng, F. Wang, P. Zhang, H. Xu, and J. Hu (2015). Risk Prediction with Electronic Health Records: A Deep Learning Approach. *SIAM International Conference on Data Mining SDM*, 16.

[3] E. Choi, M. T. Bahadori, and J. Sun (2016). Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. *Proceedings of Machine Learning for Healthcare*, 1–12, 2015.

[4] L. Xu, A. Krzyzak, and C. Y. Suen (1992). Methods for combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on System, Man, and Cybernetics*, 23, 418–435.

[5] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *In Proceedings of the Empiricial Methods in Natural Language Processing.*