Tagging Patient Notes With ICD-9 Codes

Sandeep Ayyar*
Biomedical Informatics
Stanford University
Stanford, CA 94305
ayyars@stanford.edu

Oliver Bear Don't Walk IV*
Biomedical Informatics
Stanford University
Stanford, CA 94305
oliverb4@stanford.edu

Abstract

There is substantial growth in the amount of medical/data being generated in hospitals. With over 96% adoption rate[1], Electronic Medical/Health Records are used to store most of this medical data. If harnessed correctly, this medium provides a very convenient platform for secondary data analysis of these records to improve medical and patient care. One crucial feature of the information stored in these systems are ICD9-diagnosis codes, which are used for billing purposes and integration to other databases. These codes are assigned to medical text and require expert annotators with experience and training. In this paper we formulate this problem as a multi-label classification problem and propose a deep learning framework to classify the ICD-9 codes a patient is assigned at the end of a visit. We demonstrate that a simple LSTM model with a single layer of non-linearity can learn to classify patient notes with their corresponding ICD-9 labels moderately well.

1 Introduction

The international Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM) is the US health system's adaptation of the international ICD-9 standard list of six-character codes to describe diagnoses [2]. It is a list of codes mapping to diagnoses and procedures recorded in hospital care in the US. These codes are then entered into a patient's electronic health record and further used for diagnostic, billing and reporting purposes. These codes are assigned to medical text and require expert annotators with experience and training.

The idea behind standardizing these codes is to enable consistency among physicians in recording patient symptoms, diagnoses for clinical research and reimbursements claims. It is the common terminology upon which most US health care payment systems are based and other major standards and practices have been built around it. For example, the reimbursement process by insurance companies is based on the codes assigned to medical text reports following a patient's treatment in the clinic/hospital.

Currently, medical coders assign a set of appropriate ICD-9 codes after review information about a patient's record for a clinical event. This labeling task requires expert knowledge in the field of medicine and the process in itself is expensive and prone to errors. A study [16] indicates that only 60%-80% of the assigned ICD-9 codes reflect actual patient diagnoses. This disagreement stems from coders who may assign a more serious code than is present (over-coding) or miss codes altogether (under-coding) which can result in serious financial loss. The approximate cost of icd-9 coding clinical records and correcting related errors is around $25 billion per year in the United States. [6].Codes are also very important in determining the eligibility of patients in clinical trials.

*Contributed equally

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

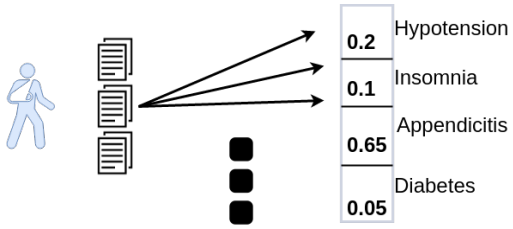


Figure 1: Going from patient visit, to clinical notes, to a list of ICD-9 codes can be done by either a human or a computer with varying degrees of success.

These issues along with the challenge of exploring interesting and unstructured data have prompted a large interest in automating the assignment of ICD-9 codes through various natural language processing and machine learning approaches. An automated system with high accuracy can reduce costs, simplify task for medical coders and help hospitals to standardize data.

2 Previous Work

The idea of automating ICD-9 codes assignment has been studied since 1990. Several researchers have looked at automatically assigning icd-9 labels using patient records by looking at text or notes written by physicians or hospital staff. Previous work has been focussed on pattern matching [10], rule base systems [4],[5] [Goldstein et ., 2007] or supervised classification methods such as Logistic Regression, k-NN and Support Vector Machines [10],[14]. Rule based systems consisted of manually crafted rules that capture lexical elements, derived from one or more algorithms, such as n-grams (sequence of consecutive words) or s-grams (sequence of consecutive words) extracted from reports, and enhanced by semantic information. The lexical elements are basically short, meaningful set of words. These rules base systems closely represent the procedure followed by medical coders for manual annotation.

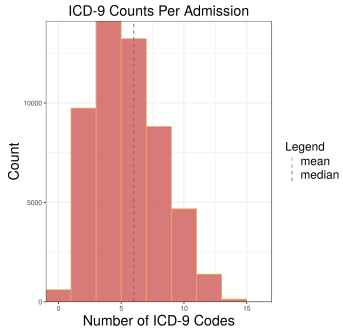
Several machine learning methods implemented such as K-NN, Naive Bayes, SVM, ridge Regression have differed in the type of data set and method used. Eg. K-NN and SVM were used for assigning ICD-9 codes to data that was preprocessed and represented in the form of manually designed features (using techniques such as remove of negative, stop words, TF-IDF of words in the text, etc). However, only limited number of codes were used for this problem[10]. Zhang et al., [8] specifically used SVMs to achieve high F1 score of 86.6, however using only radiology reports with limited ICD-9 codes (45). while Perotte et al.,[14] also used SVM on a larger corpus (22000 documents) with documents represented as a bag of words but reported lower F1 scores of 39.5. In general, most machine learning methods were outperformed by rule based methods. Manual features fail to capture the complexity found in medical records (sequential representation of information in patient records). Hence, even though representing medical text as manual features has been found to work on particular problems (such as specific set of codes, specific data sets), it has not been able to generalize on larger data sets.

Deep learning has potential to overcome the limitations of traditional machine learning and rule based systems by eliminating the task of describing explicit features or rules. There has been some work in applying neural networks to textual classification involving multiple labels. [11]. A recent paper in the previous version of this class [12] applied neural networks towards predicting ICD-9 codes. However, their approach did not deal with all ICD-9 codes but preselected data to include only the top 10 most frequently occuring ICD-9 codes. The models implemented in this study did not yield high performance (F1score: 0.37) suggesting that there is considerable room for improvement in both word representation and model architecture applied.

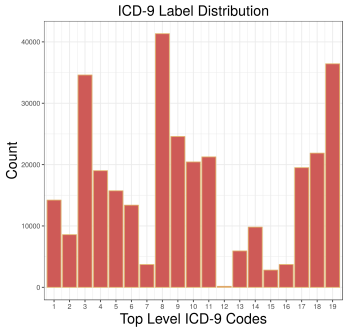
3 Dataset

MIMIC-III is the third iteration of a dataset generated by ICU/CCU patients at the Beth Israel Deaconess Medical Center between 2001 and 2011 [9]. The database represents 40,000 patients with structured and unstructured data including medications, test results, procedures performed, and free

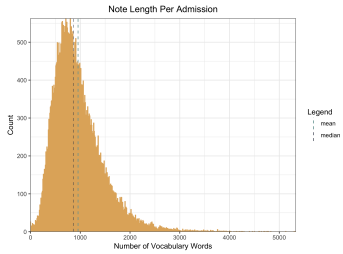
text. The nature of this dataset skews the ICD-9 distribution towards more debilitating and severe diseases/conditions. In order to focus on the NLP task our research only focused on free text from the Notes table, and only on notes under the category of discharge summary. Further filtering was performed by removing discharge summaries with no text or no associated ICD-9 codes.



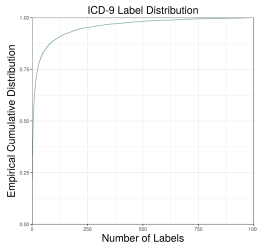
(a) The distribution of assigned ICD-9 Codes per admission



(b) The number of mentions for each top level ICD-9 Code



(c) The distribution of discharge summary note length



(d) The empirical distribution of original ICD-9 Labels

Figure 2: Information on the notes

The ICD-9 coding system is based on the World Health Organization Guidelines. One ICD-9 code indicates a classification of a disease, diagnostic or treatment procedure, injury, symptom or information from patient history. Codes are structured hierarchically where top level categories and more generic (eg neoplasm or diseases of the respiratory system) and lower level codes indicate specific diseases (eg. breast cancer, pneumonia). Higher level codes are only 3 digits while lower level codes are 4,5 digits with a decimal after the 3rd digit. These codes are distributed such that a minority of the codes make up a majority of the distribution of label mentions 2d. Because there are 6,198 codes our predictive power will be severely diminished for codes which do not often show up. We mapped all codes to their top level representation in the ICD-9, which left 19 top level ICD-9 codes 2b.

Unlike other NLP and deep learning tasks the discharge summary notes can be very long, and have a large variance in note length distribution 2c. The length of the notes can cause issues as any architecture which relies on temporal dependencies will suffer without long-term memory and how mentions of a disease early on in the note can affect predictions later on. This will be talked about in the Discussion and Future Work sections. The notes are also laden with medical jargon not common in other corpora, as well as misspellings.

4 Methods

All data manipulation and model training was done in python 3.5.3, TensorFlow 1.0.0 and R 3.3.1 on the Microsoft Azure GPU instances provided through CS224N.

4.1 Preprocessing

The table for Notes and ICD-9 Diagnoses were retrieved from the MIMIC-III database and joined using the dplyr package in R. Next, only notes which had a category value of 'Discharge' and

a description value of 'Summary' were kept. This narrows down the focus of the work so that instead of working with partial patient information we can work with information gathered after the patient has been discharged. This also narrows down the number of notes which correspond to the same admission ID, where we arbitrarily took the first one to occur chronologically. Finally, only admissions which have non-empty text and at least one diagnosis are kept. The data was split into a 75-25 training validation split resulting in 39,541 and 13,181 observations in the training and validation sets respectively.

While creating matrix representations of the clinical notes we converted tokens to arbitrary IDs and then used these IDs to map words to their corresponding word vector during training. However, because hardware constraints we were forced to set a maximum limit on the number of words in a note (the tunable hyper-parameter max length) in order to batch process our notes. This means that notes which are longer than max length only use the first max length words during training and testing, while notes which are less than max length are padded with 0s, however only the prediction corresponding to the last true word (not padded) is used during training and testing.

4.2 Model Description

Tokenization, Word Vectors from Glove: Our input to the neural network model consisted of a sequence of words from the medical text. The text was split into individual words by space and punctuation. For every word we obtained pretrained word vectors from Glove (Common Crawl 840 billion tokens, 2.2 million vocab of dimension size 300)[7]. Since our text consists of translated text from clinical notes, there are several misrepresentations or errors in spellings of words themselves. Hence for words not in our vocab, we simply represent the word vector as 0 initialized (Need to report how many words do not have word vector representation). These word vectors are then fed into the neural network for learning representations in sequential form i.e. one word vector after another .

RNN: The basic idea behind using Recurrent neural networks or RNN's for our purpose is that they make use of sequential information. Traditional neural networks have a fixed input length, which doesn't work for free text, and do not allow for temporal dependencies. In our task, this is not useful since we are trying to take into account sequential information about medical text. We ideally want to know the probability of an event (or ICD9 code in our case) given a sequence of words that occurred previously. RNN's capture this information as memory that has been computed up to and including the current word in the document. However, RNN's are limited to looking back only a few steps (short term dependency). In our case, the length of the notes is typically long (average of 910 words). Hence we need to learn long-term temporal dependencies which is difficult in case of RNN's as gradients decay exponentially with time (also known as the vanishing gradient problem). Hence we use LSTM's as our baseline model.

LSTM: Long Short Term Memory Networks or LSTMs are a special kind of RNN that can learn long term dependencies[13]. They are explicitly designed to avoid the long-term dependency problem. These use special units which include a memory cell that maintain information for a longer period of time compared to traditional RNNs. It consists of a set of gates that control when information enters memory, its output and when its forgotten. Hence this architecture is more suitable for our task which requires these cells to learn long term dependencies.

We trained our LSTMs with the 300 word vector representation from Glove as described above. Our baseline LSTM model consisted of hidden layer size 100, and a batch size of 256 for our training data (We split our data into 75% training and 25% test). Since the text in our training data was variable in size (each note had a different length of words: 1000 words on average), we tested our models on different note lengths (250, 500, 750, 1000) to see if change in the length of notes affects our model performance. We also added drop out to our LSTM model. Dropout is a regularization technique to prevent overfitting[17]. In this technique, neurons are randomly selected during training and dropped out randomly. This removes their contribution to the activation of any downstream neurons in the forward pass and any weight updates not applied to the neuron in backward pass. Therefore if any neurons are randomly dropped out of the network during training, then other neurons take over the handling of the representations to make predictions. This makes the overal network less sensitive to specific weights of neurons and results in multiple independent internal representations learned by the network. Hence the network is better at generalization and is less likely to overfit to the training data. We used input and output keep probabilities of 0.5 as our drop out. We also apply gradient clipping to our model. When gradients are backpropogated in time, there is an issue of exploding gradients.

Gradient clipping prevents this problem by clipping the gradients between two numbers to prevent it from getting very large. We employed gradient clipping with TensorFlow's clip_by_global_norm function. Given a gradient $grad_i$ its new value is calculated by $grad_i \frac{c}{max(GN,c)}$ where c is the clipping ratio (we used a value of 5 for this) and GN is the global norm defined as $\sqrt{\sum_{j=1}^{N} \|grad_j\|^2}$.

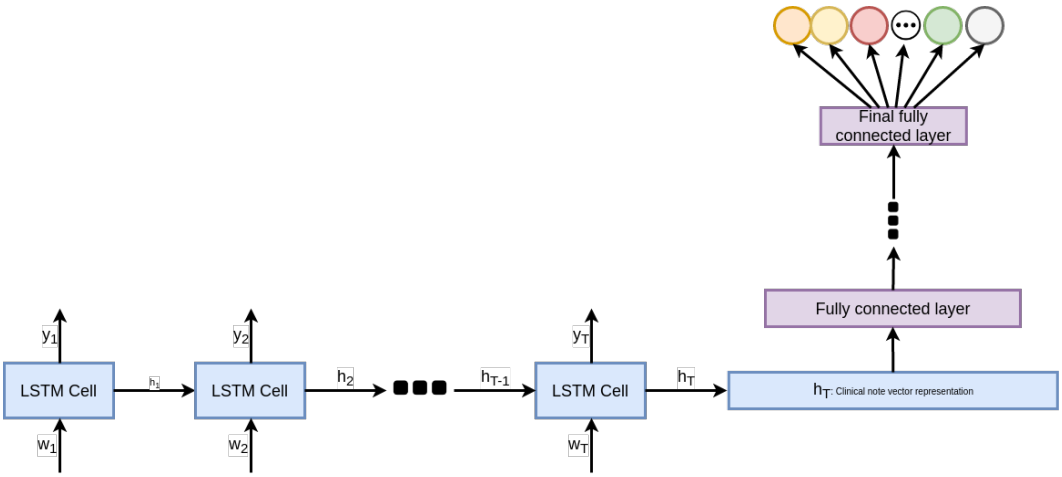


Figure 3: The main outline of the ICD-9 Tagger model

5 Results

All results are calculated for the same validation dataset, looking at the mean precision, recall, and F1-score across all samples. That is precision, recall, and F1-scores were calculated for each sample and averaged across all samples in the validation set. Due to time and hardware constraints we decided to focus on tuning the parameters max length (which controls the maximum word length of a note) the number of stacked LSTMs and batch size. Though we could have tried other parameter combinations (output and input keep probabilities on the LSTM, learning rate) we found parameters which boosted performance and stuck with those.

A quick summary of our impromptu search follows. our original settings were a batch size of 256, a max length of 500, no dropout, and a learning rate of 0.0001. We originally did not take the last true word, and instead always took the max length word when making predictions, which sometimes corresponded to 10s or 100s of steps of training on the NULL word vector of all zeros for each observation. Our F1-score around this time was .3, but when we chose to take the prediction at the last true word we improved our performance to an F1-score of 0.6. Finally, we added dropout to both the hidden state output and word vector input and noticed another improvement in our F1-score to similar values reported below. These preliminary results showed that it was important to not let the model predict on garbage inputs or else the final more accurate prediction is lost which harkens back to the quote "garbage in, garbage out". it was also observed that the model was over-fitting by around the 15th-20th epoch, which hinted at the need for regularization [17]. We settled on adding a keep probability of 0.5 to both the input and output of each LSTM cell.

We noticed a slight increase in performance when varying the max length parameter for both versions of the model 1. As the max length increases we tended to see an increase in precision, recall, and F1-score, however there was not a large difference between the model with a single LSTM cell and four stacked LSTM cells even though there was a increase in model complexity and training time. Though we are sure of observing these differences, they are so small that we cannot be sure that they're due only to model changes and not to the difference in the random initialization of weights. More runs are needed to confirm this.

Max length	LSTM Stacks	Precision	Recall	F1 Score
250	1	0.769	0.662	0.679
500	1	0.783	0.677	0.697
750	1	**0.799**	0.669	0.699
1,000	1	0.798	0.681	**0.708**
250	4	0.754	0.635	0.657
500	4	0.771	0.650	0.675
750	4	0.780	**0.685**	0.702
1,000	4	0.795	0.677	0.706

Table 1: Validation set performance of model when tuning Max Note Length and the number of stacked LSTM cells. Learning rate of 0.001, batch size of 256, and an input/output keep probability on the LSTM cell of 0.5 each

Because there was not a large jump in performance when using four stacked LSTM cells we decided to go with the more simple model when iterating over batch size, however because there seemed to be an upward trend in performance coinciding with the max length parameter we used a max length of 1,000. When varying batch size it was slightly more difficult to discern any pattern, but a batch size of 128 seemes to have lead to the best performance in terms of precision and F1-score while a batch size of 256 lad to the best recall which was closely followed by the recall for a batch size of 128.

Batch Size	Precision	Recall	F1 Score
64	0.811	0.675	0.710
128	**0.816**	0.680	**0.715**
256	0.798	**0.681**	0.708
512	0.807	0.656	0.696

Table 2: Validation set performance of a single stack LSTM, a max length of 1,000 and all parameters the same as table 1 using the same settings in Table 1 and keeping Max not length set to 1,000 while changing batch size

6 Discussion

We used word vector representations of physician clinical notes from MIMIC III database and trained LSTM networks to predict assign ICD-9 codes useful for billing purposes. We were able to achieve best precision of 0.799, recall : 0.685, F1 score : 0.708 among our best performing models which used a note length of 1000 words. We chose LSTM networks specifically for our task since we were specifically interested in learning long term temporal features of our notes. We achieved a considerable bump in our F1 scores when we changed our strategy by choosing the true last word prediction of each sentence rather than using the word corresponding to the maximum length (which changes due to the fact that we apply padding). In general, we also observed a marginal improvement in our performance by increasing the maximum length of notes used for training (1000). We probably don't achieve a large boost in performance by adding additional words as our model maybe forgetting previous information because LSTM models are capable of holding on to shot-term memory for a long time, but might not be able to hold on to long-term memory for our longer inputs. Introducing additional LSTMS as multiple layers did not really help in boosting our performance suggesting that other alternative architectures maybe required. We found that smaller batch sizes gave us better performance (size 128). This could probably attributed to the fact that we get more noise in the estimate of our gradient which is useful in pushing the model away from the shallow valleys (local minima) in the error function. We also observed that adding drop-out significantly improved our performance by reducing model overfitting, however at the expense of increased learning time. Overall, we tackled an important existing problem of clinical text annotation of ICD-9 codes by adopting a new deep learning approach which uses distributed vector representations of words. We think there is plenty of room for improvement in both the information representation and the model architecture used.

7 Future Directions

A major limitation of our data set is the occurrence of several misspellings. We represent all the words in the form of word vector representations. We get a standard vocab of pre-trained word vectors from Glove (Common crawl)[7] and look for words in our data set in the Glove set. Due to errors in our dataset, we miss representations of these misspelled words in our clinical notes. It is very likely that several of these words represent critical components of information that we are not able to capture. Hence, we need to account for these missing words by coming up with some matching strategy so that they are represented in our training information. This can be achieved through either a spell check step during preprocessing, or more involved deep learning approaches to understand how similar misspelled word vectors are to their correctly spelled counterparts.

We also need to be able to accurately learn more important words or text and hold them in memory for longer periods of time. For example, some medically relevant words are more critical in predicting the codes than others. By representing this information correctly (for eg. more weighting) and more emphasis on learning, we can hope to see improvement in our performance. A more complex architecture where a fully connected neural network learns the output of all LSTM networks before final prediction could be helpful in extracting the most relevant information.

We also believe that the learning task can be improved by using medically relevant dictionaries to obtain more pertinent word vectors. We could use databases such as ClinicalTrials.gov or UpToDate to obtain contextual information that would be more relevant for our task.

References

[1] https://dashboard.healthit.gov/evaluations/data-briefs/non-federal-acute-care-hospital-ehr-adoption-2008-2015.php

[2] https://www.cdc.gov/nchs/icd/icd9.htm

[3] Church, Kenneth Ward. "Word2Vec." Natural Language Engineering 23.01 (2016): 155-62. Web.

[4] Koby Crammer, Mark Dredze and Kuzman Ganchev and Partha Pratim Talukdar Automatic Code Assignment to Medical Text

[5] Ira Goldstein, M.B.A., Anna Arzumtsyan, M.L.S., and ozlem Uzuner, Ph.D Three Approaches to Automatic Assignment of ICD-9-CM Codes to Radiology Reports. AMIA 2007

[6] Richard Farkas, Gyorgy Szarvas. "Automatic construction of rule-based ICD-9-CM coding systems". BMC Bioinformatics 2008.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[8] Yitao Zhang A Hierarchical Approach to Encoding Medical Concepts for Clinical Notes Proceedings of the ACL-08: HLT Student Research Workshop (Companion Volume), pages 67–72

[9] Johnson, Alistair E.w., Tom J. Pollard, Lu Shen, Li-Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. "MIMIC-III, a Freely Accessible Critical Care Database." Scientific Data 3 (2016): 160035. Web.

[10] Alan R. Aronson1, Olivier Bodenreider1, Dina Demner-Fushman1, Kin Wah Fung1, Vivian K. Lee1,2, James G. Mork1, Aurelie Neveol1, Lee Peters1, Willie J. Rogers From Indexing the Biomedical Literature to Coding Clinical Text: Experience with MTI and Machine Learning Approaches. BioNLP 2007: Biological, translational, and clinical language processing, pages 105–112

[11] Zhang, M. and Zhi-Hua Z. (2014) A review on multi-label learning algorithms. Knowledge and Data Engineering, IEEE Transactions 26.8: 1819-1837.

[12] Priyanka Nigam Applying Deep Learning to ICD-9 Multi-label Classification from Medical Records cs224d Class paper presentation. 2015

[13] Sepp Hochreiter. Long Shert Term Memory Neural Computation. (1997):1735 - 1780

[14] Perotte, Adler, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noaomie Elhadad. "Diagnosis Code Assignment: Models and Evaluation Metrics." Journal of the American Medical Informatics Association 21.2 (2014): 231-37. Web.

[15] Zhang, Min-Ling, and Zhi-Hua Zhou. "A Review on Multi-Label Learning Algorithms." IEEE Transactions on Knowledge and Data Engineering 26.8 (2014): 1819-837. Web

[16] Benesch, C., D. M. Witter, A. L. Wilder, P. W. Duncan, G. P. Samsa, and D. B. Matchar. "Inaccuracy of the International Classification of Diseases (ICD-9-CM) in Identifying the Diagnosis of Ischemic Cerebrovascular Disease." Neurology 49.3 (1997): 660-64. Web.

[17] Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors." ArXiv Preprint ArXiv:1207.0580, 2012.

[18] Lita, Lucian Vlad, Yu Shipeng, Stefen Niculescu, and Jinbo Bi. "Large Scale Diagnostic Code Classification for Medical Patient Records." International Joint Conference on Natural Language Processing (2008): n. pag. Web.