NLP for Clinical Text

Masami Peak
masamip2@illinois.edu
https://mediaspace.illinois.edu/media/t/1_jzhcxgok

ABSTRACT

Clinical notes are the most important text information for each patient that is created by clinicians. They describe the patient's medical condition including signs, symptoms, treatment, and drugs given. Annotating medical codes is done by labor to abstract the medical information from the clinical notes and it assigns the medical codes representing the diagnoses and treatments to each patient encounter. This process relies on the medical coders' knowledge of the medical terminology for both procedures and diagnoses. Also, the accuracy of medical coding depends on the medical coders' ability: performing detail-oriented work and being motivated to learn any new or reorganized codes. I present and experiment with a convolutional neural network based on an attention mechanism that predicts medical codes from clinical text.

1. INTRODUCTION

Convolutional Attention for Multi-Label classification (CAML) is a decision support system for solving a multilabel text classification problem. For example, documents may be associated with multiple class labels and there is a Bayesian classification approach in which the multiple classes that comprise a document are represented by a mixture model (McCallum, 1999). The method in the CAML uses an attention mechanism per label that assigns interpretable important values to n-grams in the document as input for each code. The clinical text of all discharge summaries is preprocessed by removing non-alphabetic characters, lowercasing, and tokenizing them to words. Then, word embeddings of size 100 are generated using the Word2Vec CBOW (Continuous Bag-of-Words) method (Mikolov et al., 2013) on the text. The CBOW model uses the continuously distributed representation of the context to predict the current word. The model in the CAML is trained and evaluated on the documents containing at least one of the top 50 frequent labels. Many codes are only assigned to a few patient visits. Because of the sparsity of data, it is very difficult to train a model that can predict all of the codes accurately. Instead, the 50 most frequent codes are chosen for the study, while the model can be extended to other codes with the condition that training data is sufficient (Shi et al., 2017). The CAML that combines convolution and attention is similar to the introduction of an attentional neural network that applies convolution on the input word tokens to detect time-invariant and long-range current local attention in a context-dependent way (Allamanis et al., 2016). Furthermore, the attention mechanism itself is like a hierarchical attention network to classify documents in the hierarchical structure. The network is applied at the word and sentence levels depending on the importance of the content (Yang et al. 2016). In contrast, Grounded Recurrent Neural Network (Vani et al., 2017) is a modified Gated Recurrent Unit (GRU) with dimensions to predict the presence of labels and evaluate on a 5,000-code subset of ICD-9. The paper (Baumel et al., 2018) also applies the GRU with the hierarchical sentence and word attention (the HA-GRU) to classify ICD-9 diagnosis codes.

2. APPROACH

Based on the method, Convolutional Attention for Multi-Label classification (CAML), I perform some experiments on the CAML. I use a method to pass text through a Convolutional Neural Network (CNN) for each document per label to solve a multilabel classification problem. In this method, I apply an attention mechanism to select the sequences of words in each document that are most related to each possible medical code. The attention calculated based on the convolution is applied to the convolution and summing is used for computing a single vector for all of the 50 labels. To compare with the CAML, I also present 2 variants of Recurrent Neural Network (RNN). Long Short-Term Memory (LSTM) (Hochreiter, 1997) consists of 3 gates: forget, input, and output gates, with 2 states: cell state holding long-term memory and hidden state holding short-term memory. Gated Recurrent Unit (GRU) (Cho, et al., 2014) consists of 2 gates: reset and update gates, with 1 state: hidden state holding short-term memory and long-term memory. In both methods, each input word is passed through the layers in those gates and interacts with those states. I use the last hidden state of the last word in each text in the last layers for each document per label to solve a multilabel classification problem. Due to the capability of capturing long-range semantics in text, both the LSTM and the GRU are widely used for modeling languages and encoding sequences. In this study, I use the same preprocessed and pre-trained word embeddings as a mapping table for the text input to compare the 3 models.

3. METHODS/METRICS

3.1 Datasets and Preprocessing

I use the NOTEEVENTS, PROCEDURES_ICD, and DIAGNOSES_ICD tables from MIMIC-III v1.4 Clinical Database, and here is an overview of the 3 tables.

NOTEEVENTS: Contains clinical notes for each patient's hospital admission HADM_ID.

PROCEDURES_ICD: Contains ICD-9 procedures for each patient's hospital admission HADM_ID.

DIAGNOSES_ICD: Contains ICD-9 diagnoses for each patient's hospital admission HADM_ID.

The dataset is a freely available large database comprising de-identified health-related data associated with unique 58,976 patient stays in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. The current version of the database is v1.4 and was released on 2 September 2016. There are 9,017 unique ICD-9 codes from the 2 datasets, including 2,032 procedure codes and 6,985 diagnosis codes. I focus on 59,652 records of CATEGORY = 'Discharge summary'. Each clinical text as a document in the specific category has condensed information about a stay. Each document is tokenized and lowercased after NA, '\n', '\r', punctuation, multiple spaces, and numbers are removed.

HADM_ID	LENGTH	TEXT	ICD9_CODE
164627	10512	admission date discharge...	V43.64;v45.81;96.6;...
128930	8788	admission date discharge...	599.0;296.80;333.94;...

Table 1. *Sample Data in the Preprocessed Dataset*

WORD	VECTOR (size=100)			
brittle	0.0038737233262509108	-0.10465587675571442	0.19580629467964172	…
diabetic	-0.48406124114990234	0.9063760638237	0.946773886680603	…

Table 2. *Sample Data in the Word Embeddings*

Alphanumeric characters remain in the documents. The clinical notes with the same HADM_ID are concatenated by a space ' ', and the corresponding medical codes are concatenated by a semicolon ';'. Finally, the ICD9_CODE (medical codes) and the TEXT (clinical notes) are merged based on 52,726 unique HADM_IDs. The length of each tokenized clinical text is also added as LENGTH. **Table 1** shows 2 samples in the preprocessed dataset.

Next, I select the 50 most frequent codes that are assigned to the clinical notes, because many codes are only assigned to a few clinical notes and it is difficult to train a model on the sparse data. For all the text, I build the embeddings and the vocabulary using the Word2Vec CBOW method with the parameters, feature vector_size = 100, kernel_size = 5, and min_count = 3. The vocabulary is based on the frequency of each word in all of the documents. The embeddings are vector representations of the words and similar words are located closely (cosine similarity). For example, one of the similar words to 'diabetic' is 'brittle' (the similarity score: 0.641) which is related to the term 'brittle diabetes', meaning hard to control diabetes. Another example is 'gastroparesis' (the similarity score: 0.575) which is a type of nerve damage that slows digestion and can be caused by high blood sugar levels from diabetes. Once the embeddings and the vocabulary have been generated, the clinical text is truncated to a maximum of 2,500 words based on the frequency of each word. Then, I filter the merged dataset by the top 50 medical codes, so that each clinical text has at least one of the top 50 medical codes. The filtered dataset is split into 44,655 train, 3,473 validation, and 1,489 test datasets. Alphabetically sorted word embeddings are constructed using the vocabulary and the embeddings generated by the Word2Vec. Each index in the newly structured word embeddings matches the index of the word in the alphabetically sorted vocabulary. **Table 2** shows 2 samples in the word embeddings. When loading the dataset, each word token in TEXT is mapped to the index in the alphabetically sorted vocabulary and 0 is assigned to unknown words. Similarly, each code in ICD9_CODE is mapped to the index in the alphabetically sorted top 50 codes, and each set of codes per clinical text is represented as a multi-hot vector.

3.2 Model Design

3.2.1 CAML

In the CAML architecture shown in **Figure 1**, each batch of text is passed through the 5 layers.

3.2.1.1 Word Embeddings Layer

Initial corresponding weight vectors are generated using the word embeddings. The vector for an unknown word at index 0 is replaced by a random vector of Gaussian embeddings as noises. The weights are all normalized to be on the same scale where the learning rate will be proportional to the magnitude of the weights. Although, due to the parameter padding_idx = 0, the padding vector at index 0 will not contribute to the gradient and not be updated during training. Each vector of embeddings with its corresponding weights is looked up based on the index representation of each input word in each row of TEXT. Dropout is then performed to avoid overfitting.

3.2.1.2 1D Convolution Layer

The weights are initialized by Xavier using uniform distribution: Range = Gain * sqrt(6 / (In_Channels + Out_Channels)), to be in the range (-Range, Range), and applied to the transposed word embedding matrix. As the kernel for the convolution slides across each row of TEXT, each sequence within the current kernel is summed after element-wise multiplication by the weights, and addition by the default biases are performed.

$$h_n = \text{Sum}(W_c * x^T_{n:n+k-1} + b_c)$$

The output length is: Out_Length = ((In_Length – Kernel_Size + 2 * Padding) / Stride) + 1, and the parameter out_channels is specified as a half size of the kernel. Then, tanh (Hyperbolic Tangent) operation is applied to output in the range (-1, 1).

$$H = \text{Tanh}(H)$$

3.2.1.3 Attention Calculation Layer

Similarly, the weights are initialized by Xavier using a uniform distribution. Matrix multiplication is applied to the weights and the outputs from the previous 1D convolution layer, then exponential normalization is applied to the calculated attention using softmax operation. The sum of the attention to the convoluted values for each text per medical code will be 1.

$$\alpha_\ell = \text{SoftMax}(W_\ell \cdot H)$$

3.2.1.4 Attention Application Layer

Matrix multiplication is applied to the attention matrix from the previous layer and the transposed output matrix from the 1D convolution layer.

$$v_\ell = \Sigma_n \, \alpha_{\ell n} \cdot h^T_n$$

3.2.5 Classification Layer

Similar to the other layers, the weights are initialized by Xavier using a uniform distribution. The outputs with the attentions from the previous layer are multiplied by the current layer's weights and summed at the 2nd dimension. Bias: Range = 1 / In_Features, to be in the range (- sqrt(Range), sqrt(Range)), is then added to each sum. Finally, the sigmoid function is applied to predict the probabilities.

$$y'_\ell = \text{Sigmoid}(\text{Sum}(W_{\ell 2} * v_\ell) + b_\ell)$$

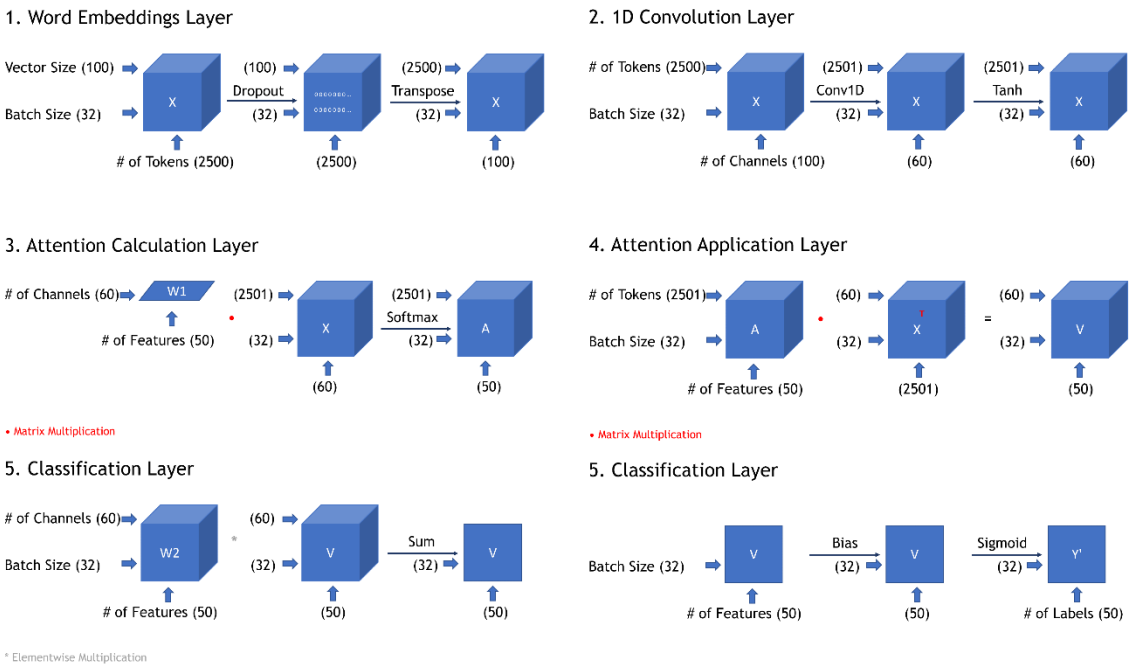

Figure 1. *CAML Architecture*

3.2.2 LSTM

In the LSTM architecture as shown in **Figure 2**, each input word is passed through the layers in the 3 gates: forget, input, and output gates, and interacts with the 2 states: cell state holding long-term memory and hidden state holding short-term memory.

3.2.2.1 Forget Gate

Forget gate decides which data should be kept in the long-term memory (cell state) from the previous time step based on the short-term memory (hidden state) from the previous time step and the current input, which are passed through a sigmoid function with the weights and the biases.

$$f_t = \text{Sigmoid}(W_f \cdot (h_{t-1}, x_t) + b_f)$$

$$C'_t = C_{t-1} * f_t$$

3.2.2.2 Input Gate

Input gate decides what new data should be stored in the long-term memory based on the multiplication of the outputs through a sigmoid function and a tanh function. The output in this layer becomes a new long-term memory to be carried over to the next time step after the output from the forget gate is added.

$$i_1 = \text{Sigmoid}(W_{i1} \cdot (h_{t-1}, x_t) + b_{i1})$$

$$i_2 = \text{Tanh}(W_{i2} \cdot (h_{t-1}, x_t) + b_{i2})$$

$$C_t = i_1 * i_2 + C'_t$$

3.2.2.3 Output Gate

Output gate decides what new data should be stored in the short-term memory based on the multiplication of the outputs through a sigmoid function and the new long-term memory that is passed through a tanh function. The short-term memory is carried over to the next time step.

$$o_1 = \text{Sigmoid}(W_{o1} \cdot (h_{t-1}, x_t) + b_{o1})$$

$$o_2 = \text{Tanh}(W_{o2} \cdot (C_t) + b_{o2})$$

$$h_t, o_t = o_1 * o_2$$

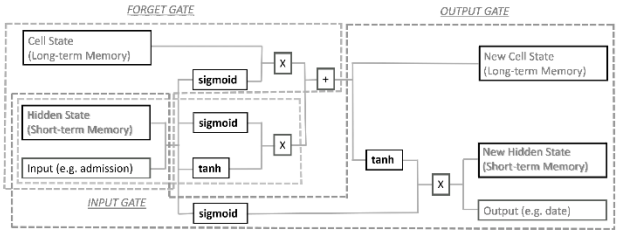

Figure 2. *LSTM Architecture*

3.2.3 GRU

In the GRU architecture, as shown in **Figure 3**, each input word is passed through the layers in the 2 gates: reset and update gates, and interacts with the 1 state: hidden state holding short-term memory and long-term memory.

3.2.3.1 Reset Gate

The reset gate decides which data should be used based on the hidden state (the long-term and the short-term memory) from the previous time step and the current input, which are passed through a sigmoid function, and a tanh function with the weights.

$$g_r = \text{Sigmoid}(W_{rh1} \cdot (h_{t-1}) + W_{ri1} \cdot (x_t))$$

$$r = \text{Tanh}(g_r\ (W_{rh2} \cdot (h_{t-1})) + W_{ri2} \cdot (x_t))$$

3.2.3.2 Update Gate

Update gate decides which data should be stored in the hidden state based on the output through a sigmoid function, the hidden state

from the previous time step, and the output from the reset gate. The hidden state is carried over to the next time step.

$$g_u = \text{Sigmoid}(W_{uh1} \cdot (h_{t-1}) + W_{ui1} \cdot (x_t))$$
$$u = g_u * h_{t-1}$$
$$h_t = r * (1 - g_u) + u$$

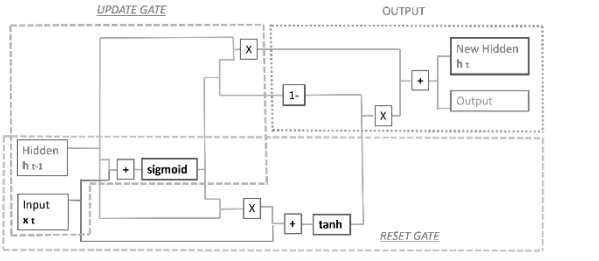

Figure 3. *GRU Architecture*

3.3 Training

I use binary cross-entropy loss for each label's loss per text to be calculated and Adaptive Moment Estimation (Adam) optimizer. Adam uses optimized gradient descent benefitted from 2 algorithms: Momentum takes the exponentially weighted average of the gradients, and Root Mean Square Propagation (RMSP) takes the exponential moving average of the gradients.

3.4 Evaluation Metrics

I use a variety of evaluation metrics with micro averaging, which calculates metrics globally by counting the total TP (True Positives), FP (False Positives), and FN (False Negatives). I compute Precision: TP / (TP + FP), Recall: TP / (TP + FN), Fβ (more general F1-Score): $(1 + \beta^2) \cdot (\text{Precision} \cdot \text{Recall}) / ((\beta^2 \cdot \text{Precision}) + \text{Recall})$, which is a weighted harmonic mean of Precision and Recall, and ROC_AUC (Area Under the Receiver Operating Characteristic Curve) using a curve created by plotting TPR (True Positive Rate) and FPR (False Positive Rate).

4. EXPERIMENTAL RESULTS

I focus on evaluating the performance of predicting the top 50 ICD-9 codes based on the clinical text of the discharge summaries in the MIMIC-III. To maximize each performance, the learning rate of the Adam optimizer for each model has to be tuned. The CAML model shows the best results on all metrics. It indicates that the CAML's convolutional attention mechanism, where the algorithm considers that the neighboring words are more important factors to predict the medical codes, works well in this specific setting. The GRU shows the second-best after the LSTM. **Table 3** shows the comparison in prediction performance between the 3 models. The GRU performs much better than the LSTM, while both models are variants of the RNN. Two of the noticeable differences between the LSTM and the GRU are that the number of gates and the number of states. The LSTM has 3 gates, whereas the GRU has 2 gates. Since all types of weights within each gate will be updated during the back-propagation of the model training, the LSTM needs to take care of the greater number of weights. Additionally, the LSTM deals with long-term memory and short-term memory as 2 separate states. With all of these aspects, the LSTM is less likely to consider the neighboring words as important factors than the GRU is. On the other hand, the GRU mechanism handles 1 state from the previous time step and the current input together, which would behave more like the CAML mechanism. Furthermore, the CAML has especially better performance in Recall compared to the GRU, which means that the CAML has fewer False Negatives. This proofs that the mechanism where the attention is focused on the neighboring words is more likely to increase the confidence to make the True Positive decisions.

5. DISCUSSION

5.1 Model Learning

Figure 4 provides each count of TP, FP, and FN for the top 50 ICD-9 codes predicted by the CAML on 1,489 test data. The ICD-9 code labels are in descending order of frequency. The code with a larger difference between TP with more counts and FP/FN with fewer counts means that the model has been trained well on the specific code (e.g., ICD-9 Codes: 401.9, 427.31). On the other hand, the code with a larger difference between TP with fewer counts and FP/FN with more counts means that the model has not been trained well on the specific code (e.g., ICD-9 Codes: 99.04, 285.9). The results of TP with more counts and FP/FN with fewer counts per label in Figure 4 reflect the CAML's best performance in Table 3.

Figure 5 provides the prediction results by the LSTM in the same setting as Figure 4. The model has not been trained well on almost all of the top 50 codes. Each FN count has higher numbers, which means that the model is hardly able to predict anything.

Figure 6 provides the prediction results by the GRU in the same setting as Figure 4. The model has been trained better on more frequent codes (e.g., 401.9, 427.31) as there are larger differences between TP with more counts and FP/FN with fewer counts. Interestingly, both the CAML and the GRU have similar patterns. For example, both models can predict well on the ICD-9 CODE = 427.31, which is Atrial Fibrillation related to heart disease, while the models do not perform well on the ICD-9 CODE = 99.04, which is Transfusion of Packed Cell (red blood cells that have been separated for blood transfusion).

5.2 Interpretability

Figure 7 shows an example of text (HADM_ID = 155451), including ICD-9 CODE = 427.31, Atrial Fibrillation related to heart disease, as one of the ICD-9 codes annotated to the clinical text, with precision = 0.92 predicted by the CAML. The attention, which is generated by the convolutions with the kernel_size = 18 for the sequences of the specific words (around the underlined italic words), affects the accuracy of multilabel classification. Thus, the CAML can identify 'atrial', 'fibrillation', 'heart', and 'respiratory'

Model	Learning Rate	Precision	Recall	Fβ	ROC_AUC
CAML	0.001	**0.72**	**0.65**	**0.68**	**0.94**
LSTM	0.01	0.64	0.12	0.19	0.75
GRU	0.01	**0.72**	0.56	0.62	0.92

Table 3. *The Top 50 ICD-9 Codes Prediction Performance on MIMIC-III Discharge Summaries*

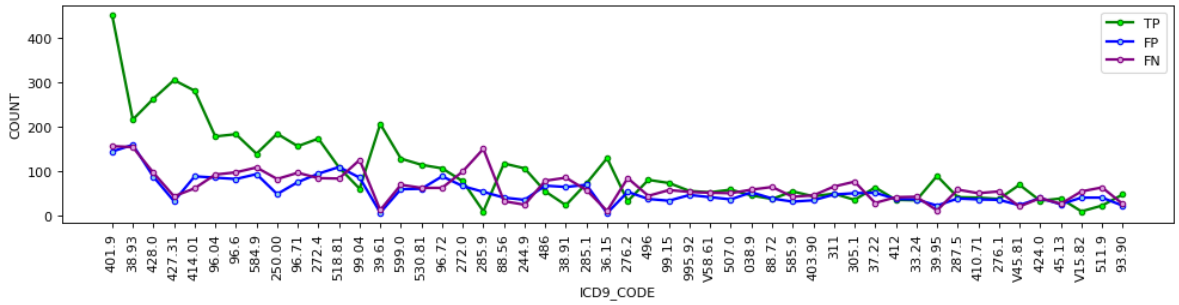

Figure 4. *The Count of the Top 50 ICD-9 Codes Predicted by the CAML*

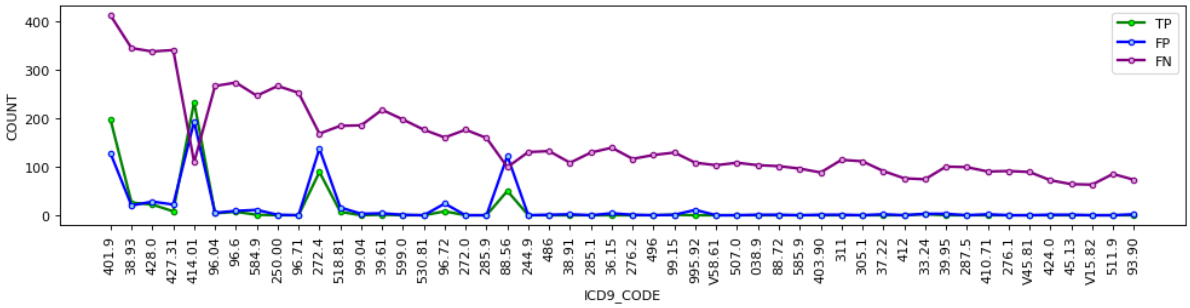

Figure 5. *The Count of the Top 50 ICD-9 Codes Predicted by the LSTM*

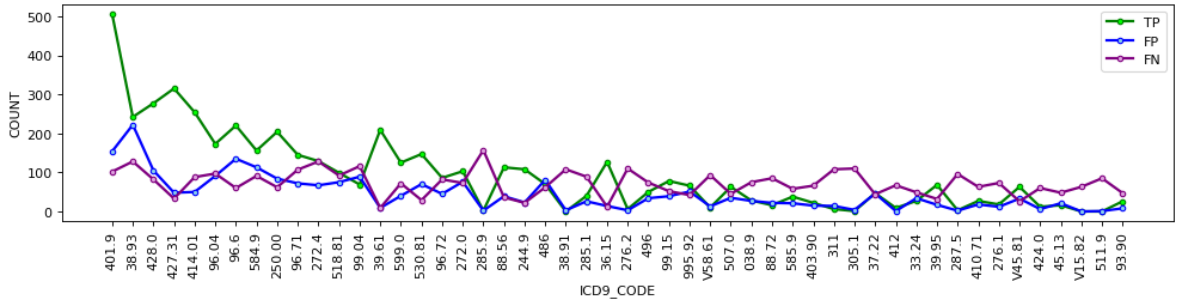

Figure 6. *The Count of the Top 50 ICD-9 Codes Predicted by the GRU*

(a similar word to 'heart' with the similarity score: 0.538 found by Word2Vec) are important words to ICD-9 CODE = 427.31. Furthermore, this demonstrates the interpretable predictions made by the CAML.

5.3 Performance Evaluation

Figure 8 provides the count of ICD-9 CODE = 427.31 annotated to or predicted for the text containing 'atrial', 'fibrillation', 'heart' or similar words to these words by the CAML. BOTH means the code is annotated and predicted, PRED means the code is predicted but not originally annotated, and ORIG means the code is originally annotated but not predicted. In almost all the cases where the 3 lines, BOTH, PRED, and ORIG, indicate the counts close to zero at a particular word means the model can predict the medical code for the corresponding clinical text containing the word. The overall results show that the CAML can recognize those words and the similar words are related to ICD-9 CODE = 427.31. When PRED has more counts, that means there are more FP (False Positives) and some of the neighboring words interact to increase the confidence to make the decision. When ORIG has more counts, that means there are more FN (False Negatives) and the neighboring words are not affected.

Moreover, 'sed' stands for 'Sedimentation' and the sed rate measures the rate at which red blood cells become sediment. That tells it could be difficult to predict ICD-9 codes perfectly by the medical terms that are shared in multiple medical procedures and diagnoses. However, further study in neighboring words, including

admission date discharge date service <words abbreviated> when the *patient was started on lasix for congestive heart failure atrial fibrillation per patient wishes the patient has not* been anticoagulated with coumadin the patient *has a history of longstanding atrial fibrillation and presented with atrial fibrillation with rapid ventricular response on lopressor* the patient <words abbreviated> he was electively *intubated at that point for respiratory distress past medical history includes atrial fibrillation with previous refusals to take* anticoagulation medicines *hypertension* <words abbreviated> micu service he *was degrees fahrenheit with a heart rate of blood pressure of respiratory rate of and satting his ventilations* settings were <words abbreviated>

Figure 7. *An Example of Affective Sequence of Words with ICD-9 CODE = 427.31*

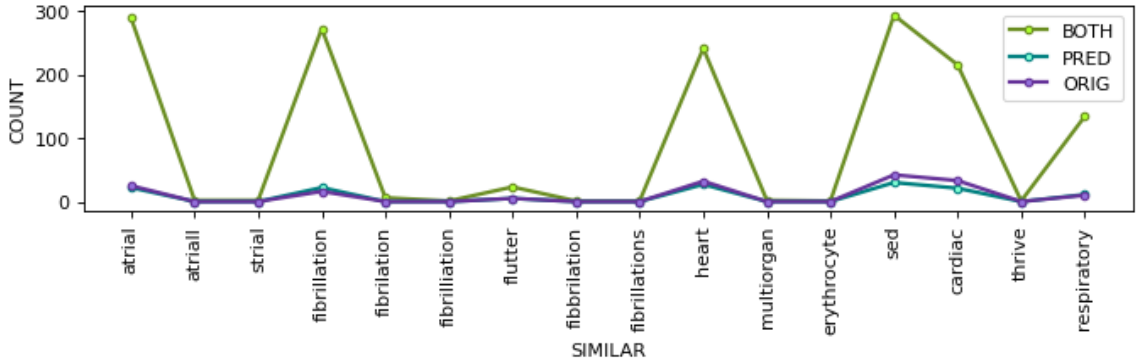

Figure 8. *The Count of ICD-9 CODE = 427.31 Annotated to or Predicted for the Text Containing 'atrial', 'fibrillation', 'heart' or the Similar Words to these words by the CAML*

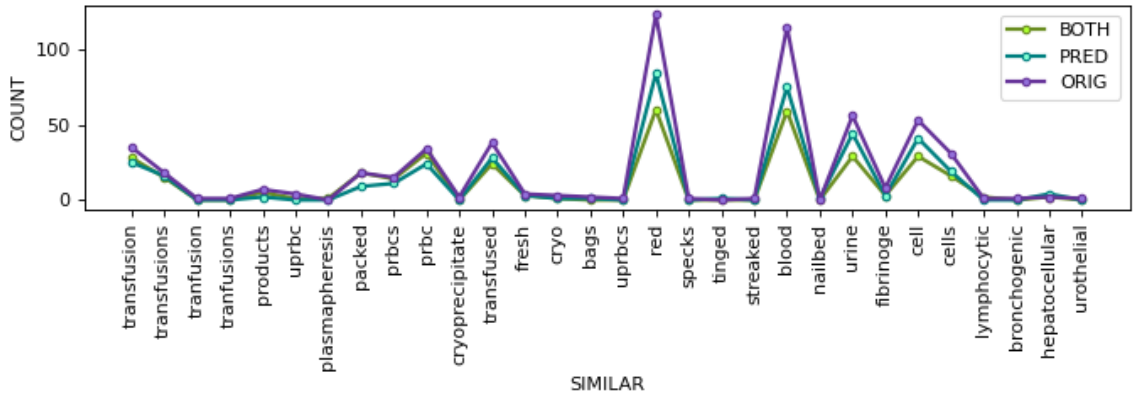

Figure 9. *The Count of ICD-9 CODE = 99.04 Annotated to or Predicted for the Text Containing 'transfusion', 'packed', 'red', 'blood', 'cell' or the Similar Words to these words by the CAML*

data preprocessing improvement and introduction of 'stop_words', could potentially improve the CAML.

Figure 9 provides the count of ICD-9 CODE = 99.04 annotated to or predicted for the text containing 'transfusion', 'packed', 'red', 'blood', 'cell' or the similar words to these words by the CAML. The code is for Transfusion of Packed Cell (red blood cells that have been separated for blood transfusion), and both the CAML and the GRU do not perform well on this code as mentioned in the section related to Figure 4. Interestingly, all the 3 lines, BOTH, PRED, and ORIG, form very similar shapes. This indicates that each word may not be a good factor for predicting the code. For example, 'red' and 'blood' are too generic words as medical terms. In the case where commonly used medical words are the only words that can describe the specific medical code, there should be new supporting methods, business processes, or clinical note composing techniques.

6. CONCLUSION AND OPTIMIZATION

I perform some experiments with the CAML (Convolutional Attention for Multi-Label classification) which uses an attention mechanism where the output values from the convolution operations act as important values to n-grams in input text for relating each possible clinical code. In convolution operation, I use a parameter kernel_size = 18 to slide across the maximum of 2,500 words, meaning each sequence of 18 words can influence to determine each possible ICD-9 code. One thing to be noted is that I select the top 2,500 frequent words if the length of the clinical text is larger than 2,500, instead of truncating all the words after the limit, for avoiding important words from being excluded. The CNN (Convolutional Neural Network) based model CAML outperforms the variants of RNN (Recurrent Neural Networks). Despite being the popular solutions for NLP (Natural Language Processing) problems, the LSTM (Long Short-Term Memory) and the GRU (Gated Recurrent Unit), which are the RNN based models, do not perform better than the CAML in this specific setting for this study. Both models carry information of each word over different time steps. Especially in LSTM architecture, the maximum of 2,500 words could be too long to manage the gate mechanism for retaining and discarding the short-term memory and the long-term memory. Compared to the LSTM, the GRU shows good performance because the GRU has a mechanism that tends to put more weight on the neighboring words as important factors. That results in producing a similar effect as the CAML architecture. In addition to the future optimization ideas I mentioned, each performance of the 3 models could be improved with more organized and formatted datasets, and automated ICD coding would become a more accurate supporting tool.

7. REFERENCES

[1] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable Prediction of Medical Codes from Clinical Text.

[2] Andrew Kahites McCallum. 1999. Multi-Label Text Classification with a Mixture Model Trained by EM.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeff Dean. 2013. Efficient estimation of word representations in vector space.

[4] Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P. Xing. 2017. Towards Automated ICD Coding Using Deep Learning.

[5] Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A Convolutional Attention Network for Extreme Summarization of Source Code.

[6] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification.

[7] Ankit Vani, Yacine Jernite, and David Sontag. 2017. Grounded Recurrent Neural Networks.

[8] Tal Baumel, Jumana Nassour-Kassis, Raphael Cohen, Michael Elhadad, and Noémie Elhadad. 2017. Multi-Label Classification of Patient Notes: Case Study on ICD Code Assignment.

[9] Sepp Hochreiter, Jurgen Schmidhuber. 1997. Long Short-Term Memory.

[10] Kyunghyun Cho, Bart van Merrienboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.