**Snobird V5**

**Introduction**

Snobird is an application designed to connect long distance family members with patients who have been diagnosed with a chronic disease and their primary care physicians. Studies have shown that emotional support from family members is the single most important aspect to a patient’s journey to recovery and treatment. Social support from the patient’s family can mitigate the impact of chronic disease diagnosis on the patient’s quality of life. Higher family support and involvement is associated with lower depressive symptoms and increased emotional well being in a patient’s life.

**Problem Statement**

The problem arises when the patient is geographically separated from their loved ones and family members who would offer the emotional support that the patient is need of during the hard times of diagnosis, treatment and recovery. Typically, these patients include elderly couples who have migrated away from their families under varying circumstances, such as wanting to enjoy warmer climate during retirement.

The distance can make it difficult for the family to stay connected and informed of the status of the patient’s health as the patient goes through treatment. This is extremely prevalent when long distance caregivers need to take the elderly patient to physician. Encounters between a physician and a patient typically last seventeen minutes and during this interaction the caregiver only has 5 minutes to communicate the patient’s health condition and symptoms to the physician. Therefore, it is vital that the caregiver is well informed and on the patient’s health and relevant symptoms and is able to present this information to the physician in an efficient and concise manner. The caregiver has the key responsibility of speaking on behalf of the patient and best advocating for the patient. Physicians are trained and educated to recognize early symptoms and signs of evolving illnesses, therefore with accurate awareness of symptoms physicians can provide timely intervention before the illness becomes full blown or difficult to treat.

Many families resort to using common communication channels, such as Whatsapp or iMessage group chats to keep the family connected and keep them informed on the health status of the patient. However, this method of communication can be challenging for a lot of members of the family to keep up with relevant patient information through all the incoming group messages and the general chatter of the groupchat. This also makes it difficult for the family to present an accurate timeline of symptoms to the primary care physician when a health care event escalates.

Ontoadaptive Fall 2019 | 1

**Solution** Snobird v5 is an innovative digital health platform that empowers long distance family members to stay connected with patients in a timely, consistent and relevant manner. Snobird interwines the human aspect of empathy and the applications of machine learning technology to bridge the gap between long distance caregivers and patients. Snobird v5 empowers family through communication. It analyzes text messages in the family group chat through a natural language processing mechanism to notify the family when a healthcare concern is raised regarding the patient’s health.

To better exemplify how Snobird works refer to the diagram below that shows a communication thread from the group chat of a family. The family is having their daily conversation and when the patient is at baseline, which is anything 0-2, this means that the patient is healthy and at their normal. However, as soon as a health issue is raised the caregiver informs or alerts everyone in the chat about it and in this particular case, it is that their father is peeing more frequently than normal and has cloudy urine. Snobird will “flag” this health concern and will notify those family members nearby that they need to be available in the case that their help is needed and for those who live long distance to check in with the patient and be more active in checking the group chat. This allows the primary caregiver to feel and know that they are not alone in taking care of the patient rather they are working with a team. Family members join the chat with suggestions on next steps like informing the doctor, sending urine for urinalysis, making him drink lots of fluids and monitoring his temperature. Also, getting the doctor involved early can alert the doctor to be aware of the fact that the disease may be evolving so that if a patient develops a fever, the family is instructed to not wait nor hesitate to take the patient to the Emergency Room for immediate attention such as IV antibiotics. In this case of the 94 year old patient, being aggressive in treating the infection could be the difference between a typical UTI being treated or the patient developing sepsis which can eventually lead to fatal outcomes like death.

Key health events are represented by the “flag” between the time that the health concern is raised and when the patient is taken to the doctor or ER. The key events are classified, tagged and pulled to create a time-stamped summary of key events or timeline of the present illness. This allows all family members, near and far, to have a better understanding of and access to information on what the patient is going through and therefore will be able to efficiently share the story with the medical team.

Ontoadaptive Fall 2019 | 2

Figure 1 visual representation of dashboard -mockup -

**Value Proposition**

As care becomes more centered on the patient, the care team need to have concise view of the patient journey over a period of time.

Snobird uses a machine learning technology approach to medicine as a way to not only keep long distance family members informed on the patient's heath, but also aids in helping the caregivers in communicating symptoms to a physician in an accurate concise sequential order by analyzing text message data from a family group chat.

**Text Classification**

Data was collected in the form of WhatsApp text messages from the family to generate insights from the patient’s daily conditions. Before inputting to the model, the text messages were classified into 6 classes based on their relevance to patient health. The baseline (or normal behavior) is considered to be 2 and anything above 2 is abnormal and above baseline. The following table shows the 6 different classes with their descriptions:

Class Description

0 Unrelated to the health of the patient

Ontoadaptive Fall 2019 | 3

1 Related to the health of the patient; General conversation about how the

patient is feeling

2 Relevant to the health of the patient; Less adverse health event

3 Relevant to health of the patient; Health issue is raised

4 Pertaining to the patient going the ER/Hospital

5 Events that are happening at the ER/Hospital

\*Normal patient health status is baseline which is 2 and below. (0, 1, 2)

0 - Conversation that is irrelevant to the patient’s health (We watched a movie today.)

1 - General conversation regarding patients health (How are you?)

2 - Not concerning information about the patient’s health (Papa feels better today.)

3 - Health concern is raised and patient is displaying abnormal health behavior, for example, patient is complaining about a headache

4 - Escalation to discussions of going to the hospital or making a doctor’s appointment

5 - Events that occur at the hospital or emergency room

**Comparative Analysis**

**Method:**

Deep neural networks (DNNs) have revolutionized the field of natural language processing (NLP). This project revolves around the concept of NLP so the aim was to experiment working with Deep Neural networks. The most widely used DNN architectures to handle NLP tasks are the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). On the given data, this is the first comparison between the CNN and RNN models aiming to highlight the performance of these models in classifying text messages.

One of the key findings from previous experimentation on this topic is that model architecture

Ontoadaptive Fall 2019 | 4

leads to better performance in some NLP tasks. This is attributed to how important it is to semantically understand the whole sequence. For instance, a task like sentiment analysis where the sequence of the sentence is not important can use the CNN model. Whereas, tasks in which translation of the sequence of the sentence is crucial can be best handled by the RNN model. [W. Yin, K. Kann,2017]

The table below discusses the important differences between these models:

CNN RNN

CNN is a class of neural networks which are deep and fee-forwarded. Here, connections between nodes do not form a cycle.

Although they are known to be used in computer vision, CNNs have also shown promising results when they’ve been applied to different NLP tasks.

RNN is a class of artificial neural network where connections between nodes form a directed graph along a sequence.

The model architecture of RNN allows it to capture sequential data. Thus, while dealing with texture data, this is a more ‘natural approach’ since the text is naturally sequential.

CNN learns to recognize patterns across space. An RNN is trained to recognize patterns

across time.

CNN performs classification tasks such as Sentiment Analysis, Spam Detection and Topic Categorization.

RNNs perform efficiently for applications where sequential information is very important because if sequential information is not used, the meaning could be misinterpreted or the grammar could be incorrect.

CNNs can learn to classify a sentence or a paragraph.

Recurrent neural networks use time-series information .

Long Short-Term Memory (LSTM) is a special kind of RNN. It helps in easier remembrance of past data in the memory.

Ontoadaptive Fall 2019 | 5

These differences supported the approach taken to experiment with both models and compare how they perform in accurately classifying the message.

**Data:**

One of the issues with the original dataset was the size of the dataset. The dataset was too small and was dominated by irrelevant data which had a negative effect on training the model. Another issue with the dataset was the imbalanced. That means the dataset have a majority and minority classes. The majority class consisted of the data that was classified as a zero which was also the irrelevant class of data. The minority classes of the dataset were classified from one to five which were the relevant classes of data. A small set of data with the dominance of irrelevant data has an effect on training the model.

The table below represents the process of increasing the size of the dataset which was received at the beginning of the project. The data evolved by getting supplement data which has been classified to help in training the model. Also, other natural language processing methods like EDA and SMOTE were applied to handle the size and imbalance in dataset.

***Description Size***

Original size of data ~~2737

After supplement data- classified by team ~~4300

After applying easy data augmentation ~~7837

After applying Smote ~~8277

**NLP Technique**

**(i). Easy Data Augmentation**

It is known that a machine learning model learns features from dataset. However, the dataset we

had was relatively small which could have a negative impact in terms of overfitting. To avoid

Ontoadaptive Fall 2019 | 6

overfitting, more data could be collected or the current dataset could be augmented to artificially

add more data. In our case, we received additional datasets but that wasn’t enough. Then **we**

**experimented applying one NLP** technique **to handle this issue which is EDA [Wei and Zou,**

**2019]**. Easy Data Augmentation (EDA) is a technique used to artificially expand the size of a

training dataset by creating a modified version of the data. It consists of four powerful

operations:

1. Synonym Replacement 2. Random Insertion 3. Random Swap 4. Random Deletion

The following table shows how each of these operations work on an example sentence from our dataset:

***Operation Sentence***

None He might have problems with his prostate, could lead to UTI.

Synonym Replacement He **may** have **complications** with his prostate, could lead to UTI.

Random Insertion He might have problems with his prostate, could **possibly** lead to

UTI.

Random Swap He might have problems with his prostate, **to** lead **could** UTI.

Random Deletion He might have problems with prostate, could lead to UTI.

Below are the graphs that show the impact of Easy Data Augmentation (EDA) on our model:

Ontoadaptive Fall 2019 | 7

**Figure 3. Distribution of data among class after applying EDA**

**(ii). Synthetic Minority Oversampling Technique (SMOTE)**

SMOTE is a statistical technique which can be used to increase the number of cases in a dataset in a balanced manner. This is useful because when one class dominates the other in the dataset, the model finds it hard to learn. Our dataset was dominated by texts classified as 0 compared to the number of texts classified as 1, 2, 3, 4 and 5. We experimented with applying this technique as a way to resolve the class imbalance. it works by generating new instances from the minority cases present in the dataset without changing the number of majority cases. This technique was applied after splitting the data into testing and training sets. Oversampling before splitting the data can allow the exact same observations to be present in both the testing and training sets. This could lead the model to memorize specific data points that cause overfitting and poor generalization of the testing dataset. Having new samples in the training set ensures that the

Ontoadaptive Fall 2019 | 8

**Figure 2 . Distribution of data among class before applying EDA**

model generalizes to unseen data. Applying this to the dataset led to an increase in dataset to approximately ~8000.

**Model**

**Hyper-Parameters**

Hyperparameters are variables that needed to be set prior to applying any learning algorithm to a dataset. Since the best value for a model hyperparameter depend on the task and dataset. To achieve the best performance, we experimented changing the ***epoch and batch size*** while keeping dropout same.

**Dropout**: The term “dropout” refers to dropping out units (hidden and visible) in a neural network. Dropout is a regularization technique to avoid overfitting. Overfitting occurs when the model learns the detail and noise in the training data to such an extent that it negatively affects the performance of the model on the new data.

**Epoch**: One epoch refers to one forward pass and one backward pass of all the training examples.

**Batch size**: The number of training examples in one forward/backward pass. The higher the batch size, the more memory space you’ll need.

In this comparison, we set up the Hyper-Parameters for both models to be as the following: Epoch 8; Dropout 0.2; Batch size 64 . Also,The validation split has been set to 0.5

***Performance of CNN and RNN:***

CNN RNN

**Accuracy %84 %82**

**Accuracy with applying EDA and SMOTE**

**%94 %94**

Ontoadaptive Fall 2019 | 9

With the original dataset, when we test the model with new predication, we have hardly seen a message classified as anything besides the 0. That was because there was not enough data to get the model to train in predicting the other classes. After applying the NLP technique, we started to see messages classified as 3, 5, 1, 4. Although the accuracy for both models is relatively close, we found that RNNs perform well in accurately classifying unseen data.

***Conclusions***

After applying two DNN models on the dataset, we found that the RNN model performs better in terms of predicating the class. That goes back to the importance of the model architecture and how it is crucial in handling the task.

**Google Cloud Platform**

After increasing the accuracy of the model, and implementing the RNN model we worked to deploy the model to a flask app on the Google Cloud Platform. The goal of this app is as shown in the architecture diagram. The goal of our app was to have a web application that a user could enter a message and the deep learning model would classify the message and the app would then display the model’s classification prediction. The goal was that the message would then be displayed on our plotly timeline along with the UTI example that was stored in the datastore. Our application currently accepts messages and provides a prediction after running through the RNN model as well as shows our stored UTI example plotly diagram on a separate page. While this app does a majority of what we were aiming for, it does need integration of the input messages and the plotly diagram, as well as the input being from a Snobird mobile app where the messaging is taking place directly.

On the Google Cloud Platform (GCP) we used Google App Engine (GAE), Datastore, and we researched and attempted to use Firebase. The GAE ran our flask web application. We attempted to use code that we found on Github for the UI of the app in the beginning, however this proved difficult because the code we found was not built for cloud deployment.

There was a learning curve with the GCP and some issues that we were not anticipating. For example, when we attempted running Bert-as-a-service, an NLP model developed by Google, the GCP account showed errors not only on GAE but on the home pages just stating that the account had an error. Eventually, it resolved itself, but we ran into issues like this frequently enough that we started using multiple accounts to make changes and test things before moving it over to the DEV account.

Google Cloud deployment was a scope change we saw in the second half of the semester/bootcamp. Ontoadaptive Fall 2019 | 10

Given the agile work environment of the bootcamp getting the tasks with the most risk done earlier is advised, because that allows you to have enough time to mitigate those risks. GCP has a lot of documentation and tutorials on how to deploy apps which we found very helpful throughout this process. A key takeaway from this is to always search for the official documentation first before going to Stack Overflow or Medium.

**App Architecture Diagram**

**Next Steps:**

1. BERT as a service: Google created a pre-trained model that is trained on all of Wikipedia called BERT. Bert-as-service is a sentence encoding service for mapping a variable-length sentence to a fixed-length vector, e.g. hello world to [0.1, 0.3, 0.9]. Each element of the vector should "encode" some semantics of the original sentence. [hanxiao/bert-as-service: https://github.com/hanxiao/bert-as-service]

2. The dataset contains more data than just the text messages. User, date, and time data should be taken into consideration when developing Snobird v6. Example: Text messages from an important user (primary caregiver) should be flagged when the frequency is abnormal.

3. Using pre-trained embedding as a layer, example: FastText. FastText is a word

Ontoadaptive Fall 2019 | 11

embedding model where each word vector is based on sub-word characters. This means that even for a previously unseen word by the model or a typo, the model can still make an educated guess on its meaning. For this approach, the embedding needs to be handcrafted for each example since the classic Keras embedding layer will not be used. Although more time is invested into preprocessing, it is justified by better results. https://cai.tools.sap/blog/glove-and-fasttext-two-popular-word-vector-models-in-nlp/

4. Predict sequences, trends, symptom-based prediction - with patients like the ones with

cancer having recurrent episodes, sequences can predict what can be anticipated next.

5. When data is classified as a 3 for consecutive text messages then the classification will

most likely rise to a 4 or 5. There should be a model that flags the message classified as a 3 as the start of a concerning sequence.

6. Incorporating text messaging within the Snobird app instead of the user inputting it in the

UI of the flask app.

Below is the Gantt Chart for the suggested next steps:

**Reference**

Ontoadaptive Fall 2019 | 12

Wei, Jason, Zou, & Kai. (2019, August 25). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. Retrieved from https://arxiv.org/abs/1901.11196v2.

W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of cnn and rnn for natural language processing,” arXiv preprint arXiv:1702.01923, 2017.

hanxiao/bert-as-service: https://github.com/hanxiao/bert-as-service

Ontoadaptive Fall 2019 | 13