EVALUATING THE EFFECTIVENESS OF DEEP LEARNING IN DETECTING SYMPTOMS FROM FREE-TEXT NURSING NOTES

1. LITERATURE REVIEW

Up to 80% of all data in healthcare is captured as unstructured and semi-structured data, such as pathology notes, discharge summaries, and daily follow-up notes, among others. To extract meaningful data from these sources, many researchers have spearheaded efforts to expand natural language processing (NLP) to healthcare. For example, Tang et al. [3] used support vector machines to identify entities in patients' discharge summaries. However, fast and scalable NLP in healthcare is still a challenge.

One remaining challenge is that for many healthcare domains, there is a lack of "ground truth", high quality labeled training datasets needed for machine learning. One such challenging domain is disease symptoms, especially specific disease symptoms, such as cardiac symptoms. A recent systematic review found only 3 studies focused specifically on cardiac symptoms; all of these studies used a rule-based NLP approach in which researchers built large vocabularies of terms and then implemented NLP using these vocabularies. Such an approach is often precise; however, it has limited scalability and feasibility across healthcare settings and domains since much effort is needed to adapt previously developed vocabularies to new domains. This is one of the reasons for the growing use of a potentially more scalable and feasible machine learning-based NLP approaches in the recent years.

Tang et al. [4] also compared three different word representations: clustering-based representation, distributional representation and word embeddings regarding their usefulness in recognizing biomedical name entity. Jagannatha et al. [5] used LSTM and conditional random field in medical entity recognition. Cheng et al. [6] utilized support vector machines and active learning to investigate the disambiguation of words in the medical field.

There are many challenges that are associated with natural language processing in healthcare. Previous studies have shown that the inability to effectively model the subtleties of the medical sublanguage hinder performance [7]. To deal with this problem, practitioners in other fields have attempted to standardize the language before feeding them into machine learning models. However, because of the invariability of the clinical vocabularies and a lack of standardization in medical annotation practices, this is not practical in healthcare. In addition, traditional NLP methods can work with synonyms but not similar terms, so many of them suffer from the curse of dimensionality.

In the face of such challenges, advanced methods in natural language processing such as CNN, LSTM and BERT are also gaining popularity. For example, Razavian et al [8] utilized one LSTM network and two CNN models to predict multiclass disease onset. Lipton et al [9] used LSTM network to discover patterns from lab measurements and diagnosed based on the discovered patterns. Jagannatha et al. [10] used LSTM and conditional random field to detect entities from full-length clinical notes.

2. STUDY AIMS

This study investigates whether using deep learning models will result in better predictive performance compared to traditional machine learning methods.

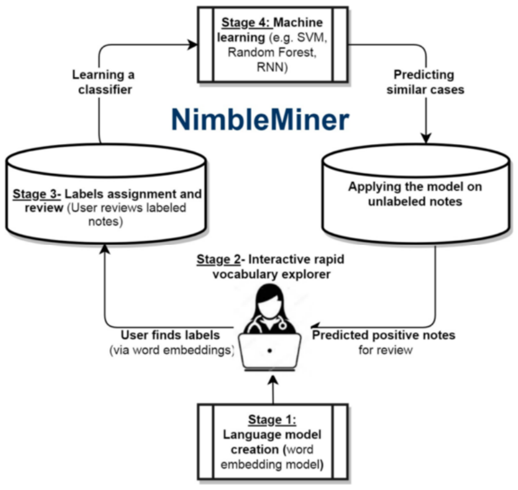
3. METHODS

In this section, we describe the deep learning models that we use to detect patients with discharge notes that exhibit symptoms of interest. This is a part of a bigger NLP system we developed called NimbleMiner.

NLP System Overview

NimbleMiner is an open-sourced vocabulary exploring and labeling system developed by our team [11, 12]. The working process of NibleMiner is demonstrated in Figure 1 below.

Figure 1. Steps of NimbleMiner working process


Stage 4: Machine learning (e.g. SVM, Random Forest, RNN)
Learning a classifier
Predicting similar cases
NimbleMiner
Stage 3- Labels assignment and review (User reviews labeled notes)
Applying the model on unlabeled notes
Stage 2- Interactive rapid vocabulary explorer
User finds labels (via word embeddings)
Predicted positive notes for review
Stage 1: Language model creation (word embedding model)

Step 1 - Creating a language model

In this step, the user chooses the corpus of clinical text of interest and defines the parameters for the language model, including which word embeddings to use for model generation. The user can customize the model settings based on their needs.

Step 2 - Interactively exploring the vocabulary

The objective of this step is to iteratively teach the system the relevant terms to search for. The user first enters an initial query term. Then, the system generates a list of similar terms that it deems relevant based on the embedding selected in Step 1. Next, the user chooses and saves the most relevant terms. Negated and irrelevant terms are also saved by the systems for negation detection in later steps.

Step 3 - Assigning and Reviewing Labels

After obtaining the list of relevant terms, the system will assign labels to the clinical notes these terms appear in, and exclude notes with irrelevant or negated terms. A clinical note receives a positive label when it has at least one relevant term. The user then reviews these notes and removes the notes he or she deem irrelevant. This form of weakly supervised learning has been validated in past research [13, 14].

Step 4 - Selecting and applying machine learning

Step 1 - 3 create the training and testing datasets that are needed for machine learning. In this step, the user selects a machine learning algorithm to generate a predictive model. The system will then apply this model to predict which clinical notes have the concept that the user is interested in. Finally, the user reviews the predicted note for accuracy. Step 2 - 4 can be repeated to add new labels.

The code repository and user manual can be found at http://github.com/mtopaz/NimbleMiner. The system is licenced under the GNU General Public Licence v3.0. In this paper, we describe in detail the deep learning models that we use in Step 4 of the NimbleMiner NLP system.

4. THE DATA

Our study is conducted on a dataset of discharge notes for patients with heart diseases. The training dataset has 8,031 notes, and the test dataset has 293 notes. The test set is a gold-standard, human-labelled dataset while the training set is generated by NimberMiner. There are eleven symptoms in our datasets. They all have class imbalances: the negative class outnumbers the positive class across symptoms. This adds another layer of difficulty to our task. Some deep learning models that we use, such as BERT, address the issue of class imbalance better than others do. The class imbalance is presented in Table 1 below.

Table 1. Prevalence of symptoms in the training and test set

	Training Set	Test Set
Dyspnea	39%	49%
Chest Pain	30%	30%
Nausea	21%	21%
Fatigue	18%	23%
Cough	16%	21%
Dizziness	14%	15%
Weight Change	10%	9%
Palpitation	8%	5%
Confusion	8%	11%
Peripheral Edema	7%	11%
Anorexia	5%	8%

Our notes are in the form of free text. They do not have strict structure and vary in length based on the symptoms of each patient and their severity. The percentiles of the note lengths are presented below in Table 2. The training set and the test set are relatively similar in terms of length: the maximum length is 4,547 words for the training set and 4,273 words for the test set, and the average length is 1,025 for the training set and 1,336 for the test set.

Table 2. Note length of training and test set

Percentile	Training Set	Test Set
25%	727	863
50%	1025	1336
75%	1482	1834
90%	2025	2435
95%	2388	2786
100% (Max)	4547	4273

Data processing

To prepare the free-text data for our deep learning models, we perform a standard tokenizing procedure. First, we remove all numbers from our free-text data. Most of the numbers in our text are random numbers (numbers that were de-identified) and therefore would serve no meaningful purpose in our symptom detection task. Then, we tokenize the text by converting them into a range of numbers: from 1 to n, where n is the number of features. This number is selected as 15,000 out of a total of 65,000 most frequent words. This means that all the words in our text are now represented by 15,002 tokens: 15,000 known tokens for 15,000 most frequent words, the “0” token for length padding and the “15,001” token representing all other words. We choose the maximum length of each text to be 2,400 words, or approximately 95% percentile of all text length. Doing so, we ensure we have enough data for training but don’t utilize too much computational resources. Next, we pad each row with the “0” token to ensure that all rows have the same length. After this process, our training data is transformed into a 8,031 by 2,400 matrix, and our test data a 293 by 2,400 matrix. Finally, we split the training set into a training and a validation set with an 80:20 ratio for the training procedure.

Baseline Models

For this study, we choose logistic regression and random forest as our two baseline models. Logistic regression and random forests have been used widely as predictive models of choice for text classification [15, 16]. For logistic regression, we cast the notes into a sparse matrix before training the model. For random forest, we also cast the notes in sparse matrices, and then used the "ranger" implementation of random forest in the R language, using twenty decision trees and "impurity" as our importance metrics. Among these two baseline models, random forest has been widely used in natural language processing tasks because it is highly interpretable.

Deep Learning Model Training

Traditional natural language processing methods, such as our baseline models, are limited by their vocabulary dictionary for predictive tasks and require complex processing steps. In addition, simplistic models such as logistic regression cannot effectively recognize the semantic relationships between words and n-grams. We choose to perform text classification using word embeddings and convoluted neural networks (CNN) because such models don't have these restrictions.

We experiment with four different architectures: (1) a 1-layer embedding and 4-layer CNN; (2) a 1-layer fastText embedding and 4-layer CNN and (3) 1-layer embedding, 4-layer CNN followed by a 1-layer Bidirectional Long-short Term Memory (LSTM) and (4) a BERT layer, 2-layer CNN and 1-layer LSTM. We train binary classifications using these three architectures for all eleven symptoms. These models are trained using the Keras library in R with a Tensorflow backend.

Word Embeddings

Word embeddings are proven to be particularly useful if we don't have a lot of training data [17]. Before being fed into a neural net architecture, each row in our training set is represented as a series of word embeddings, in which each word is projected into a distributed representation. Words that appear in similar context are learnt to have similar embeddings [18]. Therefore, misspellings, synonyms and abbreviations do not need to be corrected or expanded before training. They will be represented by relatively identical embeddings.

Word embeddings can be learned from the data we provide, or from a larger corpus of notes. The benefits of the latter are to improve the performance of the NLP system as well as to require less training data for our task. For two of our deep learning architectures, we train an embedding layer based on existing text corpus in our training set. For the last one, we use the fastText pretrained embedding. fastText is an embedding method extended from the word2vec embedding. Instead of directly learning vectors of words, fastText learns words as n-grams of characters. This helps our embedding layer to learn the meanings of shorter, rarer words, as well as suffixes and prefixes [19]. We choose fastText over GloVe and other advanced embedding methods for this feature, as our notes include a lot of abbreviations and shortened words, as commonly seen in discharge notes.

BERT (Deep Bidirectional Transformers for Language Understanding):

Open-sourced by Google AI, BERT is an advancement in state-of-the-art NLP techniques. It is the first deeply bidirectional, unsupervised language representation pre-trained on a plain text corpus. Similar to other contextual models, BERT generates a representation for each word using the other words in the sentence. If a unidirectional contextual model represents "I accessed the bank account" by looking at only "I accessed the" and not "account," BERT represents it by both components from the very bottom of the deep neural net, making it "deeply bidirectional."

Convolution Neural Network:

We use CNN as proposed by Kim et al [20] as the main building blocks for our natural language processing task. CNN in computer vision uses filters to condense neighboring pixels into a single pixel while still retaining the most important information [21]. Similarly, in our text classification task, we use CNN to combine adjacent tokens into more simple "concepts" while keeping the contribution of these tokens in predicting the symptom of interest.

The embedding texts are used as inputs in our convoluted layers. In each convolutional operation, a filter is applied to each token window and results in a signal. The filter reduces the text to a matrix of signals, called a feature map, which is then reduced to a single value using the max pooling operation.

To avoid overfitting, we use dropout layers with a dropout rate of 20% between the convolution layers.

Finally, we train CNN using the cross-entropy loss function and the RMSProp optimization algorithm. The output for each model training is a probability of the text belonging to the positive class. For consistency purposes, for all binary classifications, we train for 10 epochs and use a batch size of 64; while for all multiclass classifications, we train using 20 epochs with a batch size of 64.

Bidirectional Long-Short Term Memory:

Recurrent neural networks, such as LSTM, are proven effective in dealing with sequential data such as time series, language and speech. LSTM are currently empowering language-related applications in the healthcare domain such as electronic health records (EHR).

LSTM provides a way to carry information between multiple words and sentences in a text. Similar to a conveyor belt, LSTM can carry a token or word onto the belt at any point, transport it to a later step and bring it off intact. This is the mechanism LSTM uses to prevent older tokens from vanishing during processing: it preserves the semantic relationship between tokens in a very long text.

For one of our deep learning models, we use a bidirectional LSTM layer between the convolutional layers and the final dense layer to investigate whether or not this better captures the contextual nature of our discharge note.

Model Summary:

We provide a summary of our deep learning architecture in Table 3 below.

Table 3. Summary of the architecture of 4 deep learning models

4-layer CNN	**fastText embedding + 4-layer CNN**	**4-layer CNN + bidirectional LSTM**	**BERT + 2-layer CNN + LSTM**
Embedding Layer Input dimension = 15,000 Input length = 2,400 Output dimension = 128	**fastTex Embedding Layer** Input dimension = 15,000 Input length = 2,400 Output dimension = 300	**Embedding Layer** Input dimension = 15,000 Input length = 2,400 Output dimension = 128	**BERT Layer**
1D Convolution Layer Filter = 32 Kernel size = 3	**1D Convolution Layer** Filter = 32 Kernel size = 3	**1D Convolution Layer** Filter = 32 Kernel size = 3	
Dropout Layer Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2
1D Convolution Layer Filter = 64 Kernel size = 3	**1D Convolution Layer** Filter = 64 Kernel size = 3	**1D Convolution Layer** Filter = 64 Kernel size = 3	**1D Convolution Layer** Filter = 64 Kernel size = 5
Dropout Layer Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2
1D Convolution Layer Filter = 64 Kernel size = 3	**1D Convolution Layer** Filter = 64 Kernel size = 3	**1D Convolution Layer** Filter = 64 Kernel size = 3	**1D Convolution Layer** Filter = 64 Kernel size = 5
Dropout Layer Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2	**Dropout Layer** Dropout Rate = 0.2
		Bidirectional LSTM Layer Unit = 128	**Bidirectional LSTM Layer** Unit = 32
Global Max Pooling Layer	**Global Max Pooling Layer**	**Global Max Pooling Layer**	**Global Max Pooling Layer**
Dense Layer *Prediction*	**Dense Layer** *Prediction*	**Dense Layer** *Prediction*	**Dense Layer** *Prediction*

Model evaluation

The test set is a gold-standard dataset that is unused and hidden until after training is finished.

We report the accuracy, precision, recall and F-score for the binary classifications of all eleven symptoms. We also report the same metrics for the multiclass classifications with different chunk sizes. All metrics are derived from the confusion matrix between the model predictions and true label values from the test set. To generate this confusion matrix, we pick a threshold over which a prediction is classified as 1, otherwise it is classified as 0.

The precision (P) is calculated as the number of true positives out of all data points that are predicted to be positive: $P = \frac{TP}{TP+FP}$. The recall (R) is calculated as the number of true positives out of all data points that should have been predicted as positive: $R = \frac{TP}{TP+FN}$. F-score is the harmonic mean of precision and recall: $F - score = \frac{2*P*R}{P+R}$. We used the F-score as our main metrics to evaluate how well a deep learning architecture performs. The F-score we used is the micro-average F-score, meaning it is unweighted and therefore reflects the class imbalances that we observe in some labels with prevalent negative class such as palpitations and anorexia. All F-scores are reported at the threshold where it is the highest.

5. RESULTS

We found that deep learning models, on average, produce better performance when compared to traditional machine learning methods. A full comparison can be found in Table 4.

Table 4. Overall predictive performance (F-measure) of the best deep learning model compared to that of traditional machine learning model.

Symptom	Logistic Regression	Random Forest	4-layer CNN	fastText embeddings + 4-layer CNN	4-layer CNN + Bidirectional LSTM	BERT, 2-layer CNN and LSTM
Dyspnea	0.19	0.9	0.91	0.89	0.91	0.93
Chest Pain	0.24	0.79	0.86	0.85	0.86	0.88
Nausea	0.19	0.42	0.89	0.83	0.86	0.84
Fatigue	0.08	0.86	0.85	0.5	0.83	0.88
Cough	0.32	0.79	0.81	0.77	0.82	0.84
Dizziness	0.14	0.8	0.85	0.47	0.77	0.9
Weight Change	0.14	0.65	0.7	0.63	0.69	0.85
Palpitation	0.00	0.56	0.75	0.58	0.62	0.72
Confusion	0.21	0.9	0.88	0.3	0.88	0.88
Peripheral Edema	0.17	0.69	0.64	0.33	0.59	0.8
Anorexia	0.00	0.65	0.77	0.34	0.57	0.8

In addition, we can observe that our architectures perform better on symptoms with a more balanced prevalence. We divide our symptoms into two groups: group 1 has more than 15% of positive prevalence in the training set (dyspnea, chest pain, fatigue, nausea and cough) and group 2 has less than 15% (dizziness, confusion, weight change, peripheral edema, anorexia and palpitation). We observe that the average performance of the first group exceeds that of the second group from 4% to 33%. Among the deep learning architectures, CNN and LSTM with BERT embeddings has the least difference in average performance between two groups. It means that it suffers the least biases that result from class imbalance.

Table 5. Difference in performance between two groups - positive prevalence more and less than 15%.

Symptom	Logistic Regression	Random Forest	4-layer CNN	fastText embeddings + 4-layer CNN	4-layer CNN + Bidirectional LSTM	BERT, 2-layer CNN and LSTM
Prevalence >15% training set	20.40%	75.20%	86.40%	76.80%	85.60%	87.40%
Prevalence <15% training set	11.00%	70.83%	76.50%	44.17%	68.67%	82.50%
Difference	9.40%	4.37%	9.90%	32.63%	16.93%	4.90%

We also experiment with whether excluding or including stop words will affect our predictions' accuracy and F-score, as preprocessing steps such as excluding stop words and lemmatization are essential in traditional machine learning methods. Despite some evidence in relevant literature that excluding stop words may hinder model performances, doing so actually helps improve our model performance by a small margin.

	4-layer CNN	fastText embeddings + 4-layer CNN	4-layer CNN + Bidirectional LSTM	BERT, 2-layer CNN and LSTM
Dyspnea (with stopword)	0.91	0.89	0.91	0.93
Dyspnea (without stopword)	0.93	0.90	0.92	0.93

Since deep learning models outperform the baseline models, below we present the accuracy and F-score of all eleven symptoms using three different deep learning architectures.

Figure 1. Accuracies of three deep learning methods on eleven symptoms

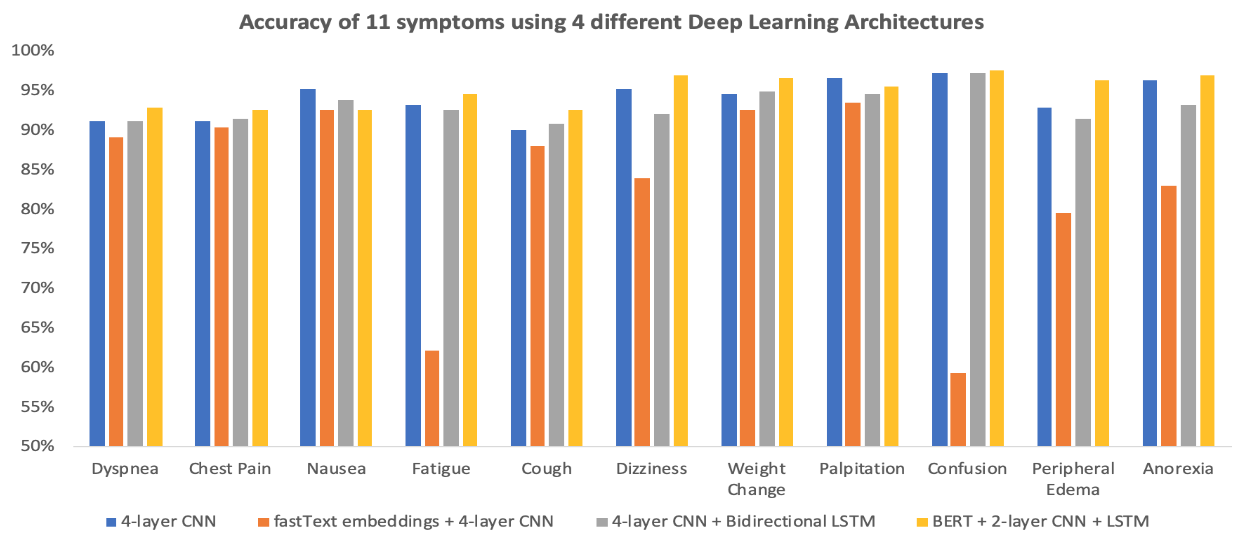
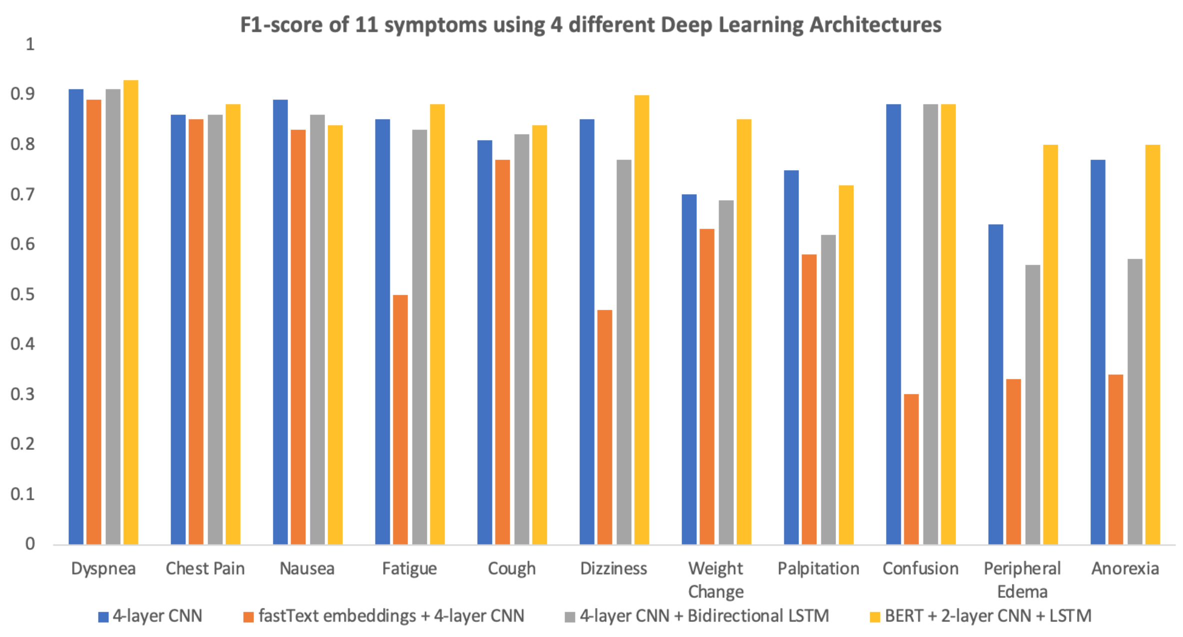


Figure 2. F-scores of three deep learning methods on eleven symptoms



For the predictions on the symptoms, we observe good performances on dyspnea, chest pain, fatigue, nausea, cough, dizziness and confusion. For these symptoms, our models produced an F-score of over 0.75 with consistently high accuracy scores, with few exceptions. This means that our models are adept at correctly predicting the less prevalent class in spite of the class imbalance. However, when dealing with the symptoms with prevalence less than 10% in the training set, model performance becomes progressively worse with F-scores dropping below 0.7.

Among the three deep learning architectures, 4-layer CNN, when combined with a simple embedding layer, performed consistently better. 4-layer CNN and bidirectional LSTM performed mostly on par with 4-layer CNN and provided some additional benefits in terms of precision. FastText embeddings and 4-layer CNN performed worse and had poor results for symptoms like fatigue and peripheral edema.

Table 6. Performance of four different deep learning architectures on all symptoms, sorted by symptom prevalence

Symptom	**4-layer CNN**	**fastText embeddings + 4-layer CNN**	**4-layer CNN + Bidirectional LSTM**	**BERT + 2-layer CNN + LSTM**
DYSPNEA				
Accuracy	91.13%	89.08%	91.13%	92.83%
Precision	0.89	0.87	0.88	0.91
Recall	0.94	0.91	0.94	0.94
F-score	**0.91**	**0.89**	**0.91**	**0.93**
CHEST PAIN				
Accuracy	91.13%	90.44%	91.47%	92.49%
Precision	0.82	0.81	0.87	0.87
Recall	0.9	0.9	0.84	0.89
F-score	**0.86**	**0.85**	**0.86**	**0.88**

NAUSEA				
Accuracy	95.22%	92.49%	93.86%	92.49%
Precision	0.84	0.8	0.83	0.77
Recall	0.95	0.85	0.89	0.92
F-score	**0.89**	**0.83**	**0.86**	**0.84**
FATIGUE				
Accuracy	93.17%	62.12%	92.49%	94.54%
Precision	0.85	0.35	0.84	0.9
Recall	0.85	0.83	0.82	0.85
F-score	**0.85**	**0.5**	**0.83**	**0.88**
COUGH				
Accuracy	90.10%	88.05%	90.78%	92.49%
Precision	0.68	0.64	0.7	0.77
Recall	1	0.95	0.98	0.92
F-score	**0.81**	**0.77**	**0.82**	**0.84**
DIZZINESS				
Accuracy	95.22%	83.96%	92.15%	96.93%
Precision	0.78	0.46	0.68	0.87
Recall	0.93	0.49	0.88	0.93
F-score	**0.85**	**0.47**	**0.77**	**0.9**
WEIGHT CHANGE				
Accuracy	94.54%	92.49%	94.88%	96.59%
Precision	0.68	0.59	0.74	0.79
Recall	0.73	0.73	0.65	0.85
F-score	**0.7**	**0.63**	**0.69**	**0.85**
PALPITATION				
Accuracy	96.59%	93.52%	94.54%	95.56%
Precision	0.6	0.43	0.48	0.54
Recall	1	0.87	0.87	0.87

F-score	**0.75**	**0.58**	**0.62**	**0.72**
CONFUSION				
Accuracy	97.27%	59.39%	97.27%	97.61%
Precision	0.88	0.18	0.88	1
Recall	0.88	0.76	0.88	0.79
F-score	**0.88**	**0.3**	**0.88**	**0.88**
PERIPHERAL EDEMA				
Accuracy	92.83%	79.52%	91.47%	96.25%
Precision	0.7	0.26	0.64	0.86
Recall	0.59	0.47	0.5	0.78
F-score	**0.64**	**0.33**	**0.56**	**0.8**
ANOREXIA				
Accuracy	96.25%	82.93%	93.17%	96.93%
Precision	0.75	0.25	0.57	0.82
Recall	0.78	0.57	0.57	0.78
F-score	**0.77**	**0.34**	**0.57**	**0.8**

6. DISCUSSION

This study found that deep learning methods outperform traditional machine learning methods. Our CNN models might have performed better because, through multiple hidden layers, they can capture the semantic relationships among the notes at different granularities: note, sentence, word and n-grams [22]. Each filter in the convolutional operations captures local semantics, while the max pooling operation captures global semantics. Another major reason that might enable CNN to perform well is that it can effectively capture correlations between different symptoms. We found that deep learning methods tend to be less accurate as the number positive cases in the training data decreases. It is understandable that with less than 10% of the positive training data provided, the models have not seen enough data to correctly detect study symptoms. We believe that if we can collect or label more data with the presence of these rare symptoms, we will be able to improve performance.

Our results show that models with FastText embeddings with 4-layer CNN performed worse than other deep learning methods. This strange phenomenon has a possible explanation based on the nature of fastText embedding. First, the standardization of annotation in discharge notes is low with many shortened terms and phrases, while fastText embedding was trained on a large corpus of formal words. While fastText embedding is supposed to be able to spot abbreviations and misspellings, it might not be able to capture implicitly understood phrases and terms among nurses and clinicians effectively. In future work, we plan to either find or develop a more effective embedding to deal with medical data or find a way to standardize the annotation standards for our discharge notes.

We also found that models without stop words perform marginally better than those with stop words. This is because while the stop words can serve as connectors between tokenized features, removing them help reduce the number of dimensions and as a result, help the models focus on the semantic relationships between the important features. This reflects in our results: for example, excluding stop words helps improve the F-score across all four models in the case of dyspnea.

For rarer symptoms, we suggest using alternatives other than deep learning. A rule-based approach may be appropriate in detecting and predicting the presence of these symptoms [23]. However, rule-based approaches require adding new rules and models every time a new symptom appears or an existing one evolves, which may be too complex to thoroughly encode in a single model. Another alternative is to combine more features with an ensemble method such as gradient boosting trees to predict these symptoms.

Limitations

First, the experiments we conducted on relatively small number of discharge notes from one academic medical center- thus our results generalizability is limited. Second, because of the relatively low number of notes in the training data, we used all the notes in the sample to train the machine learning models. Further studies might explore using more data and alternative training sample generation approaches (e.g., a recently suggested empirical-risk-minimization based method for weighting positive learning datasets with propensity scores) to improve predictive performance.

In the realm of health analytics, interpretability is highly valuable for clinicians to be able to understand what causes the symptoms. In addition, when an outcome is inaccurately predicted, it is also important to be able to interpret the outcome to fully understand the source of biases. While some applications such as input saliency analysis can help alleviate the problem of opaqueness that usually causes clinicians to distrust neural networks, newer and more advanced models such as BERT remain difficult to interpret. Another disadvantage of CNN is that the model considers a large collection of items (phrases) as contributing to the prediction of the output, so it is difficult to pinpoint exactly what is the importance of each individual phrase or word.

Future Extensions

As EHR is becoming ubiquitous and machine learning algorithms for natural language processing are developing very quickly, there are ample opportunities for improvements. There are more features available in our dataset, such as patients' ages, time and dates of the discharge notes and so on. We can potentially use these features to impose a temporal structure on our data set to not only predict a symptom, but also predict when it will occur in the future.

References

1. Pham, T., Tran, T., Phung, D. & Venkatesh, S. Deepcare: A deep dynamic memory model for predictive medicine. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, 30–41 (Springer, 2016)
2. Miotto, R., Li, L., Kidd, B. A. & Dudley, J. T. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. Sci. reports 6, 26094 (2016)
3. Tang, B., Cao, H., Wu, Y., Jiang, M. & Xu, H. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. BMC medical informatics decision making 13, 1 (2013)
4. Tang, B., Cao, H., Wang, X., Chen, Q. & Xu, H. Evaluating word representation features in biomedical named entity recognition tasks. BioMed research international 2014 (2014)
5. Jagannatha, A. N. & Yu, H. Structured prediction models for rnn based sequence labeling in clinical text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, vol. 2016, 856 (NIH Public Access, 2016)
6. Chen, Y., Cao, H., Mei, Q., Zheng, K. & Xu, H. Applying active learning to supervised word sense disambiguation in medline. J. Am. Med. Informatics Assoc. 20, 1001–1006 (2013)
7. Spasić I, Livsey J, Keane JA, Nenadić G. Text mining of cancer-related information: review of current status and future directions. Int J Med Inform 2014 Sep;83(9):605-623 [FREE Full text] [doi: 10.1016/j.ijmedinf.2014.06.009] [Medline: 25008281]
8. Razavian, N., Marcus, J. & Sontag, D. Multi-task prediction of disease onsets from longitudinal lab tests. arXiv preprint arXiv:1608.00647 (2016).
9. Lipton, Z. C., Kale, D. C., Elkan, C. & Wetzell, R. Learning to diagnose with lstm recurrent neural networks. arXiv preprint arXiv:1511.03677 (2015).
10. Jagannatha, A. N. & Yu, H. Structured prediction models for rnn based sequence labeling in clinical text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, vol. 2016, 856 (NIH Public Access, 2016).
11. M. Topaz, K. Gaddes, M. McDonald, and K. Bowles, Development and Validation of a Novel Rapid Clinical Text Mining Approach Based on Word Embeddings (NimbleMiner)., Stud. Health Technol. Inform. 244 (2017) 88.
12. M. Topaz, L. Murga, K.M. Gaddis, M. V. McDonald, O. Bar-Bachar, Y. Goldberg, and K. Bowles, Mining fall related information in clinical notes: comparison of rule based and novel word embedding-based machine learning approaches, J. Biomed. Inform. (2019) 103103. doi:10.1016/J.JBI.2019.103103.

13. Y. Halpern, S. Horng, Y. Choi, and D. Sontag, Electronic medical record phenotyping using the anchor and learn framework., J. Am. Med. Inform. Assoc. 23 (2016) 731–40. doi:10.1093/jamia/ocw011.
14. C. Elkan, and K. Noto, Learning classifiers from only positive and unlabeled data, in: Proceeding 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD 08, 2008: p. 213. doi:10.1145/1401890.1401920.
15. Cox, D. R. The regression analysis of binary sequences. J. Royal Stat. Soc. Ser. B (Methodological) 215–242 (1958).
16. Ho, T. K. Random decision forests. In Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on, vol. 1, 278–282 (IEEE, 1995).
17. Wu Y, Xu J, Jiang M, Zhang Y, Xu H. A Study of Neural Word Embeddings for Named Entity Recognition in Clinical Text. In: AMIA Annual Symposium Proceedings. vol. 2015. American Medical Informatics Association; 2015. p. 1326.
18. Carrell DS, Halgrim S, Tran D-T, et al. Using Natural Language Processing to Improve Efficiency of Manual Chart Abstraction in Research: The Case of Breast Cancer Recurrence. doi:10.1093/aje/kwt441.
19. A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of Tricks for Efficient Text Classification. arXiv:1607.0179
20. Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:14085882. 2014
21. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324. doi: 10.1109/5.726791
22. Yuan Yang, Pengtao Xie, Xin Gao, Carol Cheng, Christy Li, Zhang Hongbao, and Eric P. Xing. Predicting discharge medications at admission time based on deep learning. arXiv preprint arXiv:1711.01386, 2017. 1.1
23. Farkas R, Szarvas G, Hegedűs I, Almási A, Vincze V, Ormándi R, et al. Semi-automated construction of decision rules to predict morbidities from clinical texts. Journal of the American Medical Informatics Association. 2009;16(4):601–605. doi: 10.1197/jamia.M3097.
24. Li J, Chen X, Hovy E, Jurafsky D. Visualizing and understanding neural models in nlp. arXiv preprint arXiv:150601066. 2015