Predicting User Context Switch for Memory Based Chatbots

Rigdha Acharya[1] and Dong Si[1]
rigdha@uw.edu, dongsi@uw.edu

University of Washington, Bothell, WA, USA

Abstract. Most chat bots today use slot filling method to find information from the user to perform information extraction task. However, most chat systems require redundant communication to explain trivial user constraints [1]. In addition, if a user wants to switch between contexts such as booking a trip to Miami and then asking about a trip to Houston, today's chat system lose all other constraints about the user's request. Frame tracking task has been proposed to track the full context of user conversations [2]. In this project, we use the Maluuba Frames dataset [3] to predict whether a context switch has occurred.

Keywords: chatbot · information retrieval · Maluuba frames · memory based chat.

1 Introduction

Chatbots are popular in goal-oriented information-retrieving systems [3]. However, conversing with these systems is a sequential process with many redundant turns in which the computer tries to extract user's constraints. These bots are limited when we look at complex systems such as e-commerce where users want to compare products or gain information about multiple products. In these cases, if a user switches context such as comparing vacation in Miami vs vacation in Houston with the same budget and travel party, existing systems will require the users to input information they have already shared with every context change. The Maluuba Frames dataset and research aims to minimize the redundant information seeking by saving contexts of user requests and allowing users to switch between these contexts just like two humans can talk about the Product A and Product B, compare these two options and then go back to more questions about Product A. In this project, we extract information from user text to determine if a context switch has occurred.

1.1 Background Research

Goal oriented dialogue system research have focused on tracking the current state of a user's goal. In the paper The Dialog State Tracking Challenge [6], the authors describe state tracking as the step that deduces user's goals from the

user's text. Dialog systems rely on the user's estimated dialog state to decide which information should be presented to the user, for example: if a user asks for bus schedule for route A for 9 AM and then asks for the schedule for 10 AM, it is important for the dialog system to realize that the user is still referring to the same state constraint of Route A. Most systems today use expert designed hand-crafted rules for deciding if a state change occurred. Multiple methods such as neural networks, reinforcement learning etc. have been proposed to track state accurately. For example: In Conversation as Action Under Uncertainty [7], the authors propose multi-level Bayesian networks to track the conversation state. Additionally, in Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems the authors found that tracking dialog state significantly improves the performance of dialog systems.
While these papers have focused on calculating the current state, the authors in Frames: A Corpus for Adding Memory to Goal Oriented Dialogue Systems [3] propose that it is not sufficient to track current dialog state for complex chat systems in e-commerce context where users like to compare and explore products. They introduced a Frame Tracking task which is an extension of state tracking. It is common in e-commerce tasks for users to switch their goals but traditional state tracking would not be able to walk back to a previous goal described by the user [8], therefore, a frame tracking system needs to be able to recognize when a user changes his or her goal by switching to a new frame or when a user wants to go back to a previous goal by uttering references to a previous frame. [4] presents a corpus of 1369 dialogs between humans that illustrates the frame tracking challenge. [8] proposes a model for tracking the frames based on various user constraints such as destination, duration etc. In this paper, we use the original corpus proposed in [4] to create a model that determines if a frame switch has occurred and we also use Named Entity Recognition model to extract the constraints necessary to track dialog state and frames.

2 Data Pre-Processing and Feature Selection

2.1 Data Set and Definitions

Maluuba Frames dataset provides a corpus of 1369 dialogs between humans seeking information and booking travel packages. The dataset was collected from 12 users over 20 days where some experts, known as wizards, acted as a bot [4]. The users were given a few templates but were also asked to create their own conversations. These conversations have minimal spelling errors and avoid uncommon slangs to aid data processing. We describe the key components of the dataset below.

Turn Each conversation between the user and wizard is known as a turn and dialogs have an average of 15 turns.

Frames Frames are used to track the context of a user conversation. For example: if a user asks for a trip to Miami under $700, we can tag that as Frame A. If the user switches to a vacation in Houston, we would track that as Frame B. If a user wants to come back to the Miami vacation with additional questions, we track that again under Frame A which allows us to use all the information about the user's budget and travel party to provide the necessary information.

Constraints A user's request for Destination city, Departure city, Dates, Budget, Travel Party, Amenities make up the constraints of each frame. We track the constraints for the entire conversation and modify the constraint for each frame if the user changes locations or budget.

2.2 Data Structure

Maluuba dataset is represented in JSON with human tagged frame numbers, constraints etc. In this research, we want to predict if a frame switch has occurred. To do so, we remove all of the manully labeled entities to get directly to the text and extract the constraints during the Named Entity Extraction phase. We then return to the labeled constraints to train the classifical models.

Spell Check Although this dataset was previously processed and has very few spelling errors. We still expect some errors because these are chat logs. We used the Bing spell check API to find and correct spelling errors. Note that the code shown in the BingSpellCheck.SpellCheck class in the C# program has been copied from the sample code provided by Microsoft. [5]

Contraction Human speech and chat logs usually have contractions that make it difficult for computers to parse. For example: we found that Named Entity Extraction fails to extract Jan as a date, but it can identify January as a date much easily. Similarly, tokenization was unable to handle contractions like I'd when it handled I would fine.

Tokenization and Stop Word Removal Our dataset has a list of sentences. In order to process these sentences, we must first convert them into list of words that can be used as a semantic unit for processing the meaning of the sentences. Along with tokenization, we also applied stop word removal. Stop word removal removes words like 'to', 'from' etc. that do not add significant meaning to the sentence. As an example - here is one sentence from a sample conversation.

Book a trip to Miami for March 7 for 4 people under $700

We first tokenized the words but after applying stop word removal, we found that important words such as 'to', 'for' etc. were removed. This would remove important information such as going 'from' a city 'to' a destination city. Here is the tokenized list of words of the example sentence. 'Book', 'a', 'trip', 'to', 'Miami', 'for', 'march', '7', 'for', 4', 'people', 'under', '$700'

Part of Speech Tagging Part of Speech Tagging is used to identify the purpose of various words in a sentence. For example: when we identify Jim as a noun, like as an adjective and run as a verb, we can train the computer by creating a syntactic parser so that the computer can understand the meaning of the sentence. For our dataset, POS tagging is able to identify Cardinal numbers such as 2000, which could be a budget or date, and Nouns such as Miami. We use the output of this method to further process the sentence using Lemmatization and Named Entity Recognition. Here is the result of pos tagging of the example sentence.

Table 1. Part of Speech Tagging of Example Sentence

Token	POS
Book	VB
a	DT
Trip	NN
to	TO
Miami	NNP
for	IN
march	NN
7	CD
for	IN
4	CD
people	NNS
under	IN
$700	CD

Lemmatization Lemmatization allows us to reduce many complex and related forms of a word into a common base form. For example: this allows us to teach the algorithm that a user who wants a vacation package for 2 adults and 2 kids is the same as a user who wants vacation package for his wife, 2 children and myself. In our case, this allows us to reduce complex human conversations into simpler forms that can be used to extract information.

Named Entity Recognition Named Entity Recognition allows our algorithm to extract basic information from a user's conversation. For example: we can use NER to extract that Miami is a Geo-Location when a user says, "Book me a ticket to Miami". We use this model to extract key constraints presented by the user. Initially, we thought of extracting the entities using logistic regression based on the POS, but we realized that it is a more complex problem that requires information about sections that have already been processed. The Spacy python library provides us pre-trained models as well as the ability to train our own model. We tried both the NLTK and Spacy models and later decided to train our own model using Spacy.

3 Feature Extraction with Named Entity Recognition

The original dataset contains manually labeled constraints. In this project, we compared the default NLTK model, Spacy EN model and a custom trained Spacy model to extract the features.

3.1 Named Entity Recognition Model Results

We tested existing Named Entity Recognition Models from NLTK and Spacy using sample sentences from the dataset. In our experiments we found that the existing pretrained models were not good at extracting the features relevant for our experiment. The table below shows the strengths and weaknesses of each of the models we tried. We also provide details on our custom trained model in the following section.

The python script ner_compare.py can be used to replicate the results of the default NLTK and Spacy EN models. The file spacy_ner_test.py can be used to test the custom NER model. Note - the spacy_ner_train.py needs to first generate the custom model that will be used by the test script.

Table 2. Named Entity Recognition Model Results

Criteria	NLTK (Default)	Spacy (EN)	Spacy (Custom)
Fast	Yes	Yes	No
Extract GEO Locations	Yes	Yes	Yes
Distinguish Origin vs Destination	No	No	Yes
Extract Dates	No	Yes	Yes
Extract Numbers	Yes	Yes	Yes
Distinguish Budget vs Number of Adults	No	No	Yes

3.2 Data Processing for Training Named Entity Recognition Model

The original dataset is provided in JSON format and contains the conversations and each turn within a conversation. Each turn contains the user, text and labels. we wrote a C# program to iterate through the turns and labels to create a CSV file with the following information - 1. User, 2. Intent, 3. Origin, 4. Destination, 5. NumberOfAdults, 6. Budget, 7. Date. We used all of the 19986 turns available the original dataset and used roughly 80% of the 15987 data points for training and used the remaining datapoints for testing the data. In order to prepare the CSV file for training, we iterated through each turn and identified the beginning and ending indices of each of the entities. This was then loaded into a Named Entity Recognition pipeline in Spacy.

The file spacy_ner_train.py can be used to replicate the training of the Custom Spacy NER model.

The model training took roughly 16 hrs. and when tested with the test dataset had an accuracy of 0.68 in extracting the relevant features. This is significant because previous research has used the manually labeled entities and this project is the first attempt at using Machine Learning to extract the relevant entities. We also have many values for the same entity i.e. many different city names and not enough datapoints where each value is tagged with the correct entity for example: we have Seattle, San Francisco, Los Angeles, New York and many other words marked as Origins but we do not have enough samples where Seattle is tagged as Origin. We should be able to improve the accuracy of the NER model with additional datapoints.

The table below compares the entities extracted using NLTK, Spacy and our Custom model for a sample turn.

Table 3. Named Entity Recognition of Example Sentence

Token	NLTK	Spacy	Spacy Custom Model
Miami	GPE	GPE	Destination
March 7	NN	DATE	Date
4	CD	CARDINAL	NumberOfAdults
$700	CD	MONEY	Budget

4 Frame Prediction

While the original goal of this project was to predict the frame numbers, we found that we did not have enough data to train a model for the prediction. Although we have a large corpus, each conversation is unique and frame 1 of conversation 1 is not the same as frame 1 of conversation 2. As such, we only have around 15 turns for each conversation and only 1-2 data points for each unique frame. This is not sufficient to train a classification model, so we attempt to answer a different question in this project - how do we predict if the user is switching frames. If we can predict that a user is switching frames, we can use deterministic algorithm to select the correct frame for the current conversation.

5 Frame Switch Prediction

To prepare the dataset for the Frame Switch prediction, we wrote a C# program to parse the frames.json dataset downloaded from the Maluuba website. We then used the named entities provided in the original dataset. The program looped through each conversation and each turn in that conversation keeping track of the previous turn (if any) to create a csv file that contains the following features for each turn: 1.PreviousIntent, 2. PreviousOrigin, 3. PreviousBudget, 4. PreviousDate, 5. PreviousDestination, 6. ConversationId, 7. User, 8. Text, 9. Origin, 10. Destination, 11. Number of Adults, 12. Budget, 13. Date, 14. FrameSwitched. The original dataset does not provide a label for whether the frame has switched so the C# program used the labeled FrameNumber for each turn to determine if the FrameNumber had switched between turns. We use this generated label to score the predictions generated by the various Classification Techniques described below.

5.1 Label Encoding

This project uses text data which requires additional preprocessing before we can use it in Machine Learning libraries such as Sci-kit Learn. We had a few options to convert the text data into numerical data. The simplest model known as Label Encoding works well for small datasets such as ours because we are exploring a restricted set of cities and dates. This model takes all categories and assigns numbers to them. Generally for semantic analysis, OneHotEncoding is used to create a matrix of the labels but that would drastically increase the number of features in our dataset so we chose the simpler LabelEncoding method.

5.2 Data Imbalance

We used a 80/20 split for the training and test data. When we did this, the classes represented in the training data were imbalanced by a ratio of 1:4, with majority of the data of class 0. When we first ran Decision Tree Classification with the data, we achieved high accuracy of 0.71 but all of the data points were labeled as class 0 because of the imbalance. We first attempted to address this by removing all data points where the user is the wizard since we know from the dataset that the wizard cannot change the frame number. When we removed the datapoints for the wizard, our training data was now imbalanced 1:3 with majority of the data of class 1. At this point, our decision tree gave us accuracy of 0.36 with all data points being labeled with class 1.

To address this imbalance, we oversampled class 1 data by 3 i.e. repeated each data point representing class 1 three times in the generated dataset. This step can be seen in the C# program in the MakeDataStructured method.

5.3 Training and Test Data Split

Once we oversampled for Class 1 data, we split the new dataset using 80/20 split keeping 80% of the data for training and 20% of the data for the test. The following sections describe the classifiers used and the results of the comparison.

5.4 Naive Bayes

Naive Bayes uses a simple probabilistic model to classify the data. It assumes that the features are independent of each other. We used it here with a feature selection by wrapper method to find the 7 features that give us the best accuracy.

5.5 Support Vector Machines

Support Vector Machines with a polynomial kernel expects some degree of relation between the various features. Since the SVM model took longer to run, we did not use the full feature selection with wrapper method for this model. We only used 10 iterations to find the best features for this method.

5.6 Decision Tree

Decision Tree is an easy to understand classifier that helps us with missing values, outliers and identifying the most important features. Decision Trees divide the data by a given feature at each step. We used decision trees with feature selection using the wrapper method to identify the 7 most important features that help us predict whether a frame switch has occurred.

5.7 Random Forest

Random Forest is an ensemble model that aims to address the overfitting in decision tree models by using different subsections of the data or using random selection of features. This model produces a collection of trees and aggregates the result of many decision trees to limit overfitting as well as error.

5.8 AdaBoostClassifier

Ada Boost classifier is an ensemble model like Random forest that uses a set of weak classifier algorithms to train a better classifier. The model chooses parts of the training set depending on the accuracy of previous training. The model retrains the classifier at each step to minimize the misclassification rate.

5.9 Artificial Neural Network with Grid Search Cross Validation

Neural Networks use a series of neurons with activation functions and layers to convert the input data into the classification classes. We used a feed forward network that passes data from one layer to another. The key aspect of the neural network is that the weights associated with each input value is adjusted overtime based on the input value and predicted class for the training data. We used Neural Network with Grid Search with Cross Validation to find the best hyper parameters for the training set.

The grid search can be replicated by using the frameSwitch_CV.py file.

5.10 Comparison of the Frame Switch Classifiers

Accuracy Comparison Fig 1 shows the accuracy of the classifiers used. We got the highest accuracy with the Artificial Neural Network with Grid Search with Cross Validation. The lowest accuracy was seen with the Support Vector Machine model.

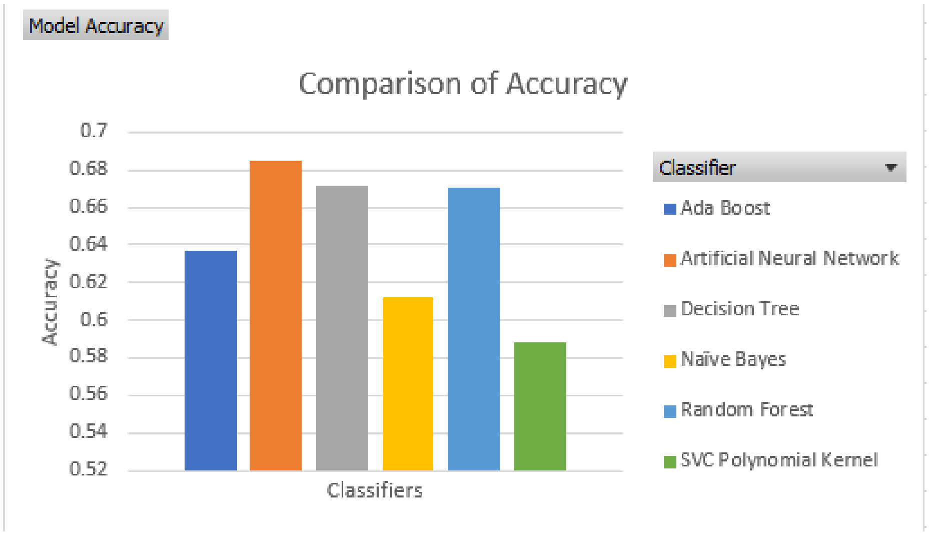

Fig. 1. Accuracy Comparison

Class 0 Precision Comparison Class 0 represents the instances where the frame switch does not occur in our dataset. Fig 2 shows the precision for class 0 determination of the classifiers tested. Overall, we can observe that the Artificial Neural Network had the best precision for Class 0 instances.

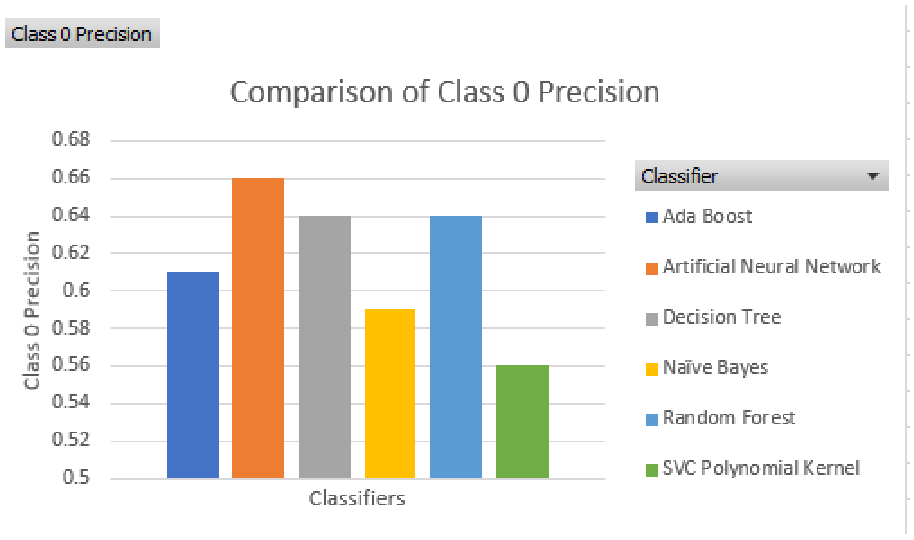

Fig. 2. Class 0 Precision Comparison

Class 1 Precision Comparison Class 1 represents the instances where the frame switch occurs in our dataset. Fig 3 shows the precision for class 1 determination of the classifiers tested. Overall, we can observe that the Random Forest and Decision Tree methods had the best precision for Class 1 instances.

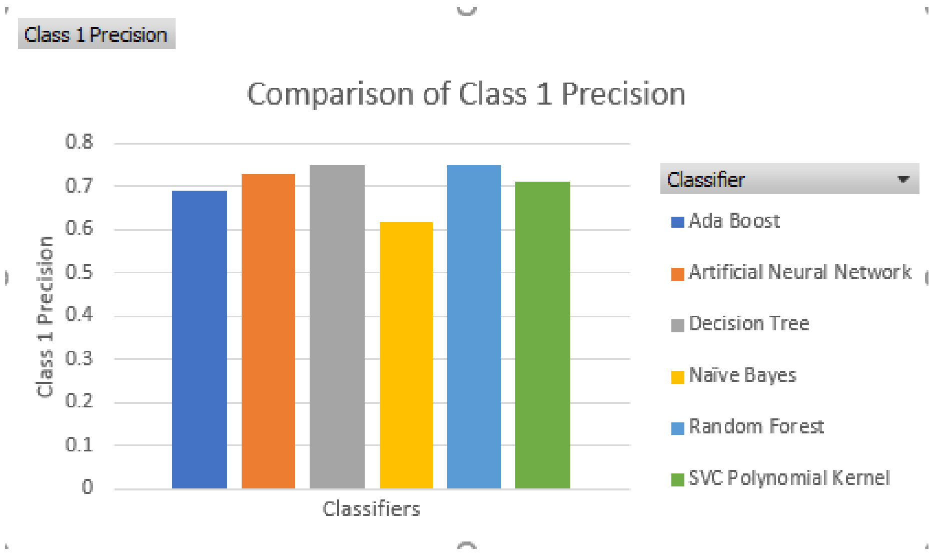

Fig. 3. Class 1 Precision Comparison

Performance Comparison Fig 4 shows the training time of all the classifiers used. SVC Polynomial took the longest time for training (with the least accuracy) while Artificial Neural Network took the second longest training time with the most accuracy. Overall, if training time is an issue, we can use the Decision Tree method whose accuracy was close to the Artificial Neural Network model with a much shorter training time.

R eceiver Operating Characteristic Comparison We compare the ROC curve for the two best performing models: Artificial Neural Networks and Decision Trees below. We can observe from the ROC curves that the two models are very close in terms of accuracy and because the Decision Tree model is faster to train, future work can apply the Decision Tree classifier.

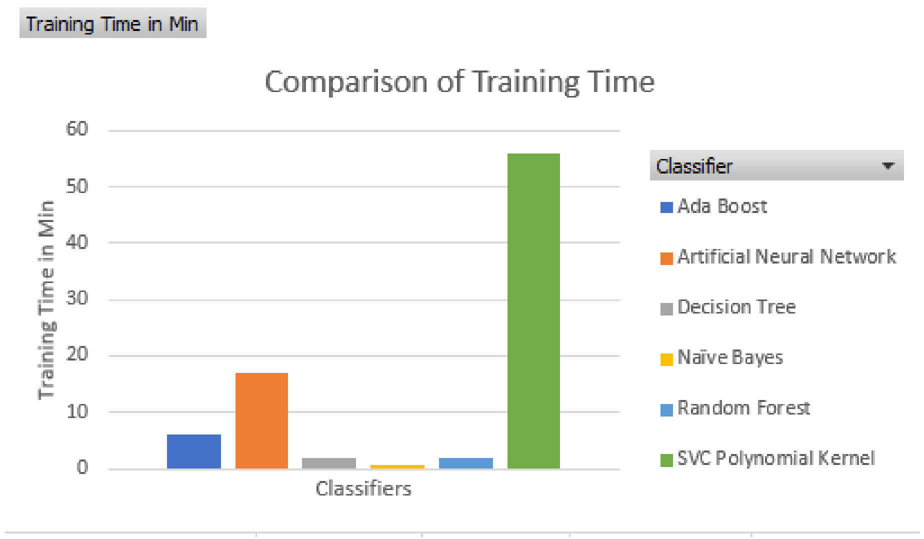

Fig. 4. Training Time Comparison

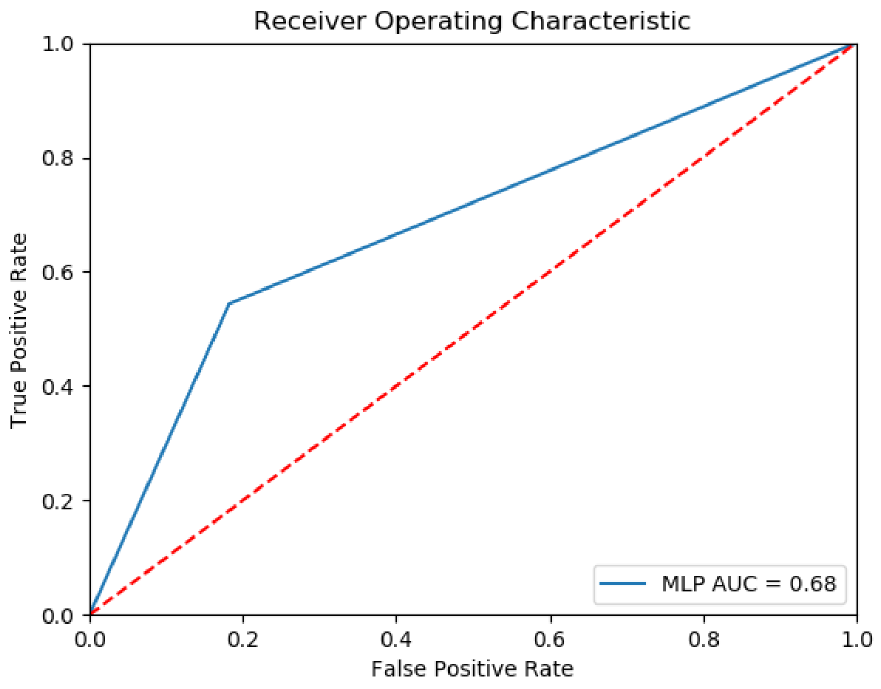

Fig. 5. Neural Network ROC

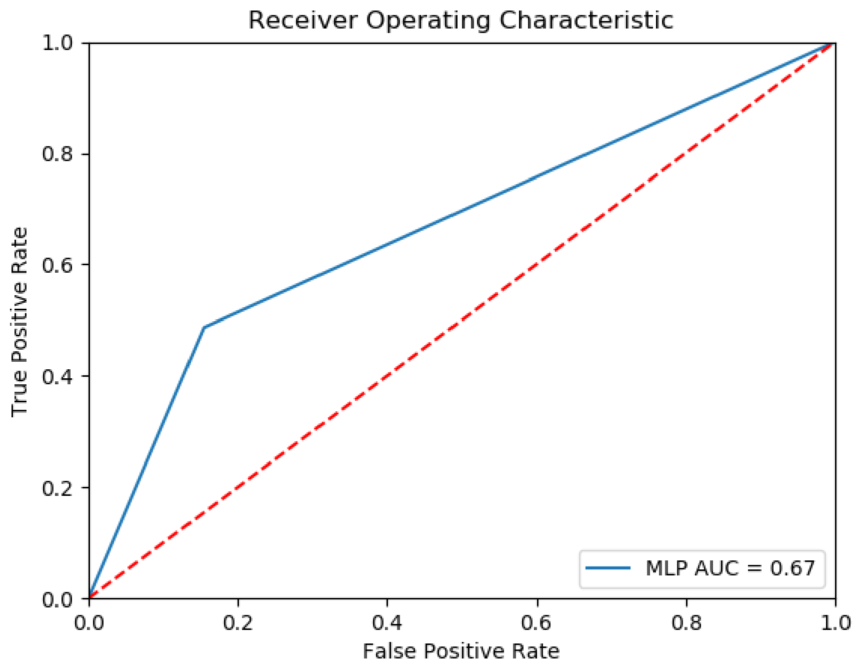

Fig. 6. Decision Tree ROC

6 Conclusion

We were able to achieve the best accuracy for identifying frame switch using Artificial Neural Networks and also using Decision Trees with feature selection by wrapper method. It is interesting to note that Random Forest and our Decision Tree with Feature selection had the same accuracy. This illustrates that the Random Tree method also tries to find the best data subset and features for training multiple decision Trees.
Further, we were able to train a Named Entity Extraction model to identify the constraints that are important for determining if a context switch has occurred. We were able to identify the constraints with an accuracy of 0.68 while previous research relied on the manually labeled constraints. In addition, we were able to classify the text to determine if a frame switch has occurred using the constraint information from the current and previous turns. We were able to reach accuracy of 0.685 for the classification portion of the problem.
While this project focused first on entity extraction then on classifying if a frame switch had occurred, it did not use the data from the entity extraction for the frame switch classification so that classification could be evaluated on it's own. It will be a logical next step to use the extracted entities and compare the classification results with the classification results from the manually tagged entities. Additionally, we did not classify which frame a turn belonged to due to the lack of data. It will be useful to write a deterministic algorithm that can take the result of the frame switch classification and then find the correct frame. In order to predict the frame using machine learning, we will need to collect or generate additional data. It will be worth exploring how to generate the training set for the frame prediction because switching contexts is a very common use case in online chats for e-commerce applications. The project also explored pre-processing with Spell Check, Contraction removal, Stop Word removal and lemmatization but did not include the results of those exercises in the final prediction. A logical next step would be to add the results of those which can easily be obtained by running SpellCheck.SpellCheck and preprocess.py programs.

7 Source Code

Source code to replicate the results of this project are provided in the zip file. The list below explains the included python files and their purpose.

1. preprocess.py - Implements the Contraction and Stop word removal as well as the lemmatization
2. frameSwitch_Ada.py - Implements the Ada booster classifier
3. frameSwitch_CV.py - Implements the Grid Search with Cross Validation for finding the best hyper parameters for the artificial neural network model
4. frameSwitch_DecTree.py - Implements the Decision Tree classifier with Feature Selection by wrapper method
5. frameSwitch_NaiveBayes.py - Implements the Naive Bayes classifier with Feature selection by wrapper method

6. frameSwitch_NN.py - Implements the artificial neural network classifier
7. frameSwitch_RandForest.py - Implements the Random Forest classifier
8. frameSwitch_SVC.py - Implements the Support Vector Machine with polynomial kernel and feature selection by wrapper method
9. NER\ner_compare.py - Small file to compare Entity Extraction using prebuilt NLTK and Spacy models
10. NER\spacy_ner_train.py - Script to train the custom Spacy NER model
11. NER\spacy_ner_test.py - Script to test the trained custom Spacy NER model

In addition to the included python files, the zip file also includes the csproj and .cs files associated with the C# program for data extraction and pre-processing. The C# program includes the following:

1. BingSpellcheck.Spellcheck - Method to correct the spellings of the frame dataset using the Bing Spell check API
2. SpellCheckSchema.cs - Object representation of the Bing Spell Check API result
3. DocumentSchema.cs - Object representation of the Frames dataset JSON
4. Program.MakeStructuredData - Program to generate the CSV file used for classification using oversampling for Class 1
5. Program.CreateNERModel - Program to generate the CSV file used for training and testing the Named Entity Recognition model

Additionally, the zip file also contains the data in JSON and CSV formats.

1. frames.json - Original data [4] downloaded from Maluuba used by the C# program
2. structured.text - CSV file used for Named Entity Recognition training and testing
3. structured_conv.text - CSV file used for training and testing the classifiers

References

1. Horiguchi, S., Inoue, A., Hoshi, T., Okada, K. (2009). GaChat:A chat system that displays online retrieval information in dialogue text. CEUR Workshop Proceedings, 443.
2. Asri, L., Schulz, H., Sharma, S., Zumer, J., Harris, J., Fine, E., . . . Suleman, K. (2017). Frames: A Corpus for Adding Memory to Goal-Oriented Dialogue Systems.
3. El Asri, L., Lemonnier, R., Laroche, R., Khouzaimi, H., Pietquin, O. (2014). NASTIA: Negotiating appointment setting interface. Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014, 266-271. 4 Oct 2017
4. Frames — Maluuba. (2018). Retrieved from https://datasets.maluuba.com/Frames
5. Authenticate to Bing Speech. (2018). Retrieved from https://docs.microsoft.com/en-us/azure/cognitive-services/speech/how-to/how-to-authentication?tabs=CSharp
6. Williams, J., Raux, A., Ramachandran, D., Black, A. (2013). The dialog state tracking challenge. In Proceedings of the SIGDIAL 2013 Conference (pp. 404-413).
7. Paek, T., Horvitz, E. (2000, June). Conversation as action under uncertainty. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence (pp. 455-464). Morgan Kaufmann Publishers Inc..
8. Schulz, H., Zumer, J., Asri, L. E., Sharma, S. (2017). A frame tracking model for memory-enhanced dialogue systems. arXiv preprint arXiv:1706.01690.