Virginia Garcia NLP Project

TEAM: MARIO ARDITO, STEVEN CRAVEN, ANDREW WILLIAMSON, RICKY TRUONG

ADVISOR: RICH PHELAN

Contents

Who We Are

The team is part of Portland State University's Master of Applied Data Science for Business (MS-ADSB). We are a team with a diverse background that spans several different industries including education, accounting, and logistics. As a part of our capstone project for the MS-ADSB program, we have been contracted to implement a data science project to help solve a business problem. We have partnered with Virginia Garcia Memorial Health Center (VG) to help them create a natural language processing (NLP) model.

The Mission and Vision of the Virginia Garcia Memorial Health Center

Mission: To provide high quality, comprehensive, and culturally appropriate primary health care to the communities of Washington and Yamhill counties with a special emphasis on migrant and seasonal farmworkers and others with barriers to receiving health care.

Vision: Our vision is simple. We work to ensure that all people in our service area have access to timely, high-quality health services, enabling them to achieve the best possible health outcome.

Since the death of Virginia Garcia due a lack of access to medical care, the focus of VG has been to provide equitable access to medical care to disenfranchised communities such as migrant workers. The purpose of this project is to help VG explore an avenue of increasing operational efficiencies through artificial intelligence/machine learning. We believe that incorporating machine learning into the medical coding/billing process can increase the operational efficiency of VG and enable the clinic to better serve its patients.

Problem statement

Misclassification of diagnostic coding causes claims denials if diagnosis codes do not align with procedures billed. This causes financial burdens to patients when they receive unexpected bills and potential loss of revenue and funding for Virginia Garcia. Miscoding also skews and misrepresents the true risk of VG's patient population to outside agencies such as insurance. Misreporting may also impact access to funding such as grants.

The group explored a variety of machine learning models to identify ICD-10 codes based on patient notes that VG can possibly integrate into their systems.

Approach to Project

Interviews

Our group initiated the project by conducting interviews to gather insight and information, determine the skills required, and brainstorm potential techniques that could be used in developing our model. We met individually and as a group with various professionals: a medical coder, physician, data analyst, and financial controller from Virgina Garcia, as well as with an external data scientist and external physician. Additionally, we leaned on Kavitha from Virginia Garcia as well as our advisor, Rich, to ensure we were on the right track. As none of us came from medical backgrounds, and as still burgeoning data science students, the knowledge gleaned through interviews was instrumental in developing our project. These interviews and conversations provided insight and recommendations while also helping our team better understand what was in our scope and outside of it, realistic and optimistic.

Data Set

Due to limitations around having access to actual patient data, we needed to create our own data set with which our model would be trained on. With a knowledge foundation of what was needed for our group to be successful, we relied on generative AI programs, ChatGPT, Gemini, and Claude to help amass unique patient chart notes and accompanying diagnosis codes. As advised, we focused on one diagnosis category, diabetes, to narrow our focus. Further, we utilized varying generative AI programs to diversify the “voices” of the patient chart notes as if to represent differences between different physicians. Over the course of the project, our dataset grew to roughly 4,500 unique patient chart notes.

Choosing and Training a Model

Our group utilized Python to build our model as it is flexible, user friendly, and familiar to all the team. Python can be collaborative as well. When utilized through DeepNote, for example, each team member can view and edit code. Moreover, Python has a large variety of easily implementable open-source libraries and extensions to help facilitate building an NLP, or Natural Language Processing model, which was our model of choice. NLPs allow for computers to interpret, analyze, and manipulate human language inputs (e.g. patient chart notes). Our NLP used logistic regression as it is useful in text classification assignments such as classifying key descriptors in patient chart notes with the proper physician determined diagnosis codes. With a basis established, creating an NLP model using Python, each group member built their own model, collaborating along the way with insight, suggestions, and troubleshooting.

Interviews and Research

The group began its primary research by interviewing several people within VG to gain an understanding of its current practices, competencies, and opportunities.

Medical Coders

The coders at VG currently lack access to analytics tools that would help them flag patient charts and to identify medical coding errors. They rely on basic reports to identify denied claims and generic reports from the data department. Their workflow involves reviewing charts after a provider enters notes and selects ICD10 codes. EPIC flags some billing errors, but many coding errors go unnoticed. These errors can lead to missed revenue, misrepresentation of patient data, and difficulties for patients when they switch providers.

The biggest challenge for VG's coding process is the lack of a process to track the coding errors and a way to include charts with errors into the work queue. Coders rely on manual chart review and claim denials to identify errors. This is time-consuming and inefficient. Improved communication and collaboration between the data department and the coding department could lead to better reporting and error identification.

There are several opportunities for improvement. VG could implement software that flags potential coding errors within EPIC. Additionally, coders could have access to more specific reports that identify trends and areas with high error rates. Finally, improving communication between departments could lead to better data sharing and training for both coders and providers.

Physicians

To help understand the situation from the root, our group interviewed a physician from Virgina Garcia, Dr. Carden as well as a physician from Reed College, Dr. Gopal. Dr. Carden shared that in his fifteen years or so with VG, he had only had maybe two cases come back to him for review due to diagnosis/medical coding errors. He explained that for the most part, once he diagnoses and sends off his evaluation, it is mostly out of his hands at that point and that mistakes happen on the back end rather than the front end.

Despite this, he shared that he saw immense value in using technology to help catch these mistakes as, in his words, errors can be “catastrophic” for the patients he works with. For example, denied claims due to errors can be expensive, time consuming, and stressful for VG’s

patient population who are already underserved as primarily migrant workers. Being able to prevent claims being dismissed would make a huge difference.

Additionally, one recommendation Dr. Carden shared as we discussed data science and technology in medical settings, would be for physicians to have access to speech to text software as opposed to typing up notes. He shared that this would expedite the time spent doing administrative work as opposed to patient facing work. Further, it would be possible to include speech to text in an NLP model: the physician's words would feed directly into the database the NLP relies on.

Controller

Anthony Perez, who has been with Virginia Garcia Medical Center (VG) for several years, discussed his responsibilities, which include ensuring adherence to accounting policies and reviewing monthly accounting tasks and journal entries. While he is not an expert in medical coding, he provided insights into the financial implications of coding practices at VG. However, there are currently no metrics to measure the impact of misclassified or rejected codes on financials, although Perez acknowledged that such metrics could be developed.

Perez highlighted that due to staffing shortages, medical coders often do not have the time to correct misclassifications within the one-month correction window, leading to potential revenue loss. Misclassifications and rejected codes also delay the revenue collection process, impacting financial and budget forecasting. He emphasized the importance of accurate coding, noting that correct classification can result in higher reimbursement rates for complex procedures and participation in quality improvement programs, which offer financial incentives for adherence to preventive care and chronic disease management protocols.

The interview with Anthony Perez highlighted the impact of medical coding accuracy on VG's financial health. Key issues include the lack of current metrics to track the financial impact of coding errors, staffing shortages that hinder the correction of misclassified codes, and the financial incentives tied to accurate coding. Improved transparency and detailed reporting on denied codes were identified as critical areas for development to enhance financial management and revenue collection.

Data Scientist

An interview with William Stadtlander, a data scientist at Virginia Garcia Memorial Health Center, provided valuable insights into the application of data science in healthcare, particularly

in predictive modeling and operational data management. Will, who earned a master's in biostatistics from OHSU, PSU School of Public Health in 2021, detailed his role in developing predictive models to improve patient care outcomes.

A key project he mentioned involved developing models to predict patient readmissions within 30 days post-discharge, which is crucial for assessing hospital performance. He also shared his expertise in using R for statistical analysis, SQL for managing databases, and Tableau for creating interactive dashboards. Although Will has limited direct experience with medical coding, he emphasized the importance of understanding the data structures and workflows that support operational efficiency in healthcare.

The conversation also discussed the significant challenges of maintaining data privacy and adhering to ethical standards in managing sensitive patient data. Will highlighted the necessity of local data processing to ensure compliance with privacy standards and discussed the restricted use of APIs to maintain data security. He provided valuable resources for the project, such as using regular expressions for data parsing and the Natural Language Toolkit (NLTK) for text data processing, which will be crucial for handling ICD-10 coding.

Methodology

NLP Model

To support Virginia Garcia in the goal to reduce medical coding errors, a Natural Language Processing model was selected as the best approach to do so. Because NLPs can process human language, they can be very effective in situations like ours where human text is inputted regularly. We used Python as the programming language of choice where our project was built. Python was familiar to each group member, can be used collaboratively, and features an ever-growing arsenal of libraries and extensions which helped our group. As a starting point, our group used logistic regression to build our first models. Logistic regression is helpful in classifying data using parameters and is relatively easy to implement as a starting point. While we knew we would need to incorporate more advanced techniques in our model, logistic regression helped show where we needed to improve in metrics such as recall and precision.

Training the Model

Next, our group utilized different machine learning tools, such as the BERT (Bidirectional Encoder Representations from Transformers) model in developing our code. The BERT model is state of the art because unlike traditional language models that read text input in a

unidirectional way (either left-to-right or right-to-left), BERT reads the entire sequence of words at once, considering the context from both directions. This allows it to capture more nuanced relationships between words and their context which is hugely important when considering medical diagnosis criteria. BERT is highly effective for capturing dependencies and relationships between words in a sentence, regardless of their distance from each other. As such, considering patient chart notes often include important patient history, details, and anecdotes over the course of hundreds to thousands of words, the BERT model fit the project needs perfectly. Lastly, BERT is a pre-trained model, meaning it has "learned" from sources such as BooksCorpus and Wikipedia to help develop its language understanding. This helps the model with tasks such as predictions.

Other models also incorporated machine learning techniques such as neural networks, which comes from its similarity to the brain which builds connections through nodes, or neurons, for recognizing patterns. We also used techniques in which models were pre-trained on the most recent ICD-10 codes. Our group utilized these different techniques to uncover which ones were most impactful and helpful in solving our problem.

Because our models were built from the ground up, we trained it on one specific ICD-10 diagnosis area, diabetes, as opposed to trying to take on all ICD-10 codes out there. We determined that if we had success in correctly identifying diabetes codes, our model could, in theory, be extrapolated to additional diagnosis categories in future iterations. There are dozens of unique ICD-10 codes from E08- E13 related to diabetes mellitus within these categories, covering a wide range of complications such as ketoacidosis, nephropathy, and retinopathy.

Describing the Data Set

Due to constraints around receiving access to actual patient data with consideration to HIPAA and Non-Disclosure Agreements, our group needed to find a database with which to train our model. To do so, we created our own. The database was composed of fictional data created with generative Artificial Intelligence models ChatGPT, Gemini, and Claude. We accomplished this by inputting prompts of varying complexity. Simple prompts were similar to the following:

"Generate detailed patient notes with for a patient with two diseases, diabetes and heart disease and the ICD-10 codes for both conditions using a table format with columns patient notes, diagnostic code1, diagnostic code 2, code name 1, and code name 2 for 5 patients with thorough patient notes, at least 100 words each, including non-vital info such as social factors."

And others were more complex, such as:

“Generate detailed patient notes for a fictional dataset to train and test an NLP model designed to detect medical coding errors at Virginia Garcia Memorial Health Center (80% are Hispanic and many are seasonal or migrant workers. Fifty-one percent are insured via the Oregon Health Plan and Medicare). Each patient should have diabetes and comorbidities such as, but not limited to, heart failure, atherosclerotic heart disease, or coronary artery disease. Include vitals, details about their social circumstances, lifestyle factors such as but not limited to diet, exercise, mode of transportation, or profession, challenges related to medication adherence, and healthcare access. Provide at least 100 words for each patient note. Use a table format with columns for patient notes, diagnostic codes, and code descriptions. Ensure the generated data closely resembles real patient data while maintaining privacy and anonymity. Vary writing style as if to represent the perspectives of different physicians on staff (sentence length, structure, word choice, etc.).”

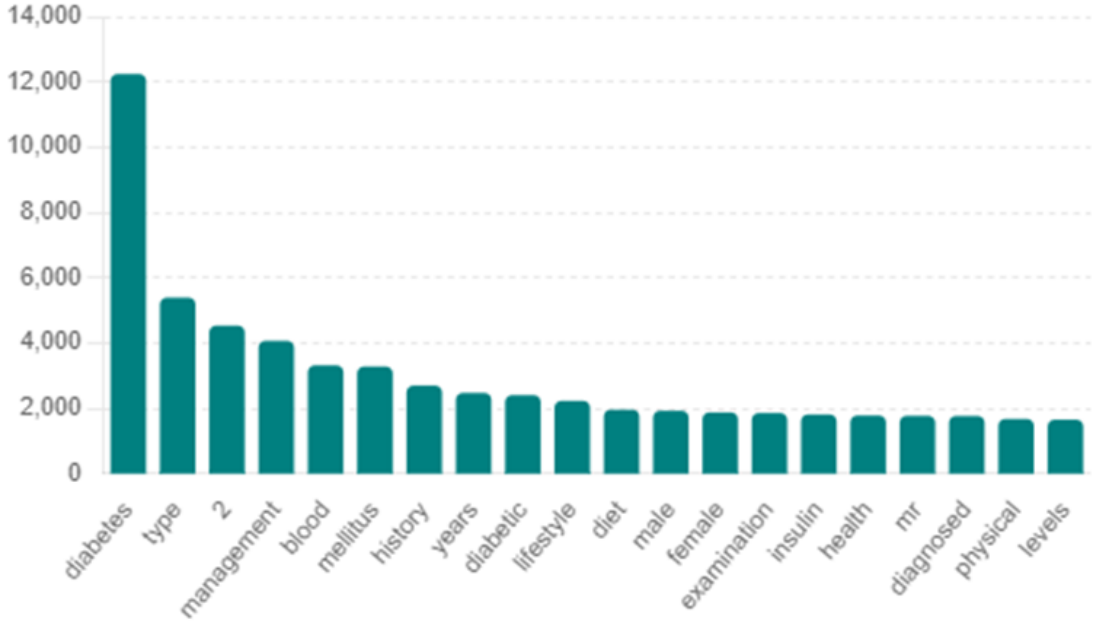
Challenges persisted, however, because consumer versions of generative AI models today are not capable of mass-producing hundreds to thousands of detailed, high fidelity, unique data points. Instead, we prompted the AI to generate roughly three to ten patient chart notes per prompt. In our conversations with Kavitha, she shared that a VG physician’s patient chart notes were roughly 1000 words. Due to technology and time constraints, we were not able to create a database where each fictional patient had 1000 words. Instead, our patient chart notes were roughly 100-500 words depending on the prompt. We shared some examples with Kavitha and were greenlit to continue.

Our first version of the database was 1000 patient entries, still much smaller than required for developing an effective NLP. We continued to add to the database eventually reaching 4,500 entries which we were satisfied with.

Data Preprocessing

First, we created a python script for removing stop words using NLTK tool kit library. The autotokenizer function from the transformer library does not perform this step.
In general, a stop word is a commonly used word (such as "the," "is," "in," "and") that is often filtered out in natural language processing tasks because it carries little meaningful information about the content of the text. Stop words are removed to reduce noise and improve the efficiency and accuracy of text analysis by focusing on the more significant words that contribute to the overall meaning.

Figure Top 20 most frequently occurring words (after preprocessing)



The script preprocesses textual data to prepare it for use in an NLP transformers model. The key steps include installing necessary libraries, importing tools for data manipulation and text processing, downloading stop words, and reading the data into a DataFrame. The script then cleans the text by converting it to lowercase, removing non-alphanumeric characters, and filtering out common stop words. The cleaned text is then saved to a new CSV file.

These preprocessing steps are essential for our transformers model because they ensure the text data is clean and free from unnecessary noise, which improves the model's ability to learn and make accurate predictions. By removing irrelevant words and characters, we focus the model on the most meaningful parts of the text, enhancing its performance and reliability. There were additional words removed from the data set after removing the stop words. Words like “patient profile”, “social history”, “occupation”, these words seem irrelevant to correctly predicting medical billing codes, so they were removed, with the find and replace tool in excel.

Model Training Overview

- Data Preparation: Load the dataset and convert the ICD-10 codes into numerical labels.

- Tokenization: Convert the patient notes into a format the model can understand. This involves breaking the text into smaller parts (tokens), making all sequences the same length, and converting tokens to numerical IDs.
- Create Datasets: Organize the tokenized data into a format suitable for the model to read during training.
- Initialize the Model: Load a pre-trained model or a specific model setup, and configure it to classify the number of labels in our dataset.
- Set Training Parameters: Define how the training will run, including the number of training cycles (epochs), batch size, logging steps, and evaluation criteria.
- Train the Model: Use the Trainer class to handle the actual training process. This class takes care of optimizing the model's parameters based on the training data and evaluating it on test data.
- Evaluate and Save: After training, evaluate the model's performance on the test data and save the best version of the model.

Importance of Training

Training is important because it allows the model to learn patterns and relationships from the data, improving its ability to make accurate predictions on new, unseen data. This process adjusts the model's internal settings to minimize errors and enhance performance.

Tokenization

Tokenization is the process of breaking down the text into smaller, manageable pieces (tokens) that the model can process. It includes Lowercasing: Converting text to lowercase for uniformity. Truncation and Padding: Making all text sequences the same length. Converting to IDs: Changing tokens into numerical IDs that the model understands.

Importance of Tokenization

Tokenization is essential because it prepares the raw text for the model, ensuring consistency and reducing complexity. This step helps the model focus on the meaningful parts of the text, making it more effective in learning and making predictions. Without tokenization, the text would be too complex for the model to process, leading to poor results.

Evaluation metrics

The performance of our medical billing code prediction model, built using Hugging Face's transformers, was evaluated on a test set comprising 60% training and 40% testing. We assessed the model's accuracy, precision, recall, and F1 score to ensure a comprehensive evaluation.

Accuracy

The accuracy of the model measures the overall correctness of the predictions. It is defined as the ratio of correctly predicted billing codes to the total number of predictions made.

Precision

Precision quantifies the number of true positive predictions among all positive predictions made by the model. It provides insight into the model's ability to avoid false positives.

Recall

Recall, or sensitivity, measures the model's ability to identify all relevant instances in the dataset. It is the ratio of true positive predictions to the sum of true positives and false negatives.

F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It is particularly useful when dealing with imbalanced datasets.

MSR BioBERT model

One model that was used was a single shot classification NLP using the MSR BiomedBERT pretrained model. This library was developed using abstracts as well as full-text articles from PubMed, a part of the National Library of Medicine, that houses biomedical literature, life science journals, and online books. Other pretrained models typically use general domain bodies of work such as the web.

Several iterations of the model were trained using datasets of different sizes and varying parameters such as the number of epochs used for training.

The first iteration of the MSR BioBERT model was trained using our initial dataset of 1,700 patient notes. The model was trained with no changes made to the patients notes. Accuracy measures were as follows:

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	No log	3.508142	0.244898	0.121553	0.764443	0.244898
2	No log	2.579801	0.437318	0.314145	0.717680	0.437318
3	No log	1.563784	0.667638	0.595512	0.776771	0.667638

The next iteration of the model was trained using our expanded dataset of 4,500 instances. There was a marginal increase in accuracy of approximately two percent.

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	No log	3.869672	0.384444	0.271615	0.788827	0.384444
2	No log	2.177769	0.613333	0.534481	0.794752	0.613333
3	3.917900	1.676806	0.696667	0.636723	0.853219	0.696667

A final attempt at refining the model using the 4,500-instance dataset with an additional training epoch. The gains in accuracy decreased from the second and third epoch, but we wanted to see if any additional gains could be made and whether they continued to decrease.

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	No log	3.869672	0.384444	0.271615	0.788827	0.384444
2	No log	2.177769	0.613333	0.534481	0.794752	0.613333
3	3.917900	1.676806	0.696667	0.636723	0.853219	0.696667
4	3.917900	1.509056	0.736667	0.696562	0.867879	0.736667

The gains in accuracy between epochs did decrease, but it is unclear if the model had reached a plateau. Running the model with additional epochs would have been beneficial, but the computational and time requirement was becoming substantial.

AkshatSurolia Model

The following summary outlines the results of our initial training session with the AkshatSurolia model using 1700 records of data. The model was pretrained on all ICD-10 codes. It appears it

utilized the most recent version of codes, ensuring its relevancy and accuracy. We then preprocessed the data, eliminating repetitive and irrelevant words, focusing only on essential information such as the patient's, age, vitals, and medical history, and results of medical examinations (if available) Additionally, one limitation was the presence of ICD-10 codes imbedded within the text, this could be a potential limitation.

After cleaning and preparing the data, we proceeded to train the model, achieving an accuracy of approximately 75%, with precision at 70%, recall at 75%, and an F1 score of 0.7. The training process spanned three epochs and took about three hours, which was a solid start. Subsequently, we implemented a prediction function to test text inputs in real time, outputting the top five predicted ICD-10 codes.

Initial Findings

There appears to be some bias in the model, which we've identified as stemming from issues with data quality. For instance, we selected the code 024419 to analyze. Our dataset was generated by four different people using four different generative AI platforms, leading to non-uniform data. Sometimes the dataset includes longer patient texts, while other times it consists of shorter texts.

The model is trained in various text lengths within the same class. It appears it struggles with longer texts because it has been trained on them less frequently compared to shorter texts. This discrepancy could lead to inaccurate predictions, particularly evident with the code O24419. In this case, there's one long sample of about 500 words, while the other nineteen records contain just a couple of sentences each. This imbalance in training data means the model is less likely to accurately predict the longer text, as it hasn't encountered such lengthy entries as frequently. The bias observed in the model is primarily due to these inconsistencies in the data. If the training data were more uniform, these issues may be mitigated because the model would recognize and process the varied text lengths more effectively.

Additional Testing

Due to the limitations of our initial data set, we decided to expand it from 1,700 records to 4,500. Each record varies in length since they were individually generated, with some patient chart notes being more detailed than others.

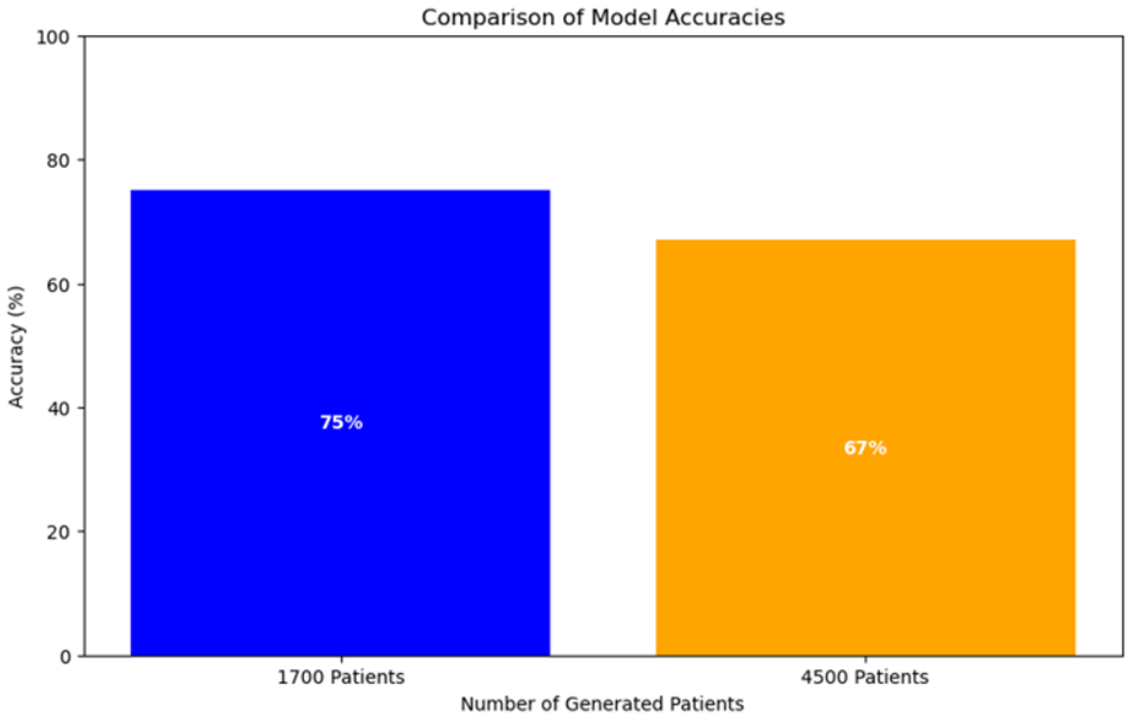
The next step was to standardize this data to ensure uniformity. We continued to focus on retaining text and features relevant to predicting diabetes-related codes, such as blood

pressure, other vital signs, and elements of the patient's medical history, including follow-up notes.

After preprocessing the data, we encountered some challenges while retraining our model. Initially, the computer we were training on crashed when we attempted to train with three epochs, indicating that the load was too heavy. To address this, we reduced the epochs and training batch arguments, planning for more training sessions to enhance model performance without overwhelming the system.

After processing the data, we trained the model again for consistency, using three total epochs. The results for the new model, trained with data from 4500 patients, were as follows: Accuracy: 67%, F1 Score: 0.61, Precision: 0.60, and Recall: 0.67.

These results indicate a decrease in accuracy compared to the previous model. Further investigation is necessary to determine the cause of this decrease. Potential factors include data quality issues or the distribution of the data. Some patient records are more detailed, while others are less comprehensive, which may contribute to the model's performance decline. Additionally, our data appears to be imbalanced, which could also be affecting the training outcomes. Overall, these results were not as expected, and additional analysis is required to understand and address the underlying issues.



```
Accuracy: 0.7551020408163265
Precision: 0.7016867642629674
Recall: 0.7551020408163265
F1 Score: 0.71375108458395
```

```
Accuracy: 0.6755555555555556
Precision: 0.5895105731788973
Recall: 0.6755555555555556
F1 Score: 0.6144120934728604
```

Neural Network Model

The neural network is another model used for predicting ICD-10 codes based on patient notes. The term "neural network" is derived from its inspiration from the human brain's structure and functioning. Just as the brain consists of interconnected neurons, the neural network consists of nodes (neurons) organized in layers. These nodes are connected via weighted edges, which are adjusted during the learning process to minimize prediction errors.

How the Neural Network Works

- Input Layer: Accepts the vectorized patient notes.
- Hidden Layers: Processes the inputs through layers of neurons applying activation functions, learning complex patterns and relationships.
- Output Layer: Produces a probability distribution over the possible ICD-10 codes, predicting the most likely code based on the input notes.

During training, the network adjusts its weights to minimize the difference between its predictions and the actual ICD-10 codes, similar to how the brain adjusts synaptic strengths based on feedback.

Model Summary and Results

The neural network model is designed to predict ICD-10 codes based on patient notes. The architecture consists of:

- Input Layer: Dense layer with 512 neurons, ReLU activation, and a dropout rate of 0.5 to prevent overfitting.
- Hidden Layer: Dense layer with 256 neurons, ReLU activation, and a dropout rate of 0.5.
- Output Layer: Dense layer with neurons equal to the number of unique ICD-10 codes, using softmax activation to output a probability distribution.

```python
# Building a neural network
model = Sequential()
model.add(Dense(512, input_shape=(X_train_tfidf.shape[1],), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(np.unique(y)), activation='softmax'))
```

Model Performance

Several model iterations were trained using different datasets and parameters, yielding the following results: using different dataset sizes and varying parameters, such as the number of epochs.

```
Epoch 1/4
4046/4046 ━━━━━━━━━━━━━━━━━━━━ 268s 65ms/step - accuracy: 0.6811 - loss: 1.6520 - val_accuracy: 0.7122 - val_loss: 2.6884
Epoch 2/4
4046/4046 ━━━━━━━━━━━━━━━━━━━━ 281s 69ms/step - accuracy: 0.9858 - loss: 0.0471 - val_accuracy: 0.7111 - val_loss: 3.3439
Epoch 3/4
4046/4046 ━━━━━━━━━━━━━━━━━━━━ 279s 69ms/step - accuracy: 0.9929 - loss: 0.0245 - val_accuracy: 0.7267 - val_loss: 3.7916
Epoch 4/4
4046/4046 ━━━━━━━━━━━━━━━━━━━━ 292s 72ms/step - accuracy: 0.9940 - loss: 0.0191 - val_accuracy: 0.7267 - val_loss: 4.1775
29/29 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7108 - loss: 4.7287
Neural Network Model Accuracy: 72.67%
29/29 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Precision: 73.37%
Recall: 72.67%
F1-Score: 72.29%
```

Model Performance Analysis

Training and Validation Performance: The model significantly improved training accuracy and learned well from the training data. However, the validation accuracy showed slower improvement, suggesting potential overfitting, especially after the second epoch, as indicated by the increasing validation loss.

Class Imbalance: Despite efforts to balance the classes, certain ICD-10 codes with fewer samples exhibited low recall and precision. This highlights the challenge of accurately predicting less frequent classes.

Suggestions for Improvement

- Increase Training Epochs with Early Stopping: Increasing the number of epochs while implementing early stopping could help capture more details without overfitting.

- Class Weight Adjustment: Adjusting class weights during training to give more importance to underrepresented classes can improve recall for these classes.
- Data Augmentation: Augmenting data for less frequent classes can help the model learn better representations for these classes.
- Hyperparameter Tuning: Tuning hyperparameters such as learning rate, batch size, and dropout rates can help find the optimal configuration for the model.
- Advanced Oversampling Techniques: Employing techniques like SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic) could better handle class imbalance.

The neural network model shows promising results with a final accuracy of 72.67% and balanced precision, recall, and F1 scores. While the model performs well in frequent classes, further improvements are necessary to enhance performance in less frequent classes. Implementing the suggested improvements could lead to a more robust and generalizable model.

Random Forest with BERT Model

The Random Forest with BERT Model was trained on roughly 4500 data points with a training: testing ratio of roughly 5:1. This model incorporated BERT which is a pre-trained deep learning model that reads data both left to right and right to left. Random Forest is a learning method that uses decision trees, bagging, and random feature selection. Further, this model incorporated hyper parameter tuning which optimizes the parameters of a machine learning model to improve its performance.

R73.0	0.50	1.00	0.67	2
R73.02	0.90	1.00	0.95	9
R73.03	0.00	0.00	0.00	3
Z34.81	0.50	1.00	0.67	3
Z79.3	1.00	0.67	0.80	3
Z79.89	1.00	0.50	0.67	2
Z83.41	0.00	0.00	0.00	3
accuracy			0.67	864
macro avg	0.33	0.33	0.32	864
weighted avg	0.65	0.67	0.65	864

Results from the first iteration of the model showed satisfactory progress but with much room for improvement. F-1 score of .65 and accuracy of .67 was adequate for a first iteration.

[675/675 3:35:23, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	2.093000	2.079631	0.613333	0.547323	0.523049	0.613333
2	1.794900	1.902897	0.655556	0.600582	0.578238	0.655556
3	1.489300	1.695705	0.690000	0.641897	0.623220	0.690000

The next iteration incorporated Scipy Randint which returns a random integer value between the two lower and higher limits. This was useful because in our data set, some ICD-10 codes only appeared once or twice which made training the model challenging as there was less than preferred amounts of data to train off. Incorporating this change decreased the F-1 score slightly but increased other metrics such as accuracy and recall by the third epoch.

Results

In this project, we fine-tuned several large language models to predict medical billing codes, as requested by Virginia Garcia. Despite challenges, particularly with data access, we overcame these by generating our own data. This allowed us to train and evaluate several models: the MSR BioBERT, AkshatSurolia, Neural Network, and Random Forest with BERT uncased.

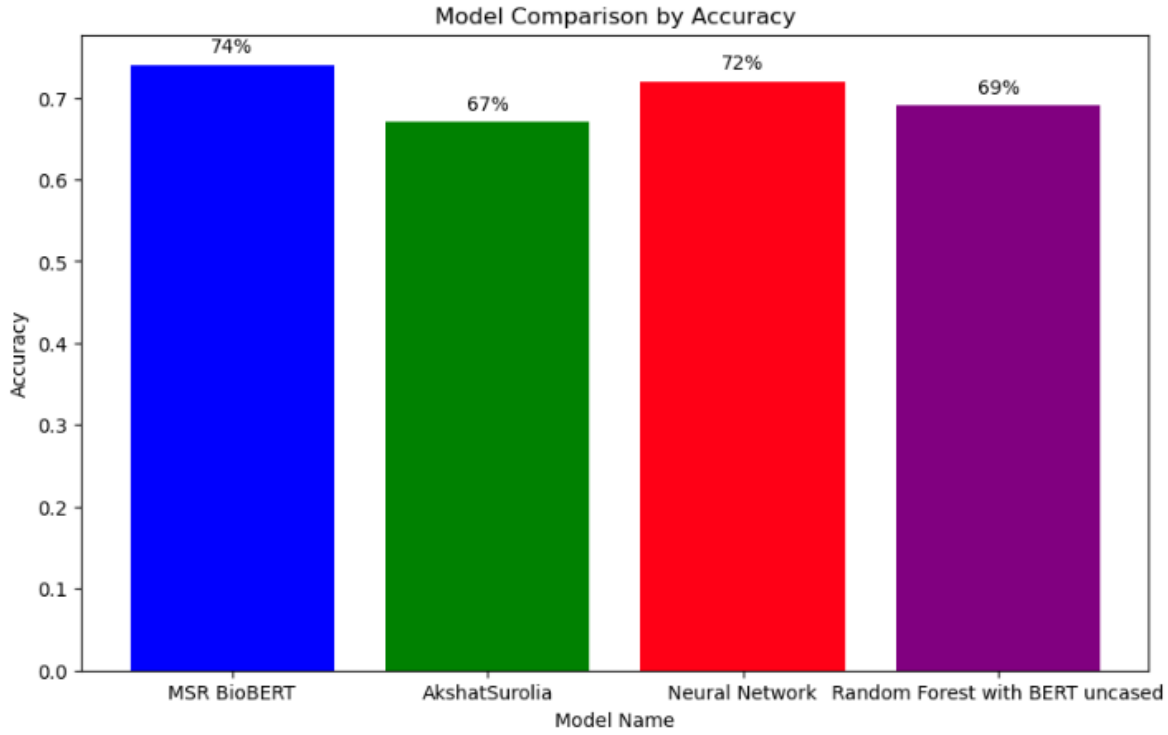
Model Evaluation

MSR BioBERT: This model achieved the highest accuracy of 0.74 and a precision of 0.87, indicating it is very precise in its predictions. However, the F1 score of 0.69 and recall of 0.74 suggest there is room for improvement in terms of balancing precision and recall.

AkshatSurolia: This model had the lowest performance among the evaluated models, with an accuracy of 0.67 and an F1 score of 0.61. Its precision and recall values were 0.58 and 0.67 respectively, indicating challenges in predicting the correct codes.

Neural Network: This model demonstrated balanced performance with an accuracy of 0.72, F1 score of 0.72, precision of 0.73, and recall of 0.72. It shows consistency across all metrics, making it a reliable model.

Random Forest with BERT uncased: This model achieved an accuracy of 0.69, with an F1 score of 0.64, precision of 0.62, and recall of 0.69. While not the highest performing, it provides a good balance between precision and recall.



The overall accuracy of these models ranged from 0.67 to 0.74, which did not meet our initial expectations. We aimed for high accuracy, hoping to land in the 0.90 range. There is uncertainty about the models' performance with new data arises from the fact that the training data was randomly generated by AI. Nonetheless, this project has laid a foundation for fine-tuning models in a medical setting. We successfully generated medical data, fine-tuned models to predict medical billing codes, and implemented Python code to evaluate the top five predictions of an input string of text, testing the correctness of our trained models.
While the current models may not be ideal, this project provides a significant step towards improving the prediction of medical billing codes.

Future Work

To develop a successful transformer model for predicting medical billing codes, future efforts should focus on ensuring access to high-quality, real-world data from the clinic that will use the model. This will ensure the model is trained to predict new codes for this clinic.

Initially generating data might be necessary, but obtaining comprehensive patient chart notes from the medical clinic will significantly enhance the fine-tuning process. The data will be more

balanced and will better reflect the billing codes that the clinic will likely need to predict in the future.
Features should be carefully selected to capture the unique characteristics of each billing code while remaining general enough to apply to various medical scenarios. Balancing the training data is crucial to avoid overfitting and to improve model performance on new data.

Regular evaluation using metrics like accuracy, precision, recall, and F1 score is essential, aiming for a higher accuracy rate that meets clinical and operational requirements. An iterative approach to fine-tuning, with validation against a separate test dataset, will help identify optimal parameters and features. Utilizing a flexible framework, such as a Python script as a template, allows easy adjustments to parameters, code, and data for continuous improvement.

Overall, future research should prioritize obtaining more comprehensive datasets and refining feature selection. Addressing these areas will support creating a model that not only meets but exceeds accuracy requirements for medical billing code prediction.

Recommendations:

Organizational change

In addition to building a model that is accurate, another important consideration in implementing an NLP technology will be integrating the technology into current systems, workflows, and policies. Having solid change management practices will be crucial to ensuring the new technology has widespread adoption and acceptance. The onus of successful adoption will be on the leadership team and how they introduce and communicate the change to all internal stakeholders.

Create goals and timelines
Before beginning development, the leadership should define goals and create a clear vision of the product that you want to create and how it ties with VG mission. Defining a clear goal will enable leadership to communicate the purpose to stakeholders. Careful attention should be paid to outlining the benefits of adoption both in terms of work flows as well as the mission and vision of VG.

A timeline of the project should be developed with stakeholders involved. Ensuring that internal stakeholders are involved in the development of the timeline will ensure that it is realistic and give them a sense of agency and ownership of the project. Benchmarks for short-term goals as well as long-term goals should be defined.

Creating buy in

Securing buy-in for a new technology is all about building a bridge between its introduction and widespread adoption. Here's a roadmap to navigate that journey with six key steps:

First, cultivate awareness. Clearly communicate the technology's purpose and its potential impact. Town hall meetings, explainer videos, and targeted email campaigns are all effective ways to get everyone on the same page.

Next, address concerns. Anticipate anxieties about how the new tech might disrupt workflows or require additional work. Be transparent about challenges and proactively demonstrate how technology solves problems or improves efficiency.

Now, spark mental engagement. Encourage users to envision themselves using the technology. Walk them through real-world scenarios where it streamlines tasks or fosters collaboration. This mental tryout fosters a sense of ownership and builds excitement for the hands-on experience.

Following this, provide a safe space for hands-on trials. This could involve pilot programs or sandbox environments where users can experiment with the technology at their own pace. This fosters comfort and allows them to discover the technology's benefits firsthand.

As users gain experience, gather their feedback. This allows for course correction and demonstrates a commitment to user needs. By incorporating valuable insights, you can refine the technology to drive acceptance.

Finally, identify champions of the change. Empower enthusiastic users to become vocal advocates. These champions can provide peer-to-peer support, answer questions, and alleviate anxieties among their colleagues. Their endorsement holds significant weight and paves the way for broader adoption. By following these steps, you can create a smooth path for new technology integration and empower your team to leverage its full potential.

Analytics

Implementing an NLP model for medical coding requires a dedicated team and an iterative process. First, hiring an NLP consultant or full-time data scientist with experience in healthcare may be beneficial.

The development should follow an iterative approach. This means starting by building a model to predict a single code based on a specific portion of the medical record. As the model's accuracy improves, it can be refined and expanded to encompass a wider range of codes and expanded to predict multiple codes.

Integration with existing software is also an important consideration. An API (Application Programming Interface) can be designed to bridge the gap between the new NLP system and current coding software. This ensures a smooth workflow and minimizes disruption for medical coders.

Voice-to-text technology presents an interesting opportunity for future development. Integrating voice recognition into the system could allow for real-time coding as physicians dictate chart notes. However, this would require additional development and validation to ensure accuracy and compliance with healthcare regulations.

Business

In implementing a new technology or process, there will be business considerations. Integration of an NLP will require a higher degree of communication between departments. Currently it seems that the organizational structure at VG is siloed. There is an opportunity to develop interdepartmental relationships, especially if VG decides to pursue integration of NLP within the medical coding process.

Feedback mechanisms will be crucial for improving whatever NLP interface that is used. A method for communicating problems or potential improvements for the products should be available outside of the current IT ticket process in place.

Additionally, workflow considerations should be carefully analyzed before and during the adoption of new technology. A system for assigning identified miscoding's to the medical coders will need to be established. If there is no way to integrate identified errors into the current work queue used by the coders, VG will need to find a way to ensure that there is a way to track to status of corrections that are outstanding.

Depending on the number of errors that are identified, the increase in the workload may necessitate increasing the number of medical coders, especially given the fact that the department already appears to be understaffed based on feedback received during our interviews.

The technical ability of the staff will also need to be assessed to determine who will implement the NLP technology, as well as how end users will interact with the data. Feedback from end users should be obtained during development to ensure that the interface is easy to use and easy to integrate into current systems.

Lastly, beyond providing VG with a means to track missed ICD-10 codes, there are other ways the technology can be used to increase operational efficiencies. If codes can be connected to specific billing amounts, VG can track the amount of revenue the institution would have missed without the new technology.

Conclusion

The Virginia Garcia NLP Project represents a significant step towards enhancing the operational efficiency and accuracy of medical coding at the Virginia Garcia Memorial Health Center. Our comprehensive approach identified critical challenges in the medical coding process, such as reliance on manual chart reviews and the lack of robust error-tracking mechanisms. Our team, comprising members with diverse backgrounds and guided by expert interviews, developed various models designed to predict ICD-10 codes based on patient notes.

Despite the constraints of generating our training data and the complexities of model training, we achieved promising results. Our models, including iterations of the MSR BioBERT, AkshatSurolia, and Neural Network, demonstrated potential in automating and improving medical coding accuracy, albeit with accuracy rates that call for further refinement. The insights gained from our interviews and data analysis highlighted the importance of continuous improvement, robust data preprocessing, and addressing class imbalances to enhance model performance.

Implementing this NLP model underscores the critical role of integrating advanced technology into healthcare settings to reduce administrative burdens, minimize coding errors, and improve patient care. Our recommendations for future work emphasize the need for high-quality, real-world data, iterative model training, and close collaboration between departments to ensure the successful adoption and integration of the NLP system.

Looking forward, the path to a fully optimized medical coding process lies in refining our models with real patient data, improving feature selection, and addressing the operational and organizational changes necessary for seamless technology integration. By doing so, Virginia Garcia Memorial Health Center can achieve its vision of providing high-quality, comprehensive

healthcare to underserved communities, ensuring operational efficiencies translate into better patient outcomes.