Embedding Methods for EHR Data and the Utility of Clinical Text

Conor K. Corbin
Department of Biomedical Data Science
Stanford University
Stanford, CA 94305
ccorbin@stanford.edu

Gautam B. Machiraju
Department of Biomedical Data Science
Stanford University
Stanford, CA 94305
gmachi@stanford.edu

Abstract

Effective use of Electronic Health Record (EHR) data promises to reveal clinical insights and improve patient care. EHR data, however, is heterogeneous, noisy, largely unstructured, and temporally asynchronous. Eighty percent of EHR data comes in the form of unstructured clinical text. Developing predictive models that perform well across health intuitions remains a challenge. While patient data in and of itself is large (Stanford's Clinical Data Warehouse houses over 3M de-identified patient timelines), often the number of patients with a clinical outcome worth predicting (development of rare disease, mortality, clinical trial eligibility) is comparatively small. Inspired by recent NLP advancements leveraging word embeddings and language models for transfer learning, we develop three neural methods to pre-train contextualized patient representations. We then leverage these representations in two downstream clinical prediction tasks (1) inpatient mortality and (2) 30-day inpatient readmission. Further, we compare the performance of learned representations that are trained on data-streams comprised of structured data only, unstructured data only, and a combination of the two.

1 Introduction

Patient profiling through the use of electronic health record (EHR) and clinical data has been a hallmark of modern-day precision medicine. Effective use of EHR data promises to reveal clinical insights and improve patient care. With EHR data we can conduct large scale observational studies, predict patient outcomes, and provide clinical decision support. EHR data, however, is heterogeneous, largely unstructured, and temporally asynchronous. It is estimated that 80% of the time required to develop clinically relevant predictive models is devoted to data wrangling and manual feature engineering. Features engineered for a particular task often fail to generalize to new tasks, and models developed using data from one clinical institution often fail to perform well at another. Further, although we have access to large amounts of patient data (Stanford's Clinical Data Warehouse has over 3M de-identified patient records), many clinically useful machine learning tasks (screening rare disease, predicting mortality, pre-screening clinical trial eligibility) are low prevalence. Leveraging NLP embedding methods to pre-train patient representations has the potential to alleviate these concerns, and particularly boost predictive performance when sample size (n) is small.

Due to the inherent high dimensionality, noise, sparsity, and bias of EHR data, creating patient representation for subsequent machine learning prediction tasks, ranging from patient outcomes to ICU triage and mortality for preventative care, remains a challenge. Roughly 80% of EHR data comes in the form of unstructured clinical text. Making use of this data has traditionally been difficult, and to our knowledge there has been no attempt to quantify the added benefit of including this data in neural patient representations. Our contribution is thus two-fold. First we compare the predictive utility of various patient representation schemes, and second we compare how this utility changes as we ablate features stemming from unstructured clinical text.

2 Related Work

There have been several recent efforts in the biomedical informatics community to leverage deep learning techniques to automate the feature engineering process and improve performance on clinical prediction tasks. Miotto et al used stacked denoising autoencoders to create patient representations from structured and unstructured EHR data [1]. Rajkomar et al created patient representations with feed-forward and recurrent neural networks [2]. These representations were ensembled in downstream prediction tasks to improve upon baseline logistic regression classifiers. Choi et al developed a graph based attention model to generate patient representations that leverage information stored in biomedical ontologies [3], and Dubouis et al compared several embedding methods including bag of GLOVE vectors [4], and a recurrent neural net to create patient representations from clinical text [5]. We build off of [5] by 1) comparing performance of embedding methods fit on differing data streams from EHR data, and 2) constructing three new neural methods to create patient representations.

3 Approach

3.1 Baseline featurizers

Below, we describe two baseline methods for creating fixed length vector patient representations. Logistic regressions are trained on each baseline featurization scheme to predict both inpatient mortality and 30-day hospital readmission. Each featurizer is trained on train and validation sets.

Term Frequency Inverse Document Frequency (TFIDF) We featurize our structured data using TFIDF — a featurization scheme similar to bag-of-words, except that terms that are frequent in one document (patient history) but also frequent in many documents are not be weighted as much as terms frequent in one document and infrequent in others (i.e., the frequency of a code in a patient timeline is divided by the frequency of the code across patient timelines). The intuition here is that medical codes and terms that appear often in a particular patients history but not in the majority of patient histories carry more information than those that appear frequently across all patients. TFIDF representations were trained on codes and terms (extracted from free text notes) separately and concatenated to create fixed length patient representations.

LSA Here, we reduce the dimensionality of our TFIDF representations by running them through a Singular Value Decomposition (SVD) operation. This yields a patient feature matrix decomposed via SVD. We used the popular `gensim` LSA implementation to perform this operation.

3.2 Neural featurizers

The following featurizers include both shallow neural networks (Word2vec), as well as networks that operate on assumed sequential (i.e. autocorrelated) data in a language model regime (GRU and Transformer). For the latter class of networks, the model formulation is as follows: predict all codes and term mentions for the next patient day given the codes and term mentions for the current patient day. After training, patients are featurized with the output layers in our neural models to predict (1) inpatient mortality and (2) 30-day hospital readmission.

Word2vec Codes and terms in a single patient day are modelled as a "sentence", and randomly permuted [6]. The final learned patient representation is achieved by taking the maximum value across patients and days. We again used the popular `gensim` Word2vec implementation to pre-train code embeddings on our both the structured codes data and term mentions.

Gated Recurrent Unit (GRU) Using a modified implementation from an in-house codebase, we were able to deploy our data streams on a GRU [7] [8] [9]. Similar to an LSTM in its gated structure, a GRU was chosen due to its smaller parameter set. The proposed neural network is defined by the following mappings, given an input patient vector $\mathbf{x}_t$ at visit day $t = 1 \dots n-1$:

$$z_t = \sigma(W_z \mathbf{x}_t + U_z h_{t-1} + b_z) \quad \text{[update gate]} \quad (1)$$

$$r_t = \sigma(W_r \mathbf{x}_t + r_z h_{t-1} + b_r) \quad \text{[reset gate]} \quad (2)$$

$$h'_t = \tanh(W_h \mathbf{x}_t + (r_t \odot U_h h_{t-1}) + b_h) \quad \text{[current memory]} \quad (3)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \quad \text{[final memory]} \quad (4)$$

where $W_{\{\cdot\}}$s and $U_{\{\cdot\}}$s, and V_o are weight matrices and $b_{\{\cdot\}}$s are bias vectors. Note here that operator $\odot$ denotes the Hadamard product, or element-wise multiplication. The outputs for $\mathbf{x}_t$ for days $t = 1 \dots n-1$ is a predicted representation of the next corresponding days, $\hat{\mathbf{x}}_t$ for days $t = 2 \dots n$. This is depicted in Figure 1.

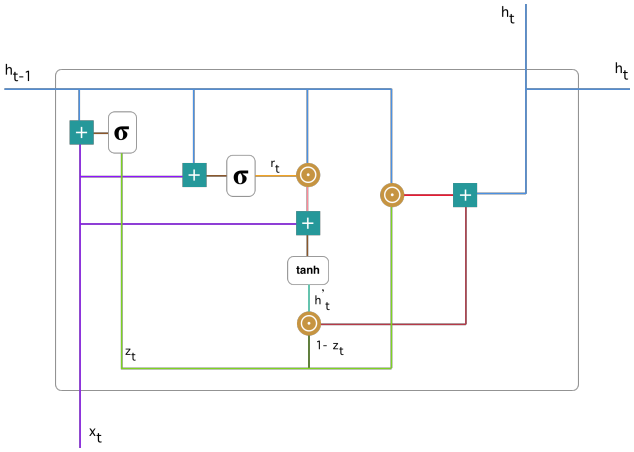

Figure 1: GRU architecture used for Language Modeling [10].

Transformer We also leverage a language model inspired by transformers [11] based on modified open-source implementations [12] [13] to create contextualized patient representations. To accomplish this, we solely use the decoder with multi-head attention, disregarding the encoder steps. The proposed neural network is defined by the following mappings, given an input patient vector $\mathbf{x}_t$ at visit day $t = 1 \dots n-1$:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK}{\sqrt{d_k}}\right) V \quad \text{[Scaled dot-product attention]} \quad (5)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \quad \text{[Multi-head attention]} \quad (6)$$

$$\text{SubOut} = \text{LayerNorm}(\mathbf{x}_t + \text{Sublayer}(\mathbf{x}_t)) \quad \text{[Sublayer output]} \quad (7)$$

$$\text{FFN}(\mathbf{x}_t) = \max(0, \mathbf{x}_t W_1 + b_1) W_2 + b_2 \quad \text{[Feed forward layer]} \quad (8)$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and $W_i^{\{\cdot\}}$ and W^O are weight matrices [11]. W_1, W_2 and b_1, b_2 are parameters and biases for the feed forward layer (FFN). Similar to the original architecture by Vaswani et al, our model assumes $N = 6$ layers with $h = 8$ heads (i.e. parallel attention layers). Like the GRU, the outputs for $\mathbf{x}_t$ for days $t = 1 \dots n-1$ is a predicted representation of the next corresponding days, $\hat{\mathbf{x}}_t$ for days $t = 2 \dots n$. This is depicted in Figure 2.

4 Experiments

4.1 Data

We utilize Stanford's Clinical Data Warehouse (STRIDE), which houses more than 3M de-identified patient medical timelines. These timelines include both structured codes data and 70M free text

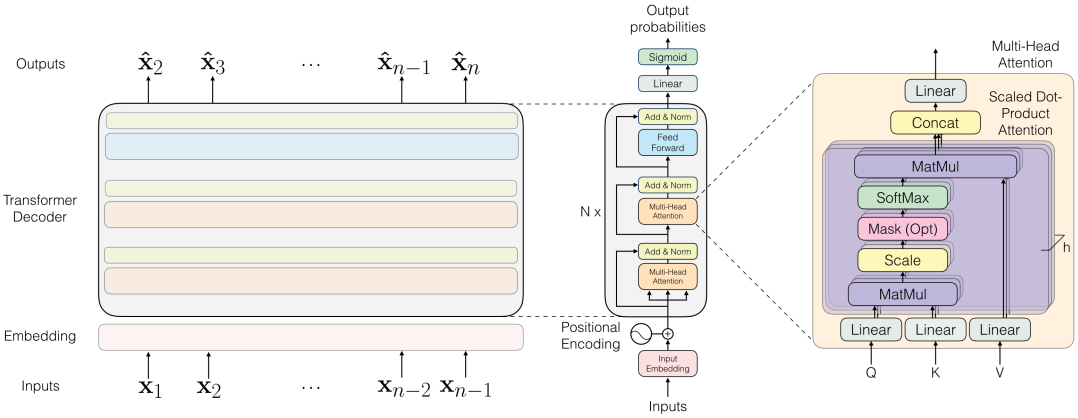

Figure 2: Transformer decoder architecture used for Language Modeling. Adapted from [11].

clinical notes. We have access to this data through Dr. Nigam Shah in the Department of Biomedical Data Science. For comparison, have developed learned patient representations trained on the following four sets of clinical data streams and multiple (baseline and neural) featurization schemes.

Codes Structured EHR data comes in the form of temporally spaced sequence of tokens/coded data (diagnosis codes, procedure codes, medications, lab orders) that together form a "patient timeline". In our dataset, the temporal resolution of our timeline is a patient day. The number of codes assigned to a patient vary from day to day, and the number of recorded days varies per patient.

Terms In this data stream, we leverage medical terms and concepts extracted from free text clinical notes. Medical terms are defined according to the Unified Medical Language System (UMLS) ontology library, and extracted via simple string parsing. These term mentions are grouped at patient day resolution.

Codes + Terms Here we merge the structured codes data stream with medical term mentions (aggregated at day-level). Like before, tokens within a patient day are shuffled.

Terms - Relations In this data-stream we hope to elucidate the importance of medical term context within the clinical notes from which they originate. Relation extraction tools such as `NegEx` and `ConText` are used to detect negated terms (ex. “patient denies cough”) and terms that refer to family history (ex. “patient’s mother has history of breast cancer”). Like in the third data-stream, term mentions are combined with structured data. Here relations are ablated, so that we can quantify their predictive power in downstream tasks.

4.2 Time splits & training regime

Time splits separate our data into train, validation, and test sets. To accomplish this, we defined our training set to span `2010-01-01` to `2015-12-31`, the validation set to span `2015-12-31` to `2016-06-30`, and the test set to span `2016-06-30` to `2016-12-31`. The LM trained on a training set of 750K patients on the Stanford Nero server’s 8 GPUs in 10 hours.

4.3 Evaluation

We evaluate the utility of our patient representations using predictive performance measures on two benchmark clinical tasks: inpatient mortality and 30-day hospital readmission. In particular, we will compare AUROC achieved with our neural model against the above baseline models using four data streams - coded data, medical term mentions, coded data with medical term mentions, and coded data with medical term mentions that include negation and family history detection. In the following subsections, we evaluate the aforementioned baselines against the neural methods (Transformer and GRU).

4.4 Prediction task: inpatient mortality

Patient labels are generated with the `stride_ml` package (an in-house package of the Shah Lab) and its built-in `inpatient_mortality` labeller. For our initial experiments we use only a subset of our labels, subsampling training and validation sets while not doing so for the test set. Accordingly, hyperparameters are tuned via the validation set. The breakdown of our subset's labels includes 110 positive cases and 5,338 negative cases for our training and validation sets, as well as 283 positive cases and 16,904 negative cases for our test set.

For our downstream prediction task, a logistic regression classifier is trained for each of the featurizers (which are trained on four datastreams: codes, terms, codes + terms, terms - relations). Results outlining AUROC performance is shown in Figure 3.

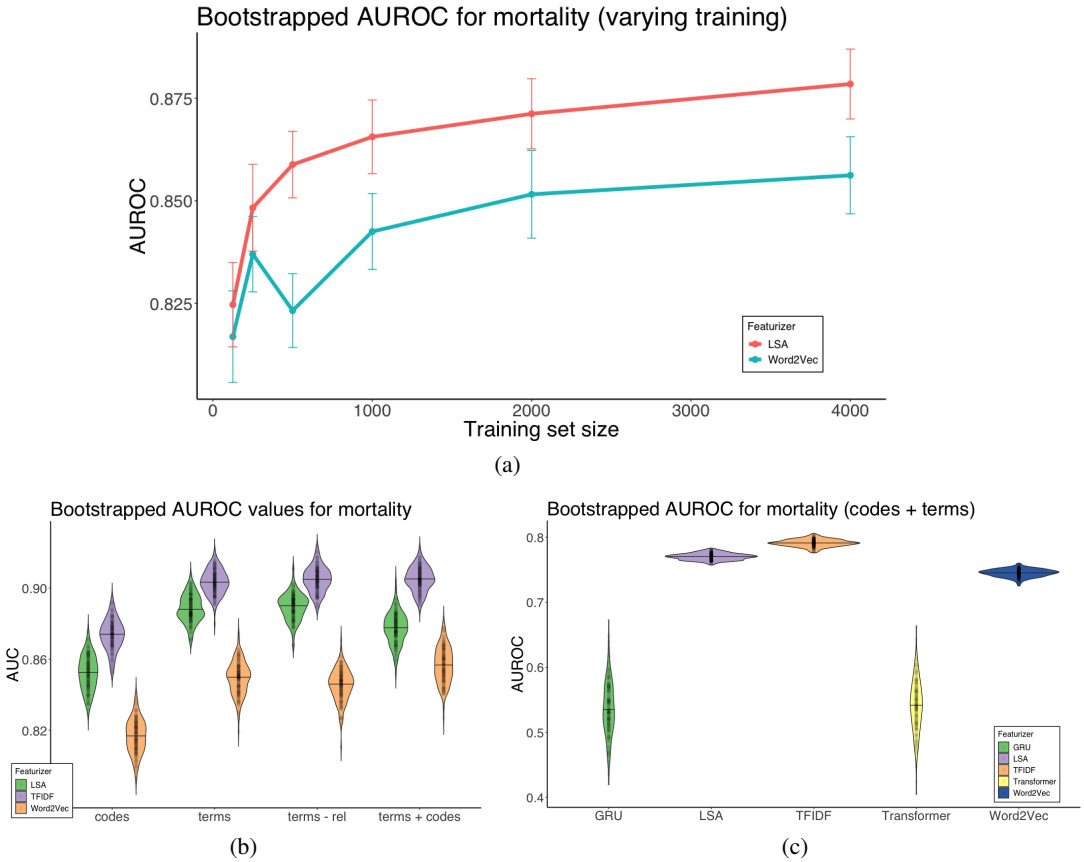

Figure 3: Inpatient mortality AUROC examined by training set size(3a), data stream (3b), and featurizer (3c).

4.5 Prediction task: inpatient readmission

Similarly in the task of predicting inpatient mortality, patient labels are generated with the `stride_ml` package and its built-in `inpatient_readmission` labeller. We again downsample to a subset of patients for our initial experiments, yet again sampling the training and validation sets, but not the test set. Hyperparameters are tuned via the validation set. The breakdown of our subset's labels includes 716 positive cases and 4,194 negative cases for our training and validation sets, as well as 2,168 positive cases and 8,822 negative cases for our test set.

For our downstream prediction task, a logistic regression classifier is trained for each of the featurizers (which are trained on the four same datastreams). Results outlining AUROC performance is shown in Figure 4.

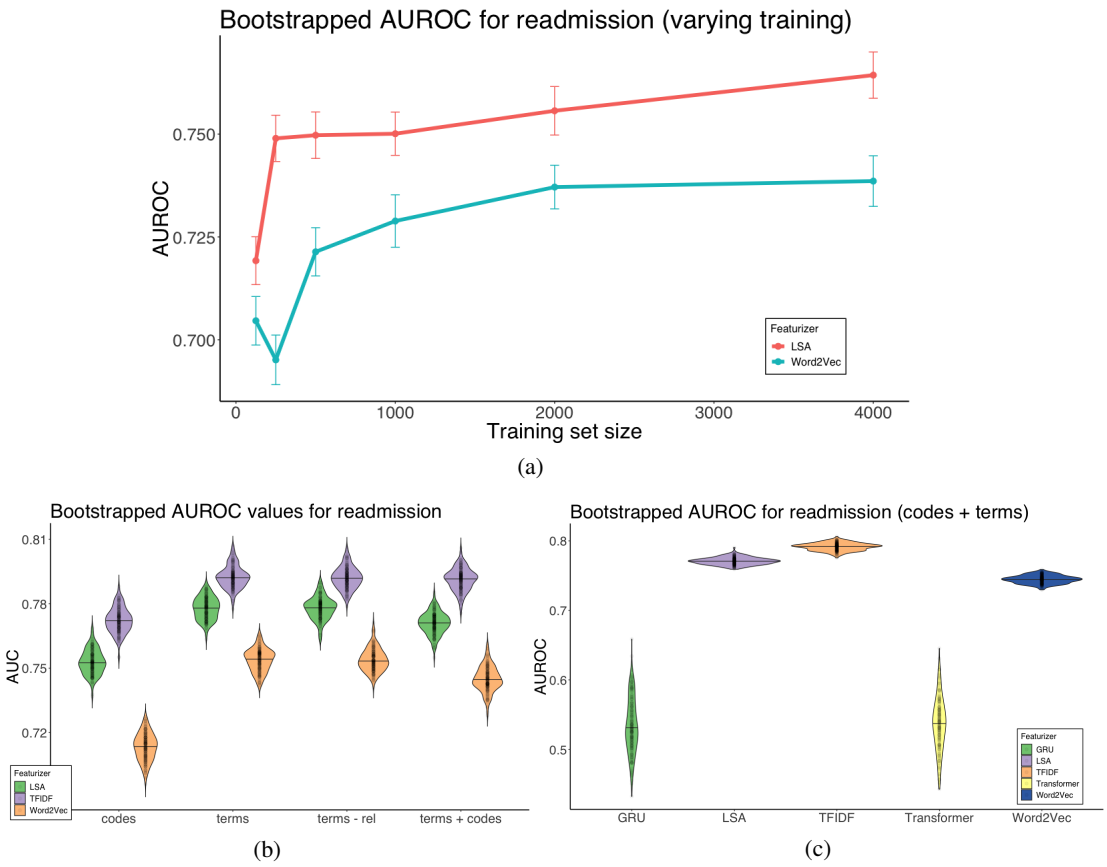

Figure 4: Inpatient readmission AUROC examined by training set size(4a), data stream (4b), and featurizer (4c).

5 Analysis & Conclusions

Survey of Embedding Methods Our results suggest that simple embedding methods yield the highest predictive performance on downstream tasks. TFIDF outperforms both LSA and Word2vec. Our language models perform essentially no better than random. Given enough training data, it makes sense that of TFIDF would perform so well. Although LSA and Word2vec produce more compact patient representations than that TFIDF, they appear to lose predictive signal in the process. Both TFIDF and LSA perform better than Word2vec. This may be the case for multiple reasons. First, our Word2vec formulation relies on the assumption that code and medical term representations can be elucidated from their neighboring codes and terms. By our own admission, the immediate neighborhood of codes and terms are stochastic (we shuffle all codes and terms in a patient day because we lack the temporal resolution to properly order them). In the majority of cases this shuffling shouldn't have a large effect because the average number of codes in a patient day is small (roughly 15) and we use a context window of size 10. However some patient days carry a large number of codes (upwards of 1000). In these cases, our context window is more or less random. Secondly, we may be losing the majority of our signal in our final aggregation method. We construct dense vectors for each code and medical term using Word2vec, but to create patient representations we take the max element wise across all codes and terms in a patient timeline.

Influence of Training Set Size In Figure 3a and 4a we show downstream predictive performance as a function of training set size for our baseline LSA and neural Word2vec embedding schemes. We hypothesized that our neural embedding methods would improve predictive performance on tasks where sample size (n) is small. We expected that pre-training embeddings on a large sample of data (750k patients) would allow downstream tasks to leverage learned patient representations in transfer style for low prevalence tasks. Although we see that the difference in performance of the

two embedding methods decreases as you decrease sample size, Word2vec never exhibits higher performance than LSA. This may be due to limitations in our Word2vec embedding scheme listed above.

Survey of Data Streams Our second aim was to elucidate the role of term mentions extracted from raw clinical text in downstream prediction tasks. In Figures 3b and 4b we see that data-streams that include term mentions perform roughly 3% better than the data-stream that uses structured data only. Although 80% of EHR data is unstructured and comes in the form of free text, extracting meaningful signal from it in a computationally efficient way is challenging. The fact that using medical term mentions extracted from clinical text as features on downstream tasks yields performance boosts over coded data is useful because: (1) it confirms that the majority of signal in EHR lies within clinical text and (2) demonstrates that enough signal can be extracted with computationally simple methods (string parsing) to make using clinical text worth it. Data streams that use a combination of text and structured data do no better than data streams that only look at text, and relation extract tools such as `Negex` and `ConText` do not appear to have any effect on downstream predictive performance for our chosen tasks.

Language Models Our language models perform worse than expected, as seen in Figures 3c and 4c. We hypothesized that codes and terms in a series of patient days would exhibit autocorrelation — i.e. we would be able to learn a model that predicts the next day's codes and terms given the current day's. This was done in [5], but with a larger temporal window. After examining the inner workings of our more complex language models (GRU and Transformer), we note that they seem to predict each code and term with probability equal to the frequency of each code and term in our training set. This may suggest that there simply exists no signal when using day sized temporal windows.

6 Future aims

One avenue we could explore is to make the transformer's positional encodings dependent on patient visits dates, as opposed to recurrent time steps of $t-1, t, t+1$, etc. Tangentially, this opens the doors to other forms of hidden state aggregation functions and pooling, such as taking the weighted average of later hidden states. One could imagine that most recent hidden state is the most predictive. Furthermore, on a given visit, further aggregation functions (e.g. mean) could be explored instead of maximum. Expanding on aggregation functions, as a third baseline featurizer, we are also interested in performing simple aggregation methods (e.g. min, max, mean) of our code and term embeddings to create a fixed length vector patient representation [5].

A point of concern with our performance is that the neural models, unlike TFIDF and LSA, take into account the sequential nature of patient visits. Such neural models may be performing worse than TFIDF and LSA due to a possible lack of autocorrelation between patient visits. A potential analogous approach could be found in the Autoencoder, in which the patient visits were treated as individual observations. This approach would be similar to that of Miotto et al's *Deep Patient* [1], in which we would create patient representations by capturing latent variables, structures, and patterns (including hierarchical and nonlinear ones) via the unsupervised approach of stacked Denoising Autoencoders.

We also are interested in trying other feature vector representations, using structured and *unstructured* data [1]. The overwhelming majority of deep learning NLP approaches limit themselves to structured clinical data in the EHR. Those that utilize free text notes, only look at medical terms/concepts extracted from the raw notes. Future work could extend these efforts by developing learned representations of patient EHR data that incorporate both structured clinical data and raw free text clinical notes. We hypothesize that patient representations trained with raw notes will outperform representations that look both solely at structured data, and structured data with medical concepts from notes on our two benchmark clinical prediction tasks. Comparing downstream performance results based on data-streams using term mentions and data-streams using raw text would be useful as we could elucidate what fraction of the signal is extracted using computationally simple string parsing tools.

When dealing with clinical text, we are also inters ted in trying out current approaches that utilize either (1) *a priori* domain knowledge bases (e.g. bag-of-words [5] and medical ontologies [3]) or (2) pretraining (e.g. BioBERT [14] [15]), allowing for the use of full sentence-level inputs.

7 Additional information

Mentors include Drs. Jason Fries and Nigam Shah of Stanford's Department of Biomedical Data Science.

References

[1] Miotto et al. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Nature*, 2016.

[2] Rajkomar et al. Scalable and accurate deep learning with electronic health records. *Nature Digital Medicine*, 2018.

[3] Choi et al. GRAM: Graph-based attention model for healthcare representation learning. *arXiv*, 2017.

[4] Jeffrey Pennington, Richard Socher, and Christopher Manning. GLOVE: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[5] Dubois et al. Efficient representations of clinical text. *arXiv*, 2018.

[6] Choi et al. Learning low-dimensional representations of medical concepts. *AMIA Joint Summits on Translational Science Proceedings*, 2016.

[7] Cho et al. On the properties of neural machine translation: Encoder–decoder approaches. *arXiv*, 2014.

[8] Chung et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*, 2014.

[9] Golmohammadi et al. Gated recurrent networks for seizure detection. *arXiv*, 2018.

[10] Simeon Kostadinov. Understanding GRU networks, https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be, 2017.

[11] Vaswani et al. Attention is all you need. *arXiv*, 2017.

[12] Lynn-Evans. Transformer, https://github.com/SamLynnEvans/Transformer, 2018.

[13] Alexander Rush. The annotated transformer, http://nlp.seas.harvard.edu/2018/04/03/attention.html, 2018.

[14] Devlin et al. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2018.

[15] Lee et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *arXiv*, 2019.