Deep Learning on Clinical Notes

Adam Lieberman, Ravish Chawla, & Garrett Mallory

March 5, 2017

Abstract

When a patient enters a clinical setting, notes are written down to help better diagnose a patient based off their symptoms, lab results, and surgical notes. Diagnoses are not always correct and in a clinical setting, an increase in confidence of a diagnosis is always welcome. We aim to create a system that leverages large patient-based datasets to confirm that a doctor prescribed diagnosis, in the form of an ICD-9 code, is accurately based upon clinical observations. This system delivers a few main advantages: (1) Our algorithm can review thousands of clinical notes and diagnose a patient before a doctor can review and diagnose a single patient (2) Our model can constantly take in new data and be updated to achieve more accurate diagnoses (3) The system can serve as a second opinion for a doctor who has a diagnosis in mind.

In this proposal, we discuss the characteristics of the data we aim to examine, the processing pipeline for this data, an overview of some features we plan to engineer, deep learning algorithms we plan to explore, architecture for our system, and a cost-benefit analysis of the end product.

1 High-level Approach

We are creating a deep learning system called Dr. ANN (Artificial Neural Network) that will take in a clinical note and output a predicted diagnosis in the form on an ICD-9 code. Throughout the development process we will explore a variety of feature representations such as bag-of-words, word-to-vec, and tfidf. We will explore deep learning algorithms such as artificial neural networks, end-to-end memory networks, and long short-term memory. We aim to use the power of deep learning to accurately recommend diagnoses based off of a patient's clinical note.

2 Today's Diagnosis

Today, doctors are manually diagnosing patients. Handwritten/digital clinical notes are being observed and doctors are using their best judgment to accurately diagnose each patient. Doctors are specialized and hence have great knowledge of the medical field at their disposal, yet there are limits to how much data a human can process. There has been a great deal of research on using deep learning, particularly on recurrent neural networks (RNN) and long short-term memory (LSTM), to diagnose patients, but no system has been deployed in a hospital or doctor's office. This is partly due to machine error. Computers are not perfect and do make mistakes. Diagnosing a patient can be a life-threatening event and to put this fate in a computer's hands can be quite dangerous. However, a computer trained to recommend diagnoses like a doctor can serve as a statistically significant second opinion for the doctor. The deep learning algorithm could potentially discover diagnoses the doctor never considered and could serve as a great point of reference to assist the doctor in diagnosing patients.

3 Our Approach

We aim to use deep learning to essentially train a computer system to make recommendations of diagnoses to the same degree that a doctor can. We are training a deep network to think like a doctor. With or system doctors will be able to input their clinical notes and the note will be run through our deep learning algorithm. A predicted ICD-9 code will then be returned to the doctor to help assist in diagnosing a patient.

Additionally, our system can predict thousands of diagnosis recommendations before a doctor can read a clinical note and make one recommendation. This can save the doctor time as calling alternate doctors for a few second opinions is a slow process. Our system will allow doctors to confirm or validate their hypothesis for a given patient. Additionally, a patient could use our system to self-diagnose themselves prior to visiting a doctor's office. They could then research their diagnosis and have a better perspective on their current condition.

4 Risks & Rewards

There are associated risks with what we are aiming to achieve. Deep learning has shown to provide great learning results, but there is a chance that our models perform poorly. This could be a result from not having enough data or not finding the optimal parameters for our model. Neural networks are notoriously difficult to evaluate and machine learning in general is not always understood as a trusted tool. This would render our product useless to doctors and patients as it would not provide accurate diagnoses. However, if we are successful we could greatly assist doctors in providing diagnosis recommendations. This system could be utilized in many hospitals and intensive care units and could potentially accurately diagnose thousands of patients in seconds. Additionally, the system has the potential to continuously update to the latest medical standards given a continual inflow of patient records during its deployment.

5 Data

We will be utilizing data from MIMIC-III, a large publicly-available database comprising of de-identified health-related data associated with approximately 60,000 patients from the critical care units of Beth Israel Deaconess Medical Center from 2001 to 2012. The database is comprised of static and dynamic data ranging from birth dates to demographics to laboratory test results to diagnoses to imaging reports to clinical notes. For our purposes, we will be analyzing clinical notes and predicting associated diagnoses. Thus, we will particularly make use of the following tables:

- diagnoses_icd
- noteevents

The diagnoses_icd table contains patient diagnoses. The noteevents contains the following type of clinical notes:

- **Discharge summary** - A summary written by the physician (or possibly a team of physicians) at the time of discharge from the ICU.
- **Radiology reports** - Reports from imaging procedures such as MRI, CT, and Ultrasounds.
- **Nursing progress reports** - Daily notes from the nursing staff.

For training, we will use all three types of notes, grouped by individual patients. Along with these notes, we will use other features such as the diagnostic date, lab result values, and other information that can be extracted. The data will be cleaned and these features will be fed into our neural network, which will retrieve the best set of features. We will also handle cases in which a patient may have notes from multiple visits, by grouping the events together to obtain a result based on all data available for the user within the observation window.

6 Feature Construction

To pass our clinical text data into our deep learning algorithm we will need to construct features. We will explore the following features:

- **Bag-of-words**: The bag of words model is a representation used in natural language processing. With this model, a text such as a sentence or document is represented as a bag of its words where grammar and word order are disregarded. After the text is transformed into the bag of words, we can calculate

various measures to characterize the text. For instance, we can look at the term frequency. This is the number of times a term appears in our text. We can construct this text vector and use this feature in our deep learning algorithms.

- **Word2vec** - We can build word projections in a latent space of N dimensions where N is the size of the word vectors obtained. The vector will contain float values that represent the coordinates of the words in this N dimensional space. This allows us to view our text vector in a different and continuous dimension space, which may attribute to different and more interesting calculus characteristics.

- **Term Frequency-Inverse Document Frequency (TFIDF)** - TFIDF allows us to reflect how important a word is to a document in a corpus. Some words are very frequent like "the", "a", "is", but carry very little meaningful information about the actual content of the document. If we were to look at a frequency count in a document that has many of these non-meaningful words our model might perform worse as it is not giving importance to the rarer yet more interesting terms. We can re-weight the count features and use this feature vector in our deep learning algorithm.

- **UMLS** - MetaMap is a module that maps tokenized words to UMLS concepts. For instance, a clinical note will have many medical concepts that the prescribing doctor has written. MetaMap will be able to extract medical concepts from the note, such as SNOMED-CT codes, CPT codes, and other patient ontology. MetaMap goes further by providing affected organs with the associated symptoms. These concepts can be more useful than a TfIDf mapping because UMLS is will remove more extraneous information from clinical text than just stop words, and will place a stronger emphasis on medical words.

7 Deep Learning Architecture

After obtaining the preprocessed features from the clinical notes, we will train a neural network in order to find a model that can best predict diagnostic codes. Input to the neural network will consist of separated words from the clinical notes, and features generated from TfIDF. Instead of predicting ICD-9 codes to the full precision, our model can be trained to predict only up to first 3 digits, limiting the number of classes substantially. Higher precision codes will be predicted using a separate model. We will be using this approach because the large amount of ICD-9 codes may limit the number of training examples the model will have available to train on for a specific class. For instance, there might be few patients with ICD-9 code 250.6, Diabetes with neurological manifestations. However, the number of patients with Diabetes, and an ICD-9 code of type 250.*, may be much higher. We are planning to analyze and research the following deep learning models:

- **Basic Artificial Neutral Network** - A model consisting of layers of nodes called neurons that where each neuron has an excitatory or inhibitory on its neighbors in the next layer. The activation of the last layer determines the classification of the example.

- **Attention Recurrent Neural Network** - An enhancement over traditional neural nets that weights subsets of features in order to return a classification as well as highlight the features that had the greatest impact in generating this classification.

- **End to End Memory Network** - A model based on the same architecture of memory networks, but trained end-to-end over input samples. This training requires less supervision during training and is therefore more robust to real-world applications.

- **Long Short-Term Memory** - A version of a RNN that is optimized to process inputs over a time series. LSTM units are adept at handling natural language as they are well suited to relating words and word groups to each other despite being separated by irregular time steps.

Training a deep learning model on a large dataset can be computationally expensive so we will throughly research each model and implement one of the above. Our current research leads us to believe that an attention recurrent neural network will best suit our needs. With regards to which model we choose to implement, we will explore different layers and model parameters to finely tune our model.

8 Performance Measurements

In order to train our neural network to find the best set of weights for each layer, we will be using a module that will allow us to use K - Fold cross validation on each epoch. The metric that will be used to validate the model at each iteration will be AUC, because it uses both true positive rate and false positive rate.

To evaluate our model, we will use different metrics, including F1 score, ROC/AUC curve, Cross Validation score, Cross Entropy value, lift between worst and best performing data subsets, and a confusion matrix visualization. We will set aside 10% of the dataset for testing, allowing us to see how our model performs on a previously unseen dataset.

9 Web Application

Once our deep learning algorithm has been finely tuned, it is ready for predictions. To make our system user friendly, we are creating a python-backed web application where doctors and patients can input their clinical record and view the model's predicted diagnoses. In our system, they will find information regarding the top 3 predicted diagnoses and sample clinical reports for each predicted diagnosis. This will allow them to evaluate and gain insight into the condition. The web application architecture is as follows:

- **Frontend** - The frontend will consist of HTML, CSS, JavaScript, and JQuery.
- **Backend** - The backend deep learning results and web scraping for ICD-9 code descriptions will be performed with python.
- **Microframework** - To connect the frontend and backend together we will use the Flask microframework. We will make use of Jinja2 to pass some backend Python data to the HTML template on the frontend so that it is viewable for users.

We have started the UI/UX process and have built the base of our frontend:

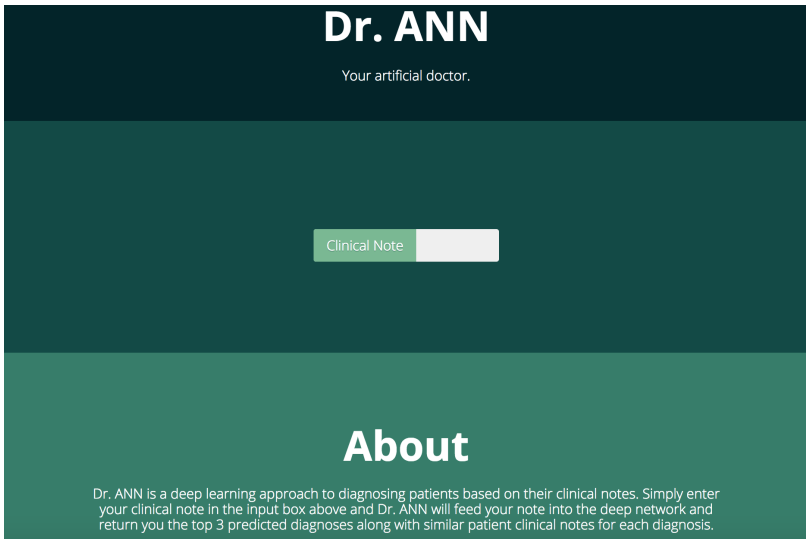

Figure 1: Homepage to enter in clinical note

The above image displays the homepage for Dr. ANN. Here, doctors and patients can enter in a clinical note in the input field and submit it to the deep learning algorithm for predictions.

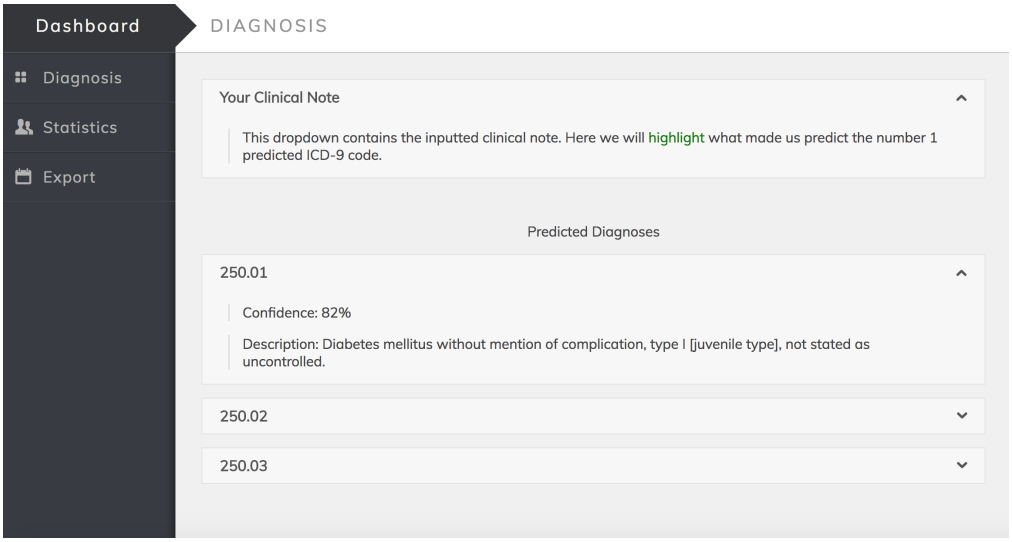

Figure 2: Diagnoses tab in user dashboard

Above we have the user dashboard on the diagnosis page. Here the user can see their clinical note. They will see certain words in their clinical note highlighted. These words are the words that contributed most significantly to the predicted ICD-9 code. Below, in the Patient Diagnosis tab, the user will find the top three predicted ICD-9 codes for the given clinical note. Additionally, they will find the confidence score for each note as well as a description about the ICD-9 code.

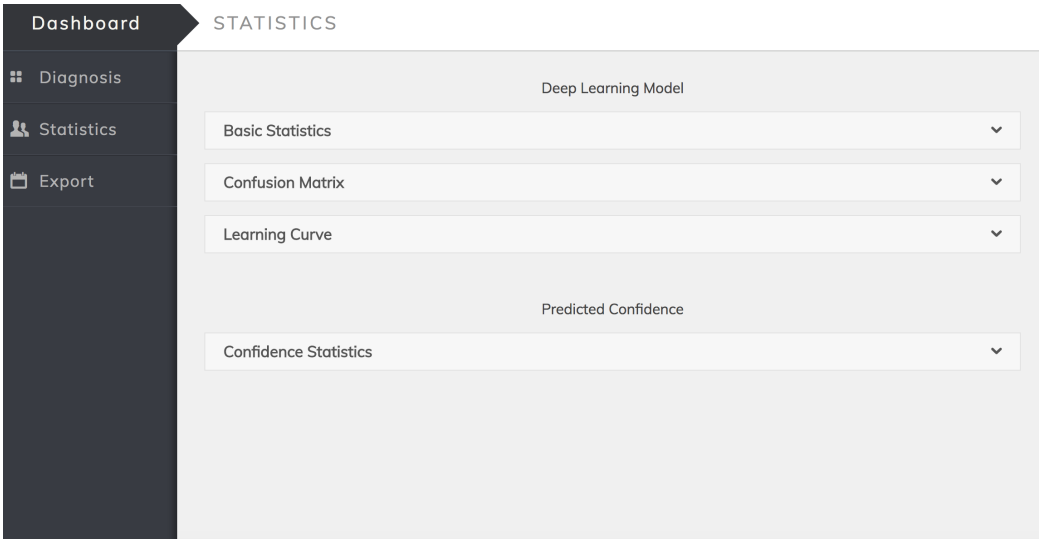

Figure 3: Statistics tab in user dashboard

Above we see that dashboard the statistics tab. Here you can find performance measurements about the trained model. Additionally, there will be some statistics about the predicted ICD-9 code.

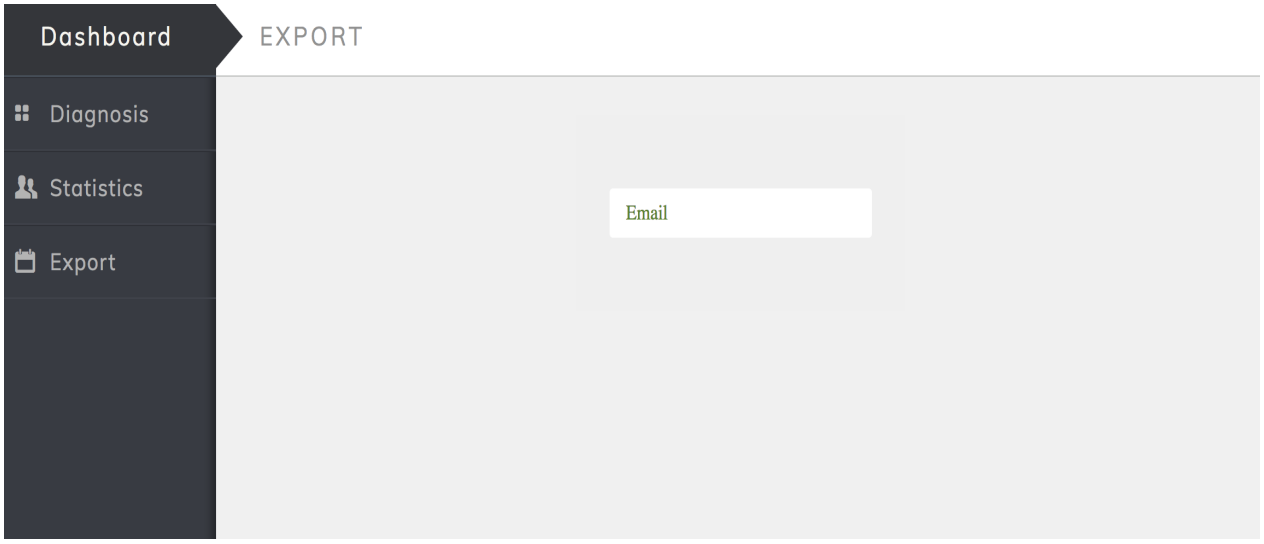

Figure 4: Email export tab in user dashboard

Above is the email export tab. Here a user can choose to email their results to a specified email address.

10 Pipeline

Below we depict the pipeline for our project:

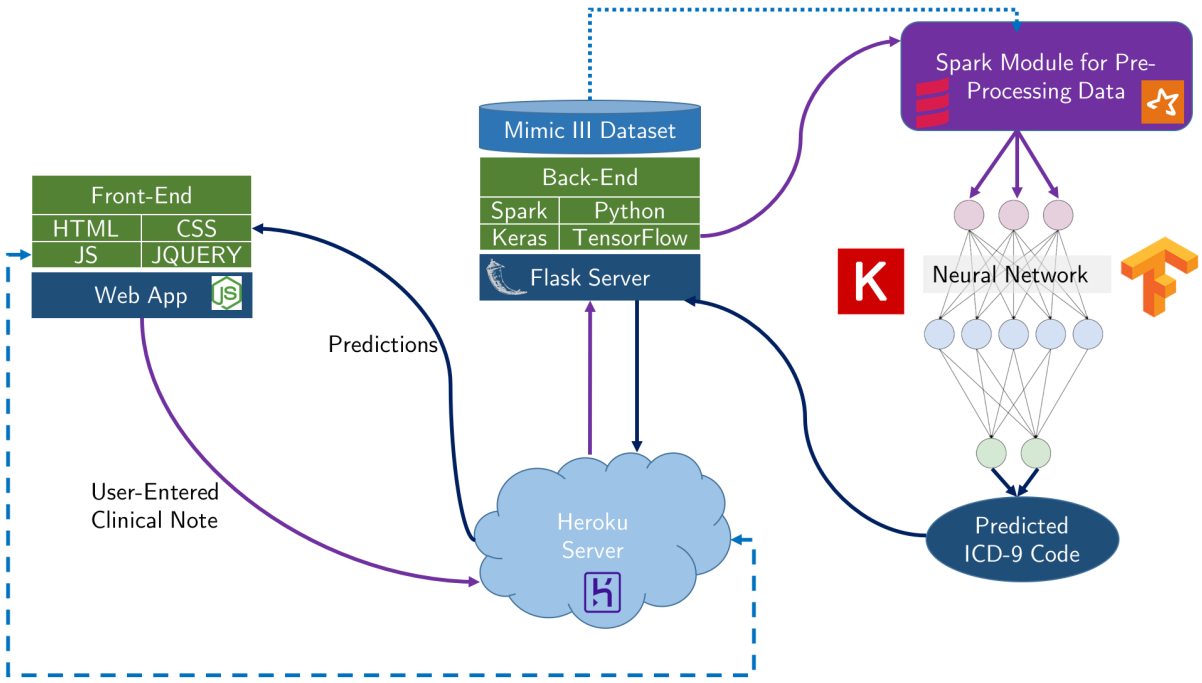

Figure 5: Architectural Overview of the system

Above we see that our frontend is built with HTML, CSS, JavaScript, and JQuery. The backend is built with Spark, Python, Keras, and TensorFlow. The backend takes in the Mimic III data and we use Spark

for the preprocessing of the data. Python, Keras, and TensorFlow are then used to build the deep learning algorithm, which predicts the ICD-9 codes from the given clinical notes. We will use the flask microframework to bridge the front and backend together and will display our web application on a Heroku server. Here, a user will be able to enter a clinical note into the application and this note will be sent to the deep learning model from the Heroku server to flask. The clinical note will then be processed and the model will make a prediction and return the information back to the user.

11 Cost

The table below displays the cost for each aspect of our project:

Task	Time Estimate	Rate	Cost
Frontend Application Development	12 hours	$45/hr	$540.00
Backend: Deep Learning Algorithm (Parameter Tuning, Model Exploration)	20 hours	$125/hr	$2500.00
Backend: Preprocessing	3 hours	$60/hr	$180.00
Backend: Feature Engineering	3 hours	$60/hr	$180.00
Backend: Data Scraping for Frontend Delivery	1 hour	$50/hr	$50.00
Resource: M4.XLarge AWS Instance	10 hours	$0.215/hr	$2.15
Resource: P2 AWS Instance	25 hours	$0.90/hr	$22.50
Total Estimate:	39 hours		**$3474.65**

Table 1: Labor Cost

We found that an average deep learning engineer charges $100-150 per hour, a frontend developer charges $40-50 per hour, and a data engineer charges $50-60 per hour. The hourly rates we set were averaged from the hourly rated we found for various freelance developers for the respective tasks. Additionally, deep learning is computationally expensive so we will let our programs run on Amazon Web Service (AWS) clusters. We examined the machines Amazon offers and chose the setup we find most useful for our project. In total, we expect our project to cost $3474.65 for development time and web services fees.

12 Timeline

Our proposed timeline is as follows:

Objectives	Tasks
Objective 1 (1-2 weeks)	Preprocess clinical notes (remove stop words, look at casing/ punctuation, etc.) Run neural net on subset of data. Mockup UI. Build web scraper.
Objective 2 (2-3 weeks)	Explore data features. Train neural net on all data. Evaluate the model using assorted metrics.
Objective 3 (2-3 weeks)	Publish a hosted frontend application that allows user input that can be run through our neural network, and display predicted diagnostic codes. Tuning for performance via adjustment of hidden layers and parameters.

Table 2: Project Development Timetable

The table is designed to capture the expected workload and delivery timeline for the application by breaking the project down into three separate subproblems. The total time estimated to accomplish objective 1 is roughly 10 hours split over 2 weeks. The object 2 and 3 will take roughly 15 hours apiece in development time in addition to the hours necessary to run the full algorithm training on the AWS machines. For this reason, 2-3 weeks is the expected turn around time.

13 Literature Survey

In order to create a more robust system, we have reviewed the following pre-existing literature.

- **Memory Networks by Jason Weston, Sumit Chopra, and Antoine Borde** - This paper introduces a variant of neural nets that leverages a memory bank for use in classification of temporally separated examples. The proposed advantage over a more traditional text processing neural net such as a recurrent neural net is that the memory component is more robust than an RNNs hidden layer network at making inferences on data separated by time or sampling rate.
- **Brundlefly at SemEval-2016 Task 12: Recurrent Neural Networks vs. Joint Inference for Clinical Temporal Information Extraction by Jason Alan Fries** - The use of a recurrent neural network on clinical notes to identify the span of time a patients record is active and the event expressions contained within the notes is relevant in that it may provide guidelines to capturing features from text notes beyond time word relations as well as guidelines on using RNNs on clinical notes.
- **Applying Deep Learning to ICD-9 Multi-label Classification from Medical Records by Priyanka Nigam** - This paper directly addresses the performance increases experienced by deep learning techniques over more traditional models such as logistic regression models. This provides a source of supporting evidence that deep learning is an appropriate approach to take to processing textual clinical notes in the hopes of making accurate classifications and predictions.
- **Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records by Riccardo Miotto, Li Li, Brian A. Kidd, and Joel T. Dudleya** - The approach taken in this paper is to use unsupervised deep learning towards preprocessing of clinical note data. The text structures discovered by taking an unsupervised approach allows a unique patient representation and for the outperformance of traditional machine learning models. The details of the approach taken to discovering semantic structures within clinical text are directly relevant to our project.
- **Structured Attention Networks, Kim Y, Denton C, Hoang L, Rush A** - This paper outlines the idea and structure of attention networks as a variation of neural networks. These networks are able to potentially learn unsupervised hidden representations inherent in the natural language inputs. We hope to use these networks to provide highlights to the user inputs to our application about which sections of the clinical note are more relevant to the neural net's classification.
- **Diagnosis code assignment: models and evaluation metrics, Perotte A, Pivovarov R, Nataranjan K, Weiskopf N, Wood F, Elhadad N** - Looking specifically at the discharge notes, this paper details the process of predicting ICD codes in either a flat manner (i.e. looking at the code as an independent label) or a hierarchical manner (i.e. considering that the first segmentation of the code provides different information than the second segment). The results of taking into account the hierarchical structure of the codes improved classifications results tremendously and is a motivating factor for why we ourselves are examining the first part of the code separately from the second.
- **A System for Predicting ICD-10-PCS Codes from Electronic Health Records, Subotin M, Davis AR** - Taking a slightly different approach, this paper looks at creating hierarchical levels of abstraction on the textual clinical note data. This gives a alternative approach that we can compare our results to as well as ideas in making meaningful features for our neural networks.

14 References

[1] Weston J, Chopra S, Bordes A. Memory networks. arXiv preprint arXiv:1410.3916. 2014 Oct 15.

[2] Fries JA. Brundlefly at SemEval-2016 Task 12: Recurrent neural networks vs. joint inference for clinical temporal information extraction. arXiv preprint arXiv:1606.01433. 2016 Jun 4.

[3] Nigam P. Applying Deep Learning to ICD-9 Multi-label Classification from Medical Records.

[4] Miotto R, Li L, Kidd BA, Dudley JT. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. Scientific reports. 2016;6.

[5] Kim Y, Denton C, Hoang L, Rush AM. Structured Attention Networks. arXiv preprint arXiv:1702.00887. 2017 Feb 3.

[6] Perotte A, Pivovarov R, Natarajan K, Weiskopf N, Wood F, Elhadad N. Diagnosis code assignment: models and evaluation metrics. Journal of the American Medical Informatics Association. 2014 Mar 1;21(2):231-7.

[7] Subotin M, Davis AR. A system for predicting ICD-10-PCS codes from electronic health records. InProc BioNLP 2014 Jun 27 (pp. 59-67).